DocBook XSL Stylesheets: Reference Documentation

Norman Walsh
$Id: reference.xml 7295 2007-08-28 09:15:14Z xmldoc $

License

Warranty

About this document

This is generated reference documentation for the DocBook XSL stylesheets. It is available in the following formats:

- HTML[1], PDF[2], plain text[3]

This is primarily documentation on the parameters you can adjust to control the behavior of the stylesheets.

Note

This is purely reference documentation – not how-to documentation. For a thorough step-by-step how-to guide to publishing content using the DocBook XSL stylesheets, see Bob Stayton’s DocBook XSL: The Complete Guide[4], available online at http://www.sagehill.net/docbookxsl/index.html

[1] http://docbook.sourceforge.net/release/xsl/current/doc/reference.html
[2] http://docbook.sourceforge.net/release/xsl/current/doc/reference.pdf
[3] http://docbook.sourceforge.net/release/xsl/current/doc/reference.txt
[4] http://www.sagehill.net/book-description.html

This document is divided into three sets of references: the first two sets provides user documentation; the third, developer documentation.

Table of Contents

DocBook XSL Stylesheets
User Reference: Parameters

DocBook XSL Stylesheets User Reference: Parameters

Abstract

This is generated reference documentation for all user-configurable parameters in the DocBook XSL stylesheets.

Note

This is purely reference documentation – not how-to documentation. For a thorough step-by-step how-to guide to publishing content using the DocBook XSL stylesheets, see Bob Stayton's DocBook XSL: The Complete Guide[5], available online at http://www.sagehill.net/docbookxsl/index.html

[5] http://www.sagehill.net/book-description.html

Table of Contents

List of Figures

Part I. HTML Parameter Reference

This is reference documentation for all user-configurable parameters in the DocBook XSL HTML stylesheets (for generating HTML output).

Admonitions

Name

admon.graphics.extension — Extension for admonition graphics

Synopsis

```
<xsl:param name="admon.graphics.extension">.png</xsl:param>
```

Description

Sets the extension to use on admonition graphics.

Name

admon.graphics.path — Path to admonition graphics

Synopsis

```
<xsl:param name="admon.graphics.path">images/</xsl:param>
```

Description

Sets the path to the directory containing the admonition graphics (caution.png, important.png etc). This location is normally relative to the output html directory. See *`base.dir`*

Name

admon.graphics — Use graphics in admonitions?

Synopsis

```
<xsl:param name="admon.graphics" select="0"></xsl:param>
```

Description

If true (non-zero), admonitions are presented in an alternate style that uses a graphic. Default graphics are provided in the distribution.

Name

admon.textlabel — Use text label in admonitions?

Synopsis

```
<xsl:param name="admon.textlabel" select="1"></xsl:param>
```

Description

If true (non-zero), admonitions are presented with a generated text label such as Note or Warning in the appropriate language. If zero, such labels are turned off, but any title child of the admonition element are still output. The default value is 1.

Name

admon.style — Specifies the CSS style attribute that should be added to admonitions.

Synopsis

```
<xsl:param name="admon.style">
  <xsl:text>margin-left: 0.5in; margin-right: 0.5in;</xsl:text>
</xsl:param>
```

Description

Specifies the value of the CSS `style` attribute that should be added to admonitions.

Callouts

Name

callout.defaultcolumn — Indicates what column callouts appear in by default

Synopsis

```
<xsl:param name="callout.defaultcolumn">60</xsl:param>
```

Description

If a callout does not identify a column (for example, if it uses the `linerange unit`), it will appear in the default column.

Name

callout.graphics.extension — File name extension for callout graphics

Synopsis

```
<xsl:param name="callout.graphics.extension">.png</xsl:param>
```

Description

Sets the extension to use on callout graphics, hence the callout graphic format. The appropriate format (and range used) should be available. svg, png and gif are provided.

Name

callout.graphics.number.limit — Number of the largest callout graphic

Synopsis

```
<xsl:param name="callout.graphics.number.limit">15</xsl:param>
```

Description

If *`callout.graphics`* is non-zero, graphics are used to represent callout numbers instead of plain text. The value of *`callout.graphics.number.limit`* is the largest number for which a graphic exists. If the callout number exceeds this limit, the default presentation "(plain text instead of a graphic)" will be used.

Name

callout.graphics.path — Path to callout graphics

Synopsis

```
<xsl:param name="callout.graphics.path">images/callouts/</xsl:param>
```

Description

Sets the path to the directory holding the callout graphics. his location is normally relative to the output html directory. see base.dir. Always terminate the directory with / since the graphic file is appended to this string, hence needs the separator.

Name

callout.graphics — Use graphics for callouts?

Synopsis

```
<xsl:param name="callout.graphics" select="1"></xsl:param>
```

Description

If non-zero, callouts are presented with graphics (e.g., reverse-video circled numbers instead of "(1)", "(2)", etc.). Default graphics are provided in the distribution.

Name

callout.list.table — Present callout lists using a table?

Synopsis

```
<xsl:param name="callout.list.table" select="1"></xsl:param>
```

Description

The default presentation of `CalloutList`s uses an HTML `DL`. Some browsers don't align DLs very well if *`callout.graphics`* are used. With this option turned on, `CalloutList`s are presented in an HTML `TABLE`, which usually results in better alignment of the callout number with the callout description.

Name

callout.unicode.number.limit — Number of the largest unicode callout character

Synopsis

```
<xsl:param name="callout.unicode.number.limit">10</xsl:param>
```

Description

If *`callout.unicode`* is non-zero, unicode characters are used to represent callout numbers. The value of *`callout.unicode.number.limit`* is the largest number for which a unicode character exists. If the callout number exceeds this limit, the default presentation "(nnn)" will always be used.

Name

callout.unicode.start.character — First Unicode character to use, decimal value.

Synopsis

```
<xsl:param name="callout.unicode.start.character">10102</xsl:param>
```

Description

If *`callout.graphics`* is zero and *`callout.unicode`* is non-zero, unicode characters are used to represent callout numbers. The value of *`callout.unicode.start.character`* is the decimal unicode value used for callout number one. Currently, only 10102 is supported in the stylesheets for this parameter.

Name

callout.unicode — Use Unicode characters rather than images for callouts.

Synopsis

```
<xsl:param name="callout.unicode" select="0"></xsl:param>
```

Description

The stylesheets can use either an image of the numbers one to ten, or the single Unicode character which represents the numeral, in white on a black background. Use this to select the Unicode character option.

Name

callouts.extension — Enable the callout extension

Synopsis

```
<xsl:param name="callouts.extension" select="1"></xsl:param>
```

Description

The callouts extension processes `areaset` elements in `ProgramListingCO` and other text-based callout elements.

EBNF

Name

ebnf.table.bgcolor — Background color for EBNF tables

Synopsis

```
<xsl:param name="ebnf.table.bgcolor">#F5DCB3</xsl:param>
```

Description

Sets the background color for EBNF tables (a pale brown). No `bgcolor` attribute is output if *`ebnf.table.bgcolor`* is set to the null string.

Name

ebnf.table.border — Selects border on EBNF tables

Synopsis

```
<xsl:param name="ebnf.table.border" select="1"></xsl:param>
```

Description

Selects the border on EBNF tables. If non-zero, the tables have borders, otherwise they don't.

Name

ebnf.assignment — The EBNF production assignment operator

Synopsis

```
<xsl:param name="ebnf.assignment">
<code>::=</code>
</xsl:param>
```

Description

The *`ebnf.assignment`* parameter determines what text is used to show “assignment” in `production`s in `productionset`s.

While “`::=`” is common, so are several other operators.

Name

ebnf.statement.terminator — Punctuation that ends an EBNF statement.

Synopsis

```
<xsl:param name="ebnf.statement.terminator"></xsl:param>
```

Description

The *`ebnf.statement.terminator`* parameter determines what text is used to terminate each `production` in `productionset`.

Some notations end each statement with a period.

ToC/LoT/Index Generation

Name

annotate.toc — Annotate the Table of Contents?

Synopsis

```
<xsl:param name="annotate.toc" select="1"></xsl:param>
```

Description

If true, TOCs will be annotated. At present, this just means that the `RefPurpose` of `RefEntry` TOC entries will be displayed.

Name

autotoc.label.separator — Separator between labels and titles in the ToC

Synopsis

```
<xsl:param name="autotoc.label.separator">. </xsl:param>
```

Description

String to use to seperate labels and title in a table of contents.

Name

autotoc.label.in.hyperlink — Include label in hyperlinked titles in TOC?

Synopsis

```
<xsl:param name="autotoc.label.in.hyperlink" select="1"></xsl:param>
```

Description

If the value of *`autotoc.label.in.hyperlink`* is non-zero, labels are included in hyperlinked titles in the TOC. If it is instead zero, labels are still displayed prior to the hyperlinked titles, but are not hyperlinked along with the titles.

Name

process.source.toc — Process a non-empty `toc` element if it occurs in a source document?

Synopsis

```
<xsl:param name="process.source.toc" select="0"></xsl:param>
```

Description

Specifies that the contents of a non-empty "hard-coded" `toc` element in a source document are processed to generate a TOC in output.

Note

This parameter has no effect on automated generation of TOCs. An automated TOC may still be generated along with the "hard-coded" TOC. To suppress automated TOC generation, adjust the value of the *`generate.toc`* paramameter.

The `process.source.toc` parameter also has no effect if the `toc` element is empty; handling for empty `toc` is controlled by the *`process.empty.source.toc`* parameter.

Name

process.empty.source.toc — Generate automated TOC if `toc` element occurs in a source document?

Synopsis

```
<xsl:param name="process.empty.source.toc" select="0"></xsl:param>
```

Description

Specifies that if an empty `toc` element is found in a source document, an automated TOC is generated at this point in the document.

Note

Depending on what the value of the *`generate.toc`* parameter is, setting this parameter to `1` could result in generation of duplicate automated TOCs. So the *`process.empty.source.toc`* is primarily useful as an "override": by placing an empty `toc` in your document and setting this parameter to `1`, you can force a TOC to be generated even if `generate.toc` says not to.

Name

bridgehead.in.toc — Should bridgehead elements appear in the TOC?

Synopsis

```
<xsl:param name="bridgehead.in.toc" select="0"></xsl:param>
```

Description

If non-zero, `bridgehead`s appear in the TOC. Note that this option is not fully supported and may be removed in a future version of the stylesheets.

Name

simplesect.in.toc — Should `simplesect` elements appear in the TOC?

Synopsis

```
<xsl:param name="simplesect.in.toc" select="0"></xsl:param>
```

Description

If non-zero, `simplesect`s will be included in the TOC.

Name

manual.toc — An explicit TOC to be used for the TOC

Synopsis

```
<xsl:param name="manual.toc"></xsl:param>
```

Description

The *`manual.toc`* identifies an explicit TOC that will be used for building the printed TOC.

Name

toc.list.type — Type of HTML list element to use for Tables of Contents

Synopsis

```
<xsl:param name="toc.list.type">dl</xsl:param>
```

Description

When an automatically generated Table of Contents (or List of Titles) is produced, this HTML element will be used to make the list.

Name

toc.section.depth — How deep should recursive `section`s appear in the TOC?

Synopsis

```
<xsl:param name="toc.section.depth">2</xsl:param>
```

Description

Specifies the depth to which recursive sections should appear in the TOC.

Name

toc.max.depth — How many levels should be created for each TOC?

Synopsis

```
<xsl:param name="toc.max.depth">8</xsl:param>
```

Description

Specifies the maximal depth of TOC on all levels.

Name

generate.toc — Control generation of ToCs and LoTs

Synopsis

```
<xsl:param name="generate.toc">
appendix  toc,title
article/appendix  nop
article   toc,title
book      toc,title,figure,table,example,equation
chapter   toc,title
part      toc,title
preface   toc,title
qandadiv  toc
qandaset  toc
reference toc,title
sect1     toc
sect2     toc
sect3     toc
sect4     toc
sect5     toc
section   toc
set       toc,title
</xsl:param>
```

Description

This parameter has a structured value. It is a table of space-delimited path/value pairs. Each path identifies some element in the source document using a restricted subset of XPath (only the implicit child axis, no wildcards, no predicates). Paths can be either relative or absolute.

When processing a particular element, the stylesheets consult this table to determine if a ToC (or LoT(s)) should be generated.

For example, consider the entry:

```
book toc,figure
```

This indicates that whenever a `book` is formatted, a Table Of Contents and a List of Figures should be generated. Similarly,

```
/chapter toc
```

indicates that whenever a document *that has a root of* `chapter` is formatted, a Table of Contents should be generated. The entry `chapter` would match all chapters, but `/chapter` matches only `chapter` document elements.

Generally, the longest match wins. So, for example, if you want to distinguish articles in books from articles in parts, you could use these two entries:

```
book/article toc,figure
part/article toc
```

Note that an article in a part can never match a `book/article`, so if you want nothing to be generated for articles in parts, you can simply leave that rule out.

If you want to leave the rule in, to make it explicit that you're turning something off, use the value “nop”. For example, the following entry disables ToCs and LoTs for articles:

```
article nop
```

Do not simply leave the word “article” in the file without a matching value. That'd be just begging the silly little path/value parser to get confused.

Section ToCs are further controlled by the *`generate.section.toc.level`* parameter. For a given section level to have a ToC, it must have both an entry in *`generate.toc`* and be within the range enabled by *`generate.section.toc.level`*.

Name

generate.section.toc.level — Control depth of TOC generation in sections

Synopsis

```
<xsl:param name="generate.section.toc.level" select="0"></xsl:param>
```

Description

The *`generate.section.toc.level`* parameter controls the depth of `section` in which TOCs will be generated. Note that this is related to, but not the same as *`toc.section.depth`*, which controls the depth to which TOC entries will be generated in a given TOC.

If, for example, *`generate.section.toc.level`* is `3`, TOCs will be generated in first, second, and third level sections, but not in fourth level sections.

Name

generate.index — Do you want an index?

Synopsis

```
<xsl:param name="generate.index" select="1"></xsl:param>
```

Description

Specify if an index should be generated.

Name

index.method — Select method used to group index entries in an index

Synopsis

```
<xsl:param name="index.method">basic</xsl:param>
```

Description

This parameter lets you select which method should be used to sort and group index entries in an index. Indexes in latin-based languages that have accented characters typically sort together accented words and unaccented words. Thus “Á” (A acute) would sort together with “A”, so both would appear in the “A” section of the index. Languages using other alphabets (such as Russian cyrillic) and languages using ideographic chararacters (such as Japanese) require grouping specific to the languages and alphabets.

The default indexing method is limited. It can group accented characters in latin-based languages only. It cannot handle non-latin alphabets or ideographic languages. The other indexing methods require extensions of one type or another, and do not work with all XSLT processors, which is why there are not used by default.

The three choices for indexing method are:

`basic`
: (default) Sort and groups words based only on the Latin alphabet. Words with accented latin letters will group and sort with their respective primary letter, but words in non-Latin alphabets will be put in the “Symbols” section of the index.

`kosek`
: Sort and groups words based on letter groups configured in the DocBook locale file for the given language. See, for example, the French locale file `common/fr.xml`. This method requires that the XSLT processor support the EXSLT extensions (most do). It also requires support for using user-defined functions in xsl:key (xsltproc does not).

  This method is suitable for any language for which you can list all the individual characters that should appear in each letter group in an index. It is probably not practical to use it for ideographic languages such as Chinese that have hundreds or thousands of characters.

  To use the kosek method, you must:

  1. Use a processor that supports its extensions, such as Saxon 6 or Xalan (xsltproc and Saxon 8 do not).

  2. Set the index.method parameter's value to “kosek”.

  3. Import the appropriate index extensions stylesheet module `fo/autoidx-kosek.xsl` or `html/autoidx-kosek.xsl` into your customization.

`kimber`
: This method uses extensions to the Saxon processor to implement sophisticated indexing processes. It uses its own configuration file, which can include information for any number of languages. Each language's configuration can group words using one of two processes. In the enumerated process similar to that used in the kosek method, you indicate the groupings character-by-character. In the between-key process, you specify the break-points in the sort order that should start a new group. The latter configuration is useful for ideographic languages such as Chinese, Japanese, and Korean. You can also define your own collation algorithms and how you want mixed Latin-alphabet words sorted.

  - For a whitepaper describing the extensions, see: http://www.innodata-isogen.com/knowledge_center/white_papers/back_of_book_for_xsl_fo.pdf.
  - To download the extension library, see http://www.innodata-isogen.com/knowledge_center/tools_downloads/i18nsupport.

  To use the kimber method, you must:

  1. Use Saxon (version 6 or 8) as your XSLT processor.
  2. Install and configure the Innodata Isogen library, using the documentation that comes with it.
  3. Set the index.method parameter's value to “kimber”.
  4. Import the appropriate index extensions stylesheet module `fo/autoidx-kimber.xsl` or `html/autoidx-kimber.xsl` into your customization.

Name

index.on.type — Select indexterms based on `type` attribute value

Synopsis

```
<xsl:param name="index.on.type" select="0"></xsl:param>
```

Description

If non-zero, then an `index` element that has a `type` attribute value will contain only those `indexterm` elements with a matching `type` attribute value. If an `index` has no `type` attribute or it is blank, then the index will contain all `indexterm`s in the current scope.

If `index.on.type` is zero, then the `type` attribute has no effect on selecting indexterms for an index.

For those using DocBook version 4.2 or earlier, the `type` attribute is not available for index terms. However, you can achieve the same effect by using the `role` attribute in the same manner on `indexterm` and `index`, and setting the stylesheet parameter *`index.on.role`* to a nonzero value.

Name

index.on.role — Select indexterms based on `role` value

Synopsis

```
<xsl:param name="index.on.role" select="0"></xsl:param>
```

Description

If non-zero, then an `index` element that has a `role` attribute value will contain only those `indexterm` elements with a matching role value. If an `index` has no `role` attribute or it is blank, then the index will contain all `indexterm`s in the current scope.

If `index.on.role` is zero, then the `role` attribute has no effect on selecting indexterms for an index.

If you are using DocBook version 4.3 or later, you should use the `type` attribute instead of `role` on `indexterm` and `index`, and set the *`index.on.type`* to a nonzero value.

Name

index.prefer.titleabbrev — Should abbreviated titles be used as back references?

Synopsis

```
<xsl:param name="index.prefer.titleabbrev" select="0"></xsl:param>
```

Description

If non-zero, and if a `titleabbrev` is defined, the abbreviated title is used as the link text of a back reference in the index.

Name

index.term.separator — Override for punctuation separating an index term from its list of page references in an index

Synopsis

```
<xsl:param name="index.term.separator"></xsl:param>
```

Description

This parameter permits you to override the text to insert between the end of an index term and its list of page references. Typically that might be a comma and a space.

Because this text may be locale dependent, this parameter's value is normally taken from a gentext template named 'term-separator' in the context 'index' in the stylesheet locale file for the language of the current document. This parameter can be used to override the gentext string, and would typically be used on the command line. This parameter would apply to all languages.

So this text string can be customized in two ways. You can reset the default gentext string using the *`local.l10n.xml`* parameter, or you can fill in the content for this normally empty override parameter. The content can be a simple string, or it can be something more complex such as a call-template. For fo output, it could be an `fo:leader` element to provide space of a specific length, or a dot leader.

Name

index.number.separator — Override for punctuation separating page numbers in index

Synopsis

```
<xsl:param name="index.number.separator"></xsl:param>
```

Description

This parameter permits you to override the text to insert between page references in a formatted index entry. Typically that would be a comma and a space.

Because this text may be locale dependent, this parameter's value is normally taken from a gentext template named 'number-separator' in the context 'index' in the stylesheet locale file for the language of the current document. This parameter can be used to override the gentext string, and would typically be used on the command line. This parameter would apply to all languages.

So this text string can be customized in two ways. You can reset the default gentext string using the *local.l10n.xml* parameter, or you can override the gentext with the content of this parameter. The content can be a simple string, or it can be something more complex such as a call-template.

In HTML index output, section title references are used instead of page number references. This punctuation appears between such section titles in an HTML index.

Name

index.range.separator — Override for punctuation separating the two numbers in a page range in index

Synopsis

```
<xsl:param name="index.range.separator"></xsl:param>
```

Description

This parameter permits you to override the text to insert between the two numbers of a page range in an index. This parameter is only used by those XSL-FO processors that support an extension for generating such page ranges (such as XEP).

Because this text may be locale dependent, this parameter's value is normally taken from a gentext template named 'range-separator' in the context 'index' in the stylesheet locale file for the language of the current document. This parameter can be used to override the gentext string, and would typically be used on the command line. This parameter would apply to all languages.

So this text string can be customized in two ways. You can reset the default gentext string using the *local.l10n.xml* parameter, or you can override the gentext with the content of this parameter. The content can be a simple string, or it can be something more complex such as a call-template.

In HTML index output, section title references are used instead of page number references. So there are no page ranges and this parameter has no effect.

Stylesheet Extensions

Name

linenumbering.everyNth — Indicate which lines should be numbered

Synopsis

```
<xsl:param name="linenumbering.everyNth">5</xsl:param>
```

Description

If line numbering is enabled, everyNth line will be numbered. Note that numbering is one based, not zero based.

Name

linenumbering.extension — Enable the line numbering extension

Synopsis

```
<xsl:param name="linenumbering.extension" select="1"></xsl:param>
```

Description

If non-zero, verbatim environments (elements that have the format='linespecific' notation attribute: address, literallayout, programlisting, screen, synopsis) that specify line numbering will have, line numbers.

Name

linenumbering.separator — Specify a separator between line numbers and lines

Synopsis

```
<xsl:param name="linenumbering.separator"><xsl:text> </xsl:text></xsl:param>
```

Description

The separator is inserted between line numbers and lines in the verbatim environment. The default value is a single white space. Note the interaction with *`linenumbering.width`*

Name

linenumbering.width — Indicates the width of line numbers

Synopsis

```
<xsl:param name="linenumbering.width">3</xsl:param>
```

Description

If line numbering is enabled, line numbers will appear right justified in a field "width" characters wide.

Name

tablecolumns.extension — Enable the table columns extension function

Synopsis

```
<xsl:param name="tablecolumns.extension" select="1"></xsl:param>
```

Description

The table columns extension function adjusts the widths of table columns in the HTML result to more accurately reflect the specifications in the CALS table.

Name

textinsert.extension — Enables the textinsert extension element

Synopsis

```
<xsl:param name="textinsert.extension" select="1"></xsl:param>
```

Description

The textinsert extension element inserts the contents of a file into the result tree (as text).

Note

To use the textinsert extension element, you must use either Saxon or Xalan as your XSLT processor (it doesn't work with xsltproc), along with either the DocBook Saxon extensions or DocBook Xalan extensions (for more information about those extensions, see DocBook XSL: TCG, DocBook Saxon Extensions[1] and DocBook XSL: TCG, DocBook Xalan Extensions[2]), and you must set both the *`use.extensions`* and *`textinsert.extension`* parameters to `1`.

As an alternative to using the textinsert element, consider using an Xinclude element with the `parse="text"` attribute and value specified, as detailed in DocBook XSL: TCG, Using XInclude for text inclusions[3].

See Also

You can also use the `<?dbhtml-include href?>` processing instruction to insert external files — both files containing plain text and files with markup content (including HTML content).

More information

For how-to documentation on inserting contents of external code files and other text files into output, see DocBook XSL: TCG, External code files[4].

For guidelines on inserting contents of HTML files into output, see DocBook XSL: TCG, Inserting external HTML code[5].

Name

textdata.default.encoding — Default encoding of external text files which are included using textdata element

Synopsis

```
<xsl:param name="textdata.default.encoding"></xsl:param>
```

[1] http://www.sagehill.net/docbookxsl/InstallingAProcessor.html#SaxonExtensions
[2] http://www.sagehill.net/docbookxsl/InstallingAProcessor.html#XalanExtensions
[3] http://www.sagehill.net/docbookxsl/ExternalCode.html#XIncludeCode
[4] http://www.sagehill.net/docbookxsl/ExternalCode.html
[5] http://www.sagehill.net/docbookxsl/InsertExtHtml.html

Description

Specifies the encoding of any external text files included using `textdata` element. This value is used only when you do not specify encoding by the appropriate attribute directly on textdata. An empty string is interpreted as the system default encoding.

Name

graphicsize.extension — Enable the getWidth()/getDepth() extension functions

Synopsis

```
<xsl:param name="graphicsize.extension" select="1"></xsl:param>
```

Description

If non-zero (and if *`use.extensions`* is non-zero and if you're using a processor that supports extension functions), the `getWidth` and `getDepth` functions will be used to extract image sizes from graphics.

Name

graphicsize.use.img.src.path — Prepend *`img.src.path`* before filenames passed to extension functions

Synopsis

```
<xsl:param name="graphicsize.use.img.src.path" select="0"></xsl:param>
```

Description

If non-zero *`img.src.path`* parameter will be appended before filenames passed to extension functions for measuring image dimensions.

Name

use.extensions — Enable extensions

Synopsis

```
<xsl:param name="use.extensions" select="0"></xsl:param>
```

Description

If non-zero, extensions may be used. Each extension is further controlled by its own parameter. But if *`use.extensions`* is zero, no extensions will be used.

Automatic labelling

Name

chapter.autolabel — Specifies the labeling format for Chapter titles

Synopsis

```
<xsl:param name="chapter.autolabel" select="1"></xsl:param>
```

Description

If non-zero, then chapters will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (arabic).

Name

appendix.autolabel — Specifies the labeling format for Appendix titles

Synopsis

```
<xsl:param name="appendix.autolabel">A</xsl:param>
```

Description

If non-zero, then appendices will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (upperalpha).

Name

part.autolabel — Specifies the labeling format for Part titles

Synopsis

```
<xsl:param name="part.autolabel">I</xsl:param>
```

Description

If non-zero, then parts will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (upperroman).

Name

reference.autolabel — Specifies the labeling format for Reference titles

Synopsis

```
<xsl:param name="reference.autolabel">I</xsl:param>
```

Description

If non-zero, references will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any non-zero value other than the above will generate the default number format (upperroman).

Name

preface.autolabel — Specifices the labeling format for Preface titles

Synopsis

```
<xsl:param name="preface.autolabel" select="0"></xsl:param>
```

Description

If non-zero then prefaces will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (arabic).

Name

qandadiv.autolabel — Are divisions in QAndASets enumerated?

Synopsis

```
<xsl:param name="qandadiv.autolabel" select="1"></xsl:param>
```

Description

If non-zero, unlabeled qandadivs will be enumerated.

Name

section.autolabel — Are sections enumerated?

Synopsis

```
<xsl:param name="section.autolabel" select="0"></xsl:param>
```

Description

If true (non-zero), unlabeled sections will be enumerated.

Name

section.autolabel.max.depth — The deepest level of sections that are numbered.

Synopsis

```
<xsl:param name="section.autolabel.max.depth">8</xsl:param>
```

Description

When section numbering is turned on by the *section.autolabel* parameter, then this parameter controls the depth of section nesting that is numbered. Sections nested to a level deeper than this value will not be numbered.

Name

section.label.includes.component.label — Do section labels include the component label?

Synopsis

```
<xsl:param name="section.label.includes.component.label" select="0"></xsl:param>
```

Description

If non-zero, section labels are prefixed with the label of the component that contains them.

Name

label.from.part — Renumber components in each part?

Synopsis

```
<xsl:param name="label.from.part" select="0"></xsl:param>
```

Description

If *label.from.part* is non-zero, then numbering of components — preface, chapter, appendix, and reference (when reference occurs at the component level) — is re-started within each part.

If *label.from.part* is zero (the default), numbering of components is *not* re-started within each part; instead, components are numbered sequentially throughout each book, regardless of whether or not they occur within part instances.

Name

component.label.includes.part.label — Do component labels include the part label?

Synopsis

```
<xsl:param name="component.label.includes.part.label" select="0"></xsl:param>
```

Description

If non-zero, number labels for chapter, appendix, and other component elements are prefixed with the label of the part element that contains them. So you might see Chapter II.3 instead of Chapter 3. Also, the labels for formal elements such as table and figure will include the part label. If there is no part element container, then no prefix is generated.

This feature is most useful when the *label.from.part* parameter is turned on. In that case, there would be more than one chapter “1”, and the extra part label prefix will identify each chapter unambiguously.

HTML

Name

html.base — An HTML base URI

Synopsis

```
<xsl:param name="html.base"></xsl:param>
```

Description

If html.base is set, it is used for the `base` element in the `head` of the html documents. The parameter specifies the base URL for all relative URLs in the document. This is useful for dynamically served html where the base URI needs to be shifted.

Name

html.stylesheet.type — The type of the stylesheet used in the generated HTML

Synopsis

```
<xsl:param name="html.stylesheet.type">text/css</xsl:param>
```

Description

The type of the stylesheet to place in the HTML `link` tag.

Name

html.stylesheet — Name of the stylesheet(s) to use in the generated HTML

Synopsis

```
<xsl:param name="html.stylesheet"></xsl:param>
```

Description

The *`html.stylesheet`* parameter is either empty, indicating that no stylesheet `link` tag should be generated in the html output, or it is a list of one or more stylesheet files.

Multiple stylesheets are space-delimited. If you need to reference a stylesheet URI that includes a space, encode it with `%20`. A separate html `link` element will be generated for each stylesheet in the order they are listed in the parameter.

Name

css.decoration — Enable CSS decoration of elements

Synopsis

```
<xsl:param name="css.decoration" select="1"></xsl:param>
```

Description

If non-zero, then html elements produced by the stylesheet may be decorated with `style` attributes. For example, the `li` tags produced for list items may include a fragment of CSS in the `style` attribute which sets the CSS property "list-style-type".

Name

spacing.paras — Insert additional <p> elements for spacing?

Synopsis

```
<xsl:param name="spacing.paras" select="0"></xsl:param>
```

Description

When non-zero, additional, empty paragraphs are inserted in several contexts (for example, around informal figures), to create a more pleasing visual appearance in many browsers.

Name

emphasis.propagates.style — Pass emphasis role attribute through to HTML?

Synopsis

```
<xsl:param name="emphasis.propagates.style" select="1"></xsl:param>
```

Description

If non-zero, the `role` attribute of `emphasis` elements will be passed through to the HTML as a class attribute on a `span` that surrounds the emphasis.

Name

para.propagates.style — Pass para role attribute through to HTML?

Synopsis

```
<xsl:param name="para.propagates.style" select="1"></xsl:param>
```

Description

If true, the role attribute of `para` elements will be passed through to the HTML as a class attribute on the `p` generated for the paragraph.

Name

phrase.propagates.style — Pass phrase role attribute through to HTML?

Synopsis

```
<xsl:param name="phrase.propagates.style" select="1"></xsl:param>
```

Description

If non-zero, the role attribute of `phrase` elements will be passed through to the HTML as a `class` attribute on a `span` that surrounds the phrase.

Name

entry.propagates.style — Pass entry role attribute through to HTML?

Synopsis

```
<xsl:param name="entry.propagates.style" select="1"></xsl:param>
```

Description

If true, the role attribute of `entry` elements will be passed through to the HTML as a class attribute on the `td` or `th` generated for the table cell.

Name

html.longdesc — Should longdesc URIs be created?

Synopsis

```
<xsl:param name="html.longdesc" select="1"></xsl:param>
```

Description

If non-zero, HTML files will be created for the `longdesc` attribute. These files are created from the `textobject`s in `mediaobject`s and `inlinemediaobject`.

Name

html.longdesc.link — Should a link to the longdesc be included in the HTML?

Synopsis

```
<xsl:param name="html.longdesc.link" select="$html.longdesc"></xsl:param>
```

Description

If non-zero, links will be created to the HTML files created for the `longdesc` attribute. It makes no sense to turn enable this option without also enabling the *`html.longdesc`* parameter.

Name

make.valid.html — Attempt to make sure the HTML output is valid HTML

Synopsis

```
<xsl:param name="make.valid.html" select="0"></xsl:param>
```

Description

If *`make.valid.html`* is true, the stylesheets take extra effort to ensure that the resulting HTML is valid. This may mean that some `para` tags are translated into HTML `div`s or that other substitutions occur.

This parameter is different from *`html.cleanup`* because it changes the resulting markup; it does not use extension functions to manipulate result-tree-fragments and is therefore applicable to any XSLT processor.

Name

html.cleanup — Attempt to clean up the resulting HTML?

Synopsis

```
<xsl:param name="html.cleanup" select="1"></xsl:param>
```

Description

If non-zero, and if the EXSLT[1] extensions are supported by your processor, the resulting HTML will be "cleaned up". This improves the chances that the resulting HTML will be valid. It may also improve the formatting of some elements.

This parameter is different from *`make.valid.html`* because it uses extension functions to manipulate result-tree-fragments.

Name

html.append — Specifies content to append to HTML output

Synopsis

```
<xsl:param name="html.append"></xsl:param>
```

Description

Specifies content to append to the end of HTML files output by the `html/docbook.xsl` stylesheet, after the closing `<html>` tag. You probably don't want to set any value for this parameter; but if you do, the only value it should ever be set to is a newline character: `
` or `
`

Name

draft.mode — Select draft mode

Synopsis

```
<xsl:param name="draft.mode">maybe</xsl:param>
```

Description

Selects draft mode. If *`draft.mode`* is "`yes`", the entire document will be treated as a draft. If it is "`no`", the entire document will be treated as a final copy. If it is "`maybe`", individual sections will be treated as draft or final independently, depending on how their `status` attribute is set.

Name

draft.watermark.image — The URI of the image to be used for draft watermarks

Synopsis

```
<xsl:param \
name="draft.watermark.image">http://docbook.sourceforge.net/release/images/draft.png</xsl:param>
```

Description

The image to be used for draft watermarks.

Name

generate.id.attributes — Generate ID attributes on container elements?

[1] http://www.exslt.org/

Synopsis

```
<xsl:param name="generate.id.attributes" select="0"></xsl:param>
```

Description

If non-zero, the HTML stylesheet will generate ID attributes on containers. For example, the markup:

```
<section id="foo"><title>Some Title</title>
<para>Some para.</para>
</section>
```

might produce:

```
<div class="section" id="foo">
<h2>Some Title</h2>
<p>Some para.</p>
</div>
```

The alternative is to generate anchors:

```
<div class="section">
<h2><a name="foo"></a>Some Title</h2>
<p>Some para.</p>
</div>
```

Because the `name` attribute of the `a` element and the `id` attribute of other tags are both of type “ID”, producing both generates invalid documents.

As of version 1.50, you can use this switch to control which type of identifier is generated. For backwards-compatibility, generating `a` anchors is preferred.

Note: at present, this switch is incompletely implemented. Disabling ID attributes will suppress them, but enabling ID attributes will not suppress the anchors.

Name

generate.meta.abstract — Generate HTML `META` element from `abstract`?

Synopsis

```
<xsl:param name="generate.meta.abstract" select="1"></xsl:param>
```

Description

If non-zero, document abstracts will be reproduced in the HTML `head`, with `>meta name="description" content="..."`

XSLT Processing

Name

rootid — Specify the root element to format

Synopsis

```
<xsl:param name="rootid"></xsl:param>
```

Description

If *`rootid`* is not empty, it must be the value of an ID that occurs in the document being formatted. The entire document will be loaded and parsed, but formatting will begin at the element identified, rather than at the root. For example, this allows you to process only `chapter` 4 of a `book`.

Because the entire document is available to the processor, automatic numbering, cross references, and other dependencies are correctly resolved.

Name

suppress.navigation — Disable header and footer navigation

Synopsis

```
<xsl:param name="suppress.navigation" select="0"></xsl:param>
```

Description

If non-zero, header and footer navigation will be suppressed.

Name

suppress.header.navigation — Disable header navigation

Synopsis

```
<xsl:param name="suppress.header.navigation" select="0"></xsl:param>
```

Description

If non-zero, header navigation will be suppressed.

Name

suppress.footer.navigation — Disable footer navigation

Synopsis

```
<xsl:param name="suppress.footer.navigation">0</xsl:param>
```

Description

If non-zero, footer navigation will be suppressed.

Name

header.rule — Rule under headers?

Synopsis

```
<xsl:param name="header.rule" select="1"></xsl:param>
```

Description

If non-zero, a rule will be drawn below the page headers.

Name

footer.rule — Rule over footers?

Synopsis

```
<xsl:param name="footer.rule" select="1"></xsl:param>
```

Description

If non-zero, a rule will be drawn above the page footers.

Name

id.warnings — Should warnings be generated for titled elements without IDs?

Synopsis

```
<xsl:param name="id.warnings" select="0"></xsl:param>
```

Description

If non-zero, the stylesheet will issue a warning for any element (other than the root element) which has a `title` but does not have an ID.

Meta/*Info and Titlepages

Name

inherit.keywords — Inherit keywords from ancestor elements?

Synopsis

```
<xsl:param name="inherit.keywords" select="1"></xsl:param>
```

Description

If *`inherit.keywords`* is non-zero, the keyword `meta` for each HTML `head` element will include all of the keywords from ancestor elements. Otherwise, only the keywords from the current section will be used.

Name

make.single.year.ranges — Print single-year ranges (e.g., 1998-1999)

Synopsis

```
<xsl:param name="make.single.year.ranges" select="0"></xsl:param>
```

Description

If non-zero, year ranges that span a single year will be printed in range notation (1998-1999) instead of discrete notation (1998, 1999).

Name

make.year.ranges — Collate copyright years into ranges?

Synopsis

```
<xsl:param name="make.year.ranges" select="0"></xsl:param>
```

Description

If non-zero, multiple copyright `year` elements will be collated into ranges. This works only if each year number is put into a separate year element. The copyright element permits multiple year elements. The stylesheet will not successfully parse a complex year element such as `<year>2001,2002,2003</year>` into a range.

Name

author.othername.in.middle — Is `othername` in `author` a middle name?

Synopsis

```
<xsl:param name="author.othername.in.middle" select="1"></xsl:param>
```

Description

If non-zero, the `othername` of an `author` appears between the `firstname` and `surname`. Otherwise, `othername` is suppressed.

Name

blurb.on.titlepage.enabled — Display personblurb and authorblurb on title pages?

Synopsis

```
<xsl:param name="blurb.on.titlepage.enabled" select="0"></xsl:param>
```

Description

If non-zero, output from `authorblurb` and `personblurb` elements is displayed on title pages. If zero (the default), output from those elements is suppressed on title pages (unless you are using a `titlepage` customization that causes them to be included).

Name

contrib.inline.enabled — Display contrib output inline?

Synopsis

```
<xsl:param name="contrib.inline.enabled">1</xsl:param>
```

Description

If non-zero (the default), output of the `contrib` element is displayed as inline content rather than as block content.

Name

editedby.enabled — Display “Edited by” heading above editor name?

Synopsis

```
<xsl:param name="editedby.enabled">1</xsl:param>
```

Description

If non-zero, a localized **Edited by** heading is displayed above editor names in output of the `editor` element.

Name

abstract.notitle.enabled — Suppress display of abstract titles?

Synopsis

```
<xsl:param name="abstract.notitle.enabled" select="0"></xsl:param>
```

Description

If non-zero, in output of the `abstract` element on titlepages, display of the abstract title is suppressed.

Name

othercredit.like.author.enabled — Display othercredit in same style as author?

Synopsis

```
<xsl:param name="othercredit.like.author.enabled">0</xsl:param>
```

Description

If non-zero, output of the `othercredit` element on titlepages is displayed in the same style as `author` and `editor` output. If zero then `othercredit` output is displayed using a style different than that of `author` and `editor`.

Name

generate.legalnotice.link — Write legalnotice to separate chunk and generate link?

Synopsis

```
<xsl:param name="generate.legalnotice.link" select="0"></xsl:param>
```

Description

If the value of *`generate.legalnotice.link`* is non-zero, the stylesheet:

- writes the contents of `legalnotice` to a separate HTML file
- inserts a hyperlink to the `legalnotice` file
- adds (in the HTML `head`) either a single `link` or element or multiple `link` elements (depending on the value of the *`html.head.legalnotice.link.multiple`* parameter), with the value or values derived from the *`html.head.legalnotice.link.types`* parameter

Otherwise, if *`generate.legalnotice.link`* is zero, `legalnotice` contents are rendered on the title page.

Name

generate.revhistory.link — Write revhistory to separate chunk and generate link?

Synopsis

```
<xsl:param name="generate.revhistory.link" select="0"></xsl:param>
```

Description

If non-zero, the contents of `revhistory` are written to a separate HTML file and a link to the file is generated. Otherwise, `revhistory` contents are rendered on the title page.

Name

html.head.legalnotice.link.types — Specifies link types for legalnotice link in html head

Synopsis

```
<xsl:param name="html.head.legalnotice.link.types">copyright</xsl:param>
```

Description

The value of *`html.head.legalnotice.link.types`* is a space-separated list of link types, as described in Section 6.12 of the HTML 4.01 specification[1]. If the value of the *`generate.legalnotice.link`* parameter is non-zero, then the stylesheet generates (in the `head` section of the HTML source) either a single HTML `link` element or, if the value of the *`html.head.legalnotice.link.multiple`* is non-zero, one `link` element for each link type specified. Each `link` has the following attributes:

- a `rel` attribute whose value is derived from the value of *`html.head.legalnotice.link.types`*

[1] http://www.w3.org/TR/html401/types.html#type-links

- an `href` attribute whose value is set to the URL of the file containing the `legalnotice`
- a `title` attribute whose value is set to the title of the corresponding `legalnotice` (or a title programatically determined by the stylesheet)

For example:

```
<link rel="license" href="ln-id2524073.html" title="Legal Notice">
```

About the default value

In an ideal world, the default value of *`html.head.legalnotice.link.types`* would probably be "`license`", since the content of the DocBook `legalnotice` is typically license information, not copyright information. However, the default value is "`copyright`" for pragmatic reasons: because that's among the set of "recognized link types" listed in Section 6.12 of the HTML 4.01 specification[2], and because certain browsers and browser extensions are preconfigured to recognize that value.

Name

html.head.legalnotice.link.multiple — Generate multiple link instances in html head for legalnotice?

Synopsis

```
<xsl:param name="html.head.legalnotice.link.multiple" select="1"></xsl:param>
```

Description

If *`html.head.legalnotice.link.multiple`* is non-zero and the value of *`html.head.legalnotice.link.types`* contains multiple link types, then the stylesheet generates (in the `head` section of the HTML source) one `link` element for each link type specified. For example, if the value of *`html.head.legalnotice.link.types`* is "`copyright license`":

```
<link rel="copyright" href="ln-id2524073.html" title="Legal Notice">
<link rel="license" href="ln-id2524073.html" title="Legal Notice">
```

Otherwise, the stylesheet generates generates a single `link` instance; for example:

```
<link rel="copyright license" href="ln-id2524073.html" title="Legal Notice">
```

[2] http://www.w3.org/TR/html401/types.html#type-links

Reference Pages

Name

funcsynopsis.decoration — Decorate elements of a `funcsynopsis`?

Synopsis

```
<xsl:param name="funcsynopsis.decoration" select="1"></xsl:param>
```

Description

If non-zero, elements of the `funcsynopsis` will be decorated (e.g. rendered as bold or italic text). The decoration is controlled by templates that can be redefined in a customization layer.

Name

funcsynopsis.style — What style of `funcsynopsis` should be generated?

Synopsis

```
<xsl:param name="funcsynopsis.style">kr</xsl:param>
```

Description

If *`funcsynopsis.style`* is `ansi`, ANSI-style function synopses are generated for a `funcsynopsis`, otherwise K&R-style function synopses are generated.

Name

funcsynopsis.tabular.threshold — Width beyond which a tabular presentation will be used

Synopsis

```
<xsl:param name="funcsynopsis.tabular.threshold">40</xsl:param>
```

Description

If *`funcsynopsis.tabular.threshold`* is greater than zero then if a `funcprototype` is wider than the threshold value, it will be presented in a table.

Name

function.parens — Generate parens after a function?

Synopsis

```
<xsl:param name="function.parens" select="0"></xsl:param>
```

Description

If non-zero, the formatting of a `function` element will include generated parentheses.

Name

refentry.generate.name — Output NAME header before 'RefName'(s)?

Synopsis

```
<xsl:param name="refentry.generate.name" select="1"></xsl:param>
```

Description

If non-zero, a "NAME" section title is output before the list of 'RefName's. This parameter and *`refentry.generate.title`* are mutually exclusive. This means that if you change this parameter to zero, you should set *`refentry.generate.title`* to non-zero unless you want get quite strange output.

Name

refentry.generate.title — Output title before 'RefName'(s)?

Synopsis

```
<xsl:param name="refentry.generate.title" select="0"></xsl:param>
```

Description

If non-zero, the reference page title or first name is output before the list of 'RefName's. This parameter and *`refentry.generate.name`* are mutually exclusive. This means that if you change this parameter to non-zero, you should set *`refentry.generate.name`* to zero unless you want get quite strange output.

Name

refentry.xref.manvolnum — Output `manvolnum` as part of `refentry` cross-reference?

Synopsis

```
<xsl:param name="refentry.xref.manvolnum" select="1"></xsl:param>
```

Description

if non-zero, the `manvolnum` is used when cross-referencing `refentry`s, either with `xref` or `citerefentry`.

Name

citerefentry.link — Generate URL links when cross-referencing RefEntrys?

Synopsis

```
<xsl:param name="citerefentry.link" select="0"></xsl:param>
```

Description

If non-zero, a web link will be generated, presumably to an online man->HTML gateway. The text of the link is generated by the `generate.citerefentry.link` template.

Name

refentry.separator — Generate a separator between consecutive RefEntry elements?

Synopsis

```
<xsl:param name="refentry.separator" select="1"></xsl:param>
```

Description

If true, a separator will be generated between consecutive reference pages.

Name

refclass.suppress — Suppress display of refclass contents?

Synopsis

```
<xsl:param name="refclass.suppress" select="0"></xsl:param>
```

Description

If the value of *`refclass.suppress`* is non-zero, then display of `refclass` contents is suppressed in output.

Tables

Name

default.table.width — The default width of tables

Synopsis

```
<xsl:param name="default.table.width"></xsl:param>
```

Description

If non-zero, this value will be used for the `width` attribute on `table`s that do not specify an alternate width (with the dbhtml processing instruction).

Name

nominal.table.width — The (absolute) nominal width of tables

Synopsis

```
<xsl:param name="nominal.table.width">6in</xsl:param>
```

Description

In order to convert CALS column widths into HTML column widths, it is sometimes necessary to have an absolute table width to use for conversion of mixed absolute and relative widths. This value must be an absolute length (not a percentag).

Name

table.borders.with.css — Use CSS to specify table, row, and cell borders?

Synopsis

```
<xsl:param name="table.borders.with.css" select="0"></xsl:param>
```

Description

If non-zero, CSS will be used to draw table borders.

Name

table.cell.border.style

Synopsis

```
<xsl:param name="table.cell.border.style">solid</xsl:param>
```

Description

FIXME:

Name

table.cell.border.thickness

Synopsis

```
<xsl:param name="table.cell.border.thickness">0.5pt</xsl:param>
```

Description

If non-zero, specifies the thickness of borders on table cells. The units are points. See CSS[1]

Name

table.cell.border.color

Synopsis

```
<xsl:param name="table.cell.border.color"></xsl:param>
```

Description

Set the color of table borders. If non-zero, the value is used for the border coloration. See CSS[1]. A `color` is either a keyword or a numerical RGB specification. Keywords are aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow.

Name

table.frame.border.style

Synopsis

```
<xsl:param name="table.frame.border.style">solid</xsl:param>
```

Description

FIXME:

Name

table.frame.border.thickness — Specifies the thickness of the frame border

Synopsis

```
<xsl:param name="table.frame.border.thickness">0.5pt</xsl:param>
```

Description

Specifies the thickness of the border on the table's frame.

Name

table.frame.border.color

Synopsis

```
<xsl:param name="table.frame.border.color"></xsl:param>
```

[1] http://www.w3.org/TR/CSS21/box.html#border-width-properties
[1] http://www.w3.org/TR/CSS21/syndata.html#value-def-color

Description

FIXME:

Name

default.table.frame — The default framing of tables

Synopsis

```
<xsl:param name="default.table.frame">all</xsl:param>
```

Description

This value will be used when there is no frame attribute on the table.

Name

html.cellspacing — Default value for cellspacing in HTML tables

Synopsis

```
<xsl:param name="html.cellspacing"></xsl:param>
```

Description

If non-zero, this value will be used as the default cellspacing value in HTML tables. nn for pixels or nn% for percentage length. E.g. 5 or 5%

Name

html.cellpadding — Default value for cellpadding in HTML tables

Synopsis

```
<xsl:param name="html.cellpadding"></xsl:param>
```

Description

If non-zero, this value will be used as the default cellpadding value in HTML tables. nn for pixels or nn% for percentage length. E.g. 5 or 5%

QAndASet

Name

qanda.defaultlabel — Sets the default for defaultlabel on QandASet.

Synopsis

```
<xsl:param name="qanda.defaultlabel">number</xsl:param>
```

Description

If no `defaultlabel` attribute is specified on a `qandaset`, this value is used. It must be one of the legal values for the defaultlabel attribute, one from none, number or qanda. The default value is 'number'.

Meaning

qanda - `question`s are labeled “Q:” and `answer`s are labeled “A:”.

number - The entries are enumerated.

none - No distinguishing label precedes Questions or Answers.

Name

qanda.inherit.numeration — Does enumeration of QandASet components inherit the numeration of parent elements?

Synopsis

```
<xsl:param name="qanda.inherit.numeration" select="1"></xsl:param>
```

Description

If non-zero, numbered `qandadiv` elements and `question` and `answer` inherit the enumeration of the ancestors of the `qandaset`.

Name

qanda.in.toc — Should qandaentry questions appear in the document table of contents?

Synopsis

```
<xsl:param name="qanda.in.toc" select="0"></xsl:param>
```

Description

If true (non-zero), then the generated table of contents for a document will include `qandaset` titles, `qandadiv` titles, and `question` elements. The default value (zero) excludes them from the TOC.

This parameter does not affect any tables of contents that may be generated inside a qandaset or qandadiv.

Name

qanda.nested.in.toc — Should nested answer/qandaentry instances appear in TOC?

Synopsis

```
<xsl:param name="qanda.nested.in.toc" select="0"></xsl:param>
```

Description

If non-zero, instances of `qandaentry` that are children of `answer` elements are shown in the TOC.

Linking

Name

target.database.document — Name of master database file for resolving olinks

Synopsis

```
<xsl:param name="target.database.document">olinkdb.xml</xsl:param>
```

Description

To resolve olinks between documents, the stylesheets use a master database document that identifies the target datafiles for all the documents within the scope of the olinks. This parameter value is the URI of the master document to be read during processing to resolve olinks. The default value is `olinkdb.xml`.

The data structure of the file is defined in the `targetdatabase.dtd` DTD. The database file provides the high level elements to record the identifiers, locations, and relationships of documents. The cross reference data for individual documents is generally pulled into the database using system entity references or XIncludes. See also *`targets.filename`*.

Name

targets.filename — Name of cross reference targets data file

Synopsis

```
<xsl:param name="targets.filename">target.db</xsl:param>
```

Description

In order to resolve olinks efficiently, the stylesheets can generate an external data file containing information about all potential cross reference endpoints in a document. This parameter lets you change the name of the generated file from the default name `target.db`. The name must agree with that used in the target database used to resolve olinks during processing. See also *`target.database.document`*.

Name

olink.base.uri — Base URI used in olink hrefs

Synopsis

```
<xsl:param name="olink.base.uri"></xsl:param>
```

Description

When cross reference data is collected for resolving olinks, it may be necessary to prepend a base URI to each target's href. This parameter lets you set that base URI when cross reference data is collected. This feature is needed when you want to link to a document that is processed without chunking. The output filename for such a document is not known to the XSL stylesheet; the only target information consists of fragment identifiers such as `#idref`. To enable the resolution of olinks between documents, you should pass the name of the HTML output file as the value of this parameter. Then the hrefs recorded in the cross reference data collection look like `outfile.html#idref`, which can be reached as links from other documents.

Name

use.local.olink.style — Process olinks using xref style of current document

Synopsis

```
<xsl:param name="use.local.olink.style" select="0"></xsl:param> \
```

Description

When cross reference data is collected for use by olinks, the data for each potential target includes one field containing a completely assembled cross reference string, as if it were an xref generated in that document. Other fields record the separate title, number, and element name of each target. When an olink is formed to a target from another document, the olink resolves to that preassembled string by default. If the *`use.local.olink.style`* parameter is set to non-zero, then instead the cross reference string is formed again from the target title, number, and element name, using the stylesheet processing the targeting document. Then olinks will match the xref style in the targeting document rather than in the target document. If both documents are processed with the same stylesheet, then the results will be the same.

Name

current.docid — targetdoc identifier for the document being processed

Synopsis

```
<xsl:param name="current.docid"></xsl:param>
```

Description

When olinks between documents are resolved for HTML output, the stylesheet can compute the relative path between the current document and the target document. The stylesheet needs to know the `targetdoc` identifiers for both documents, as they appear in the *`target.database.document`* database file. This parameter passes to the stylesheet the targetdoc identifier of the current document, since that identifier does not appear in the document itself.

This parameter can also be used for print output. If an olink's `targetdoc` id differs from the `current.docid`, then the stylesheet can append the target document's title to the generated olink text. That identifies to the reader that the link is to a different document, not the current document. See also *`olink.doctitle`* to enable that feature.

Name

olink.doctitle — show the document title for external olinks?

Synopsis

```
<xsl:param name="olink.doctitle">no</xsl:param>
```

Description

When olinks between documents are resolved, the generated text may not make it clear that the reference is to another document. It is possible for the stylesheets to append the other document's title to external olinks. For this to happen, two parameters must be set.

- This *`olink.doctitle`* parameter should be set to either `yes` or `maybe` to enable this feature.
- And you should also set the *`current.docid`* parameter to the document id for the document currently being processed for output.

Then if an olink's `targetdoc` id differs from the `current.docid` value, the stylesheet knows that it is a reference to another document and can append the target document's title to the generated olink text.

The text for the target document's title is copied from the olink database from the `ttl` element of the top-level `div` for that document. If that `ttl` element is missing or empty, no title is output.

The supported values for *`olink.doctitle`* are:

`yes`
: Always insert the title to the target document if it is not the current document.

`no`
: Never insert the title to the target document, even if requested in an `xrefstyle` attribute.

`maybe`
: Only insert the title to the target document, if requested in an `xrefstyle` attribute.

An `xrefstyle` attribute may override the global setting for individual olinks. The following values are supported in an `xrefstyle` attribute using the `select:` syntax:

`docname`
: Insert the target document name for this olink using the `docname` gentext template, but only if the value of *`olink.doctitle`* is not `no`.

`docnamelong`
: Insert the target document name for this olink using the `docnamelong` gentext template, but only if the value of *`olink.doctitle`* is not `no`.

`nodocname`
: Omit the target document name even if the value of *`olink.doctitle`* is `yes`.

Another way of inserting the target document name for a single olink is to employ an `xrefstyle` attribute using the `template:` syntax. The `%o` placeholder (the letter o, not zero) in such a template will be filled in with the target document's title when it is processed. This will occur regardless of the value of *`olink.doctitle`*.

Note that prior to version 1.66 of the XSL stylesheets, the allowed values for this parameter were 0 and 1. Those values are still supported and mapped to 'no' and 'yes', respectively.

Name

olink.debug — Turn on debugging messages for olinks

Synopsis

```
<xsl:param name="olink.debug" select="0"></xsl:param>
```

Description

If non-zero, then each olink will generate several messages about how it is being resolved during processing. This is useful when an olink does not resolve properly and the standard error messages are not sufficient to find the problem.

You may need to read through the olink XSL templates to understand the context for some of the debug messages.

Name

olink.properties — Properties associated with the cross-reference text of an olink.

Synopsis

```
<xsl:attribute-set name="olink.properties">
  <xsl:attribute name="show-destination">replace</xsl:attribute>
</xsl:attribute-set>
```

Description

This `attribute set` is applied to the `fo:basic-link` element of an olink. It is not applied to the optional page number or optional title of the external document.

Name

olink.lang.fallback.sequence — look up translated documents if olink not found?

Synopsis

```
<xsl:param name="olink.lang.fallback.sequence"></xsl:param>
```

Description

This parameter defines a list of lang values to search among to resolve olinks.

Normally an olink tries to resolve to a document in the same language as the olink itself. The language of an olink is determined by its nearest ancestor element with a `lang` attribute, otherwise the value of the *`l10n.gentext.default.lang`* parameter.

An olink database can contain target data for the same document in multiple languages. Each set of data has the same value for the `targetdoc` attribute in the `document` element in the database, but with a different `lang` attribute value.

When an olink is being resolved, the target is first sought in the document with the same language as the olink. If no match is found there, then this parameter is consulted for additional languages to try.

The *`olink.lang.fallback.sequence`* must be a whitespace separated list of lang values to try. The first one with a match in the olink database is used. The default value is empty.

For example, a document might be written in German and contain an olink with `targetdoc="adminguide"`. When the document is processed, the processor first looks for a target dataset in the olink database starting with:

```
<document targetdoc="adminguide" lang="de">.
```

If there is no such element, then the *`olink.lang.fallback.sequence`* parameter is consulted. If its value is, for example, “fr en”, then the processor next looks for `targetdoc="adminguide" lang="fr"`, and then for `targetdoc="adminguide" lang="en"`. If there is still no match, it looks for `targetdoc="adminguide"` with no lang attribute.

This parameter is useful when a set of documents is only partially translated, or is in the process of being translated. If a target of an olink has not yet been translated, then this parameter permits the processor to look for the document in other languages. This assumes the reader would rather have a link to a document in a different language than to have a broken link.

Name

insert.olink.page.number — Turns page numbers in olinks on and off

Synopsis

```
<xsl:param name="insert.olink.page.number">no</xsl:param>
```

Description

The value of this parameter determines if cross references made between documents with `olink` will include page number citations. In most cases this is only applicable to references in printed output.

The parameter has three possible values.

no
: No page number references will be generated for olinks.

yes
: Page number references will be generated for all `olink` references. The style of page reference may be changed if an `xrefstyle` attribute is used.

maybe
: Page number references will not be generated for an `olink` element unless it has an `xrefstyle` attribute whose value specifies a page reference.

Olinks that point to targets within the same document are treated as `xref`s, and controlled by the *`insert.xref.page.number`* parameter.

Page number references for olinks to external documents can only be inserted if the information exists in the olink database. This means each olink target element (`div` or `obj`) must have a `page` attribute whose value is its page number in the target document. The XSL stylesheets are not able to extract that information during processing because pages have not yet been created in XSLT transformation. Only the XSL-FO processor knows what page each element is placed on. Therefore some postprocessing must take place to populate page numbers in the olink database.

Name

insert.olink.pdf.frag — Add fragment identifiers for links into PDF files

Synopsis

```
<xsl:param name="insert.olink.pdf.frag" select="0"></xsl:param>
```

Description

The value of this parameter determines whether the cross reference URIs to PDF documents made with `olink` will include fragment identifiers.

When forming a URI to link to a PDF document, a fragment identifier (typically a '#' followed by an id value) appended to the PDF filename can be used by the PDF viewer to open the PDF file to a location within the document instead of the first page. However, not all PDF files have id values embedded in them, and not all PDF viewers can handle fragment identifiers.

If *`insert.olink.pdf.frag`* is set to a non-zero value, then any olink targeting a PDF file will have the fragment identifier appended to the URI. The URI is formed by concatenating the value of the *`olink.base.uri`* parameter, the value of the `baseuri` attribute from the `document` element in the olink database with the matching `targetdoc` value, and the value of the `href` attribute for the targeted element in the olink database. The `href` attribute contains the fragment identifier.

If *`insert.olink.pdf.frag`* is set to zero (the default value), then the `href` attribute from the olink database is not appended to PDF olinks, so the fragment identifier is left off. A PDF olink is any

olink for which the `baseuri` attribute from the matching `document` element in the olink database ends with '.pdf'. Any other olinks will still have the fragment identifier added.

Name

prefer.internal.olink — Prefer a local olink reference to an external reference

Synopsis

```
<xsl:param name="prefer.internal.olink" select="0"></xsl:param>
```

Description

If you are re-using XML content modules in multiple documents, you may want to redirect some of your olinks. This parameter permits you to redirect an olink to the current document.

For example: you are writing documentation for a product, which includes 3 manuals: a little installation booklet (booklet.xml), a user guide (user.xml), and a reference manual (reference.xml). All 3 documents begin with the same introduction section (intro.xml) that contains a reference to the customization section (custom.xml) which is included in both user.xml and reference.xml documents.

How do you write the link to custom.xml in intro.xml so that it is interpreted correctly in all 3 documents?

- If you use xref, it will fail in user.xml.
- If you use olink (pointing to reference.xml), the reference in user.xml will point to the customization section of the reference manual, while it is actually available in user.xml.

If you set the *`prefer.internal.olink`* parameter to a non-zero value, then the processor will first look in the olink database for the olink's `targetptr` attribute value in document matching the *`current.docid`* parameter value. If it isn't found there, then it tries the document in the database with the `targetdoc` value that matches the olink's `targetdoc` attribute.

This feature permits an olink reference to resolve to the current document if there is an element with an id matching the olink's `targetptr` value. The current document's olink data must be included in the target database for this to work.

Caution

There is a potential for incorrect links if the same `id` attribute value is used for different content in different documents. Some of your olinks may be redirected to the current document when they shouldn't be. It is not possible to control individual olink instances.

Name

link.mailto.url — Mailto URL for the LINK REL=made HTML HEAD element

Synopsis

```
<xsl:param name="link.mailto.url"></xsl:param>
```

Description

If not the empty string, this address will be used for the rel=made `link` element in the html `head`

Name

ulink.target — The HTML anchor target for ULinks

Synopsis

```
<xsl:param name="ulink.target">_top</xsl:param>
```

Description

If *`ulink.target`* is non-zero, its value will be used for the `target` attribute on anchors generated for `ulink`s.

Name

olink.fragid — Names the fragment identifier portion of an OLink resolver query

Synopsis

```
<xsl:param name="olink.fragid">fragid=</xsl:param>
```

Description

The fragment identifier portion of an `olink` target.

Name

olink.outline.ext — The extension of OLink outline files

Synopsis

```
<xsl:param name="olink.outline.ext">.olink</xsl:param>
```

Description

The extension to be expected for OLink outline files

Bob has this parameter as dead. Please don't use

Name

olink.pubid — Names the public identifier portion of an OLink resolver query

Synopsis

```
<xsl:param name="olink.pubid">pubid</xsl:param>
```

Description

Name

olink.sysid — Names the system identifier portion of an OLink resolver query

Synopsis

```
<xsl:param name="olink.sysid">sysid</xsl:param>
```

Description

FIXME

Name

olink.resolver — The root name of the OLink resolver (usually a script)

Synopsis

```
<xsl:param name="olink.resolver">/cgi-bin/olink</xsl:param>
```

Description

FIXME:

Cross References

Name

collect.xref.targets — Controls whether cross reference data is collected

Synopsis

```
<xsl:param name="collect.xref.targets">no</xsl:param>
```

Description

In order to resolve olinks efficiently, the stylesheets can generate an external data file containing information about all potential cross reference endpoints in a document. This parameter determines whether the collection process is run when the document is processed by the stylesheet. The default value is `no`, which means the data file is not generated during processing. The other choices are `yes`, which means the data file is created and the document is processed for output, and `only`, which means the data file is created but the document is not processed for output. See also *`targets.filename`*.

Name

insert.xref.page.number — Turns page numbers in xrefs on and off

Synopsis

```
<xsl:param name="insert.xref.page.number">no</xsl:param>
```

Description

The value of this parameter determines if cross references (`xref`s) in printed output will include page number citations. It has three possible values.

no
: No page number references will be generated.

yes
: Page number references will be generated for all `xref` elements. The style of page reference may be changed if an `xrefstyle` attribute is used.

maybe
: Page number references will not be generated for an `xref` element unless it has an `xrefstyle` attribute whose value specifies a page reference.

Name

use.role.as.xrefstyle — Use `role` attribute for `xrefstyle` on `xref`?

Synopsis

```
<xsl:param name="use.role.as.xrefstyle" select="1"></xsl:param>
```

Description

If non-zero, the `role` attribute on `xref` will be used to select the cross reference style. The DocBook Technical Committee[1] recently added an `xrefstyle` attribute for this purpose. If the `xrefstyle` attribute is present, `role` will be ignored, regardless of this setting.

[1] http://www.oasis-open.org/docbook/

Until an official DocBook release that includes the new attribute, this flag allows `role` to serve that purpose.

Example

The following small stylesheet shows how to configure the stylesheets to make use of the cross reference style:

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

<xsl:import href="../xsl/html/docbook.xsl"/>

<xsl:output method="html"/>

<xsl:param name="local.l10n.xml" select="document('')"/>
<l:i18n xmlns:l="http://docbook.sourceforge.net/xmlns/l10n/1.0">
  <l:l10n xmlns:l="http://docbook.sourceforge.net/xmlns/l10n/1.0" language="en">
   <l:context name="xref">
      <l:template name="chapter" style="title" text="Chapter %n, %t"/>
      <l:template name="chapter" text="Chapter %n"/>
    </l:context>
  </l:l10n>
</l:i18n>

</xsl:stylesheet>
```

With this stylesheet, the cross references in the following document:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
                  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book id="book"><title>Book</title>

<preface>
<title>Preface</title>

<para>Normal: <xref linkend="ch1"/>.</para>
<para>Title: <xref xrefstyle="title" linkend="ch1"/>.</para>

</preface>

<chapter id="ch1">
<title>First Chapter</title>

<para>Irrelevant.</para>

</chapter>
</book>
```

will appear as:

Normal: Chapter 1.

Title: Chapter 1, *First Chapter*.

Name

xref.with.number.and.title — Use number and title in cross references

Synopsis

```
<xsl:param name="xref.with.number.and.title" select="1"></xsl:param>
```

Description

A cross reference may include the number (for example, the number of an example or figure) and the `title` which is a required child of some targets. This parameter inserts both the relevant number as well as the title into the link.

Name

xref.label-page.separator — Punctuation or space separating label from page number in xref

Synopsis

```
<xsl:param name="xref.label-page.separator"><xsl:text> </xsl:text></xsl:param>
```

Description

This parameter allows you to control the punctuation of certain types of generated cross reference text. When cross reference text is generated for an `xref` or `olink` element using an `xrefstyle` attribute that makes use of the `select:` feature, and the selected components include both label and page but no title, then the value of this parameter is inserted between label and page number in the output. If a title is included, then other separators are used.

Name

xref.label-title.separator — Punctuation or space separating label from title in xref

Synopsis

```
<xsl:param name="xref.label-title.separator">: </xsl:param>
```

Description

This parameter allows you to control the punctuation of certain types of generated cross reference text. When cross reference text is generated for an `xref` or `olink` element using an `xrefstyle` attribute that makes use of the `select:` feature, and the selected components include both label and title, then the value of this parameter is inserted between label and title in the output.

Name

xref.title-page.separator — Punctuation or space separating title from page number in xref

Synopsis

```
<xsl:param name="xref.title-page.separator"><xsl:text> </xsl:text></xsl:param>
```

Description

This parameter allows you to control the punctuation of certain types of generated cross reference text. When cross reference text is generated for an `xref` or `olink` element using an `xrefstyle` attribute that makes use of the `select:` feature, and the selected components include both title and page number, then the value of this parameter is inserted between title and page number in the output.

Lists

Name

segmentedlist.as.table — Format segmented lists as tables?

Synopsis

```
<xsl:param name="segmentedlist.as.table" select="0"></xsl:param>
```

Description

If non-zero, `segmentedlist`s will be formatted as tables.

Name

variablelist.as.table — Format `variablelist`s as tables?

Synopsis

```
<xsl:param name="variablelist.as.table" select="0"></xsl:param>
```

Description

If non-zero, `variablelist`s will be formatted as tables. A processing instruction exists to specify a particualar width for the column containing the `term`s: `<?dbhtml term-width=".25in"?>`

You can override this setting with a processing instruction as the child of `variablelist`: `<?dbhtml list-presentation="table"?>` or `<?dbhtml list-presentation="list"?>`.

This parameter only applys to the HTML transformations. In the FO case, proper list markup is robust enough to handle the formatting. But see also *`variablelist.as.blocks`*.

```
  <variablelist>
        <?dbhtml list-presentation="table"?>
        <?dbhtml term-width="1.5in"?>
        <?dbfo list-presentation="list"?>
        <?dbfo term-width="1in"?>
        <varlistentry>
          <term>list</term>
          <listitem>
                <para>
                  Formatted as a table even if variablelist.as.table is set to 0.
                </para>
          </listitem>
        </varlistentry>
  </variablelist>
```

Name

variablelist.term.separator — Text to separate `term`s within a multi-term `varlistentry`

Synopsis

```
<xsl:param name="variablelist.term.separator">, </xsl:param>
```

Description

When a `varlistentry` contains multiple `term` elements, the string specified in the value of the *`variablelist.term.separator`* parameter is placed after each `term` except the last.

Note

To generate a line break between multiple `term`s in a `varlistentry`, set a non-zero value for the *`variablelist.term.break.after`* parameter. If you do so, you may also want to set the value of the *`variablelist.term.separator`* parameter to an empty string (to suppress rendering of the default comma and space after each `term`).

Name

variablelist.term.break.after — Generate line break after each `term` within a multi-term `varlistentry`?

Synopsis

```
<xsl:param name="variablelist.term.break.after">0</xsl:param>
```

Description

Set a non-zero value for the *`variablelist.term.break.after`* parameter to generate a line break between `term`s in a multi-term `varlistentry`.

Note

If you set a non-zero value for *`variablelist.term.break.after`*, you may also want to set the value of the *`variablelist.term.separator`* parameter to an empty string (to suppress rendering of the default comma and space after each `term`).

Bibliography

Name

bibliography.style — Style used for formatting of biblioentries.

Synopsis

```
<xsl:param name="bibliography.style">normal</xsl:param>
```

Description

Currently only `normal` and `iso690` styles are supported.

In order to use ISO690 style to the full extent you might need to use additional markup described on the following WiKi page[1].

Name

biblioentry.item.separator — Text to separate bibliography entries

Synopsis

```
<xsl:param name="biblioentry.item.separator">. </xsl:param>
```

Description

Text to separate bibliography entries

Name

bibliography.collection — Name of the bibliography collection file

Synopsis

```
<xsl:param \
name="bibliography.collection">http://docbook.sourceforge.net/release/bibliography/bibliography.xml</xsl:param>
```

Description

Maintaining bibliography entries across a set of documents is tedious, time consuming, and error prone. It makes much more sense, usually, to store all of the bibliography entries in a single place and simply “extract” the ones you need in each document.

That's the purpose of the *`bibliography.collection`* parameter. To setup a global bibliography “database”, follow these steps:

First, create a stand-alone bibliography document that contains all of the documents that you wish to reference. Make sure that each bibliography entry (whether you use `biblioentry` or `bibliomixed`) has an ID.

My global bibliography, `~/bibliography.xml` begins like this:

```
<!DOCTYPE bibliography
  PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
```

[1] http://wiki.docbook.org/topic/ISO690Bibliography

```
  "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
<bibliography><title>References</title>

<bibliomixed id="xml-rec"><abbrev>XML 1.0</abbrev>Tim Bray,
Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler, editors.
<citetitle><ulink url="http://www.w3.org/TR/REC-xml">Extensible Markup
Language (XML) 1.0 Second Edition</ulink></citetitle>.
World Wide Web Consortium, 2000.
</bibliomixed>

<bibliomixed id="xml-names"><abbrev>Namespaces</abbrev>Tim Bray,
Dave Hollander,
and Andrew Layman, editors.
<citetitle><ulink url="http://www.w3.org/TR/REC-xml-names/">Namespaces in
XML</ulink></citetitle>.
World Wide Web Consortium, 1999.
</bibliomixed>

<!-- ... -->
</bibliography>
```

When you create a bibliography in your document, simply provide *empty* `bibliomixed` entries for each document that you wish to cite. Make sure that these elements have the same ID as the corresponding “real” entry in your global bibliography.

For example:

```
<bibliography><title>Bibliography</title>

<bibliomixed id="xml-rec"/>
<bibliomixed id="xml-names"/>
<bibliomixed id="DKnuth86">Donald E. Knuth. <citetitle>Computers and
Typesetting: Volume B, TeX: The Program</citetitle>. Addison-Wesley,
1986.  ISBN 0-201-13437-3.
</bibliomixed>
<bibliomixed id="relaxng"/>

</bibliography>
```

Note that it's perfectly acceptable to mix entries from your global bibliography with “normal” entries. You can use `xref` or other elements to cross-reference your bibliography entries in exactly the same way you do now.

Finally, when you are ready to format your document, simply set the *`bibliography.collection`* parameter (in either a customization layer or directly through your processor's interface) to point to your global bibliography.

The stylesheets will format the bibliography in your document as if all of the entries referenced appeared there literally.

Name

bibliography.numbered — Should bibliography entries be numbered?

Synopsis

```
<xsl:param name="bibliography.numbered" select="0"></xsl:param>
```

Description

If non-zero bibliography entries will be numbered

Glossary

Name

glossterm.auto.link — Generate links from glossterm to glossentry automaticaly?

Synopsis

```
<xsl:param name="glossterm.auto.link" select="0"></xsl:param>
```

Description

If true, a link will be automatically created from glossterm to glossentry for that glossary term. This is usefull when your glossterm names are consistent and you don't want to add links manually.

If there is `linkend` on `glossterm` then is used instead of autogeneration of link.

Name

firstterm.only.link — Does automatic glossterm linking only apply to firstterms?

Synopsis

```
<xsl:param name="firstterm.only.link" select="0"></xsl:param>
```

Description

If non-zero, only `firstterm`s will be automatically linked to the glossary. If glossary linking is not enabled, this parameter has no effect.

Name

glossary.collection — Name of the glossary collection file

Synopsis

```
<xsl:param name="glossary.collection"></xsl:param>
```

Description

Glossaries maintained independently across a set of documents are likely to become inconsistent unless considerable effort is expended to keep them in sync. It makes much more sense, usually, to store all of the glossary entries in a single place and simply “extract” the ones you need in each document.

That's the purpose of the *`glossary.collection`* parameter. To setup a global glossary “database”, follow these steps:

Setting Up the Glossary Database

First, create a stand-alone glossary document that contains all of the entries that you wish to reference. Make sure that each glossary entry has an ID.

Here's an example glossary:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE glossary
  PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
```

```
<glossary>
<glossaryinfo>
<editor><firstname>Eric</firstname><surname>Raymond</surname></editor>
<title>Jargon File 4.2.3 (abridged)</title>
<releaseinfo>Just some test data</releaseinfo>
</glossaryinfo>

<glossdiv><title>0</title>

<glossentry>
<glossterm>0</glossterm>
<glossdef>
<para>Numeric zero, as opposed to the letter `O' (the 15th letter of
the English alphabet). In their unmodified forms they look a lot
alike, and various kluges invented to make them visually distinct have
compounded the confusion. If your zero is center-dotted and letter-O
is not, or if letter-O looks almost rectangular but zero looks more
like an American football stood on end (or the reverse), you're
probably looking at a modern character display (though the dotted zero
seems to have originated as an option on IBM 3270 controllers). If
your zero is slashed but letter-O is not, you're probably looking at
an old-style ASCII graphic set descended from the default typewheel on
the venerable ASR-33 Teletype (Scandinavians, for whom /O is a letter,
curse this arrangement). (Interestingly, the slashed zero long
predates computers; Florian Cajori's monumental "A History of
Mathematical Notations" notes that it was used in the twelfth and
thirteenth centuries.) If letter-O has a slash across it and the zero
does not, your display is tuned for a very old convention used at IBM
and a few other early mainframe makers (Scandinavians curse <emphasis>this</emphasis>
arrangement even more, because it means two of their letters collide).
Some Burroughs/Unisys equipment displays a zero with a <emphasis>reversed</emphasis>
slash. Old CDC computers rendered letter O as an unbroken oval and 0
as an oval broken at upper right and lower left. And yet another
convention common on early line printers left zero unornamented but
added a tail or hook to the letter-O so that it resembled an inverted
Q or cursive capital letter-O (this was endorsed by a draft ANSI
standard for how to draw ASCII characters, but the final standard
changed the distinguisher to a tick-mark in the upper-left corner).
Are we sufficiently confused yet?</para>
</glossdef>
</glossentry>

<glossentry>
<glossterm>1TBS</glossterm>
<glossdef>
<para role="accidence">
<phrase role="pronounce"></phrase>
<phrase role="partsofspeach">n</phrase>
</para>
<para>The "One True Brace Style"</para>
<glossseealso>indent style</glossseealso>
</glossdef>
</glossentry>

<!-- ... -->

</glossdiv>

<!-- ... -->

</glossary>
```

Marking Up Glossary Terms

That takes care of the glossary database, now you have to get the entries into your document. Unlike bibliography entries, which can be empty, creating “placeholder” glossary entries would be very tedious. So instead, support for *`glossary.collection`* relies on implicit linking.

In your source document, simply use `firstterm` and `glossterm` to identify the terms you wish to have included in the glossary. The stylesheets assume that you will either set the `baseform` attribute correctly, or that the content of the element exactly matches a term in your glossary.

If you're using a *`glossary.collection`*, don't make explicit links on the terms in your document.

So, in your document, you might write things like this:

```
<para>This is dummy text, without any real meaning.
The point is simply to reference glossary terms like <glossterm>0</glossterm>
and the <firstterm baseform="1TBS">One True Brace Style (1TBS)</firstterm>.
The <glossterm>1TBS</glossterm>, as you can probably imagine, is a nearly
religious issue.</para>
```

If you set the *`firstterm.only.link`* parameter, only the terms marked with `firstterm` will be links. Otherwise, all the terms will be linked.

Marking Up the Glossary

The glossary itself has to be identified for the stylesheets. For lack of a better choice, the `role` is used. To identify the glossary as the target for automatic processing, set the role to "`auto`". The title of this glossary (and any other information from the `glossaryinfo` that's rendered by your stylesheet) will be displayed, but the entries will come from the database.

Unfortunately, the glossary can't be empty, so you must put in at least one `glossentry`. The content of this entry is irrelevant, it will not be rendered:

```
<glossary role="auto">
<glossentry>
<glossterm>Irrelevant</glossterm>
<glossdef>
<para>If you can see this, the document was processed incorrectly. Use
the <parameter>glossary.collection</parameter> parameter.</para>
</glossdef>
</glossentry>
</glossary>
```

What about glossary divisions? If your glossary database has glossary divisions *and* your automatic glossary contains at least one `glossdiv`, the automic glossary will have divisions. If the `glossdiv` is missing from either location, no divisions will be rendered.

Glossary entries (and divisions, if appropriate) in the glossary will occur in precisely the order they occur in your database.

Formatting the Document

Finally, when you are ready to format your document, simply set the *`glossary.collection`* parameter (in either a customization layer or directly through your processor's interface) to point to your global glossary.

The stylesheets will format the glossary in your document as if all of the entries implicilty referenced appeared there literally.

Limitations

Glossary cross-references *within the glossary* are not supported. For example, this *will not* work:

```
<glossentry>
<glossterm>gloss-1</glossterm>
<glossdef><para>A description that references <glossterm>gloss-2</glossterm>.</para>
<glossseealso>gloss-2</glossseealso>
</glossdef>
</glossentry>
```

If you put glossary cross-references in your glossary that way, you'll get the cryptic error: `Warning: glossary.collection specified, but there are 0 automatic glossaries`.

Instead, you must do two things:

1. Markup your glossary using `glossseealso`:

```
<glossentry>
<glossterm>gloss-1</glossterm>
<glossdef><para>A description that references <glossterm>gloss-2</glossterm>.</para>
<glossseealso>gloss-2</glossseealso>
</glossdef>
</glossentry>
```

2. Make sure there is at least one `glossterm` reference to *gloss-2 in your document*. The easiest way to do that is probably within a `remark` in your automatic glossary:

```
<glossary role="auto">
<remark>Make sure there's a reference to <glossterm>gloss-2</glossterm>.</remark>
<glossentry>
<glossterm>Irrelevant</glossterm>
<glossdef>
<para>If you can see this, the document was processed incorrectly. Use
the <parameter>glossary.collection</parameter> parameter.</para>
</glossdef>
</glossentry>
</glossary>
```

Name

glossary.sort — Sort glossentry elements?

Synopsis

```
<xsl:param name="glossary.sort" select="0"></xsl:param>
```

Description

If non-zero, then the glossentry elements within a glossary, glossdiv, or glosslist are sorted on the glossterm, using the current lang setting. If zero (the default), then glossentry elements are not sorted and are presented in document order.

Name

glossentry.show.acronym — Display `glossentry` acronyms?

Synopsis

```
<xsl:param name="glossentry.show.acronym">no</xsl:param>
```

Description

A setting of “yes” means they should be displayed; “no” means they shouldn't. If “primary” is used, then they are shown as the primary text for the entry.

Note

This setting controls both `acronym` and `abbrev` elements in the `glossentry`.

Miscellaneous

Name

formal.procedures — Selects formal or informal procedures

Synopsis

```
<xsl:param name="formal.procedures" select="1"></xsl:param>
```

Description

Formal procedures are numbered and always have a title.

Name

formal.title.placement — Specifies where formal object titles should occur

Synopsis

```
<xsl:param name="formal.title.placement">
figure before
example before
equation before
table before
procedure before
task before
</xsl:param>
```

Description

Specifies where formal object titles should occur. For each formal object type (`figure`, `example`, `equation`, `table`, and `procedure`) you can specify either the keyword "`before`" or "`after`".

Name

runinhead.default.title.end.punct — Default punctuation character on a run-in-head

Synopsis

```
<xsl:param name="runinhead.default.title.end.punct">.</xsl:param>
```

Description

If non-zero, For a `formalpara`, use the specified string as the separator between the title and following text. The period is the default value.

Name

runinhead.title.end.punct — Characters that count as punctuation on a run-in-head

Synopsis

```
<xsl:param name="runinhead.title.end.punct">.!?:</xsl:param>
```

Description

Specify which characters are to be counted as punctuation. These characters are checked for a match with the last character of the title. If no match is found, the

runinhead.default.title.end.punct contents are inserted. This is to avoid duplicated punctuation in the output.

Name

show.comments — Display `remark` elements?

Synopsis

```
<xsl:param name="show.comments" select="1"></xsl:param>
```

Description

If non-zero, comments will be displayed, otherwise they are suppressed. Comments here refers to the `remark` element (which was called `comment` prior to DocBook 4.0), not XML comments (<-- like this -->) which are unavailable.

Name

show.revisionflag — Enable decoration of elements that have a revisionflag

Synopsis

```
<xsl:param name="show.revisionflag" select="0"></xsl:param>
```

Description

If *show.revisionflag* is turned on, then the stylesheets may produce additional markup designed to allow a CSS stylesheet to highlight elements that have specific revisionflag settings.

The markup inserted will be usually be either a or with an appropriate `class` attribute. (The value of the class attribute will be the same as the value of the revisionflag attribute). In some contexts, for example tables, where extra markup would be structurally illegal, the class attribute will be added to the appropriate container element.

In general, the stylesheets only test for revisionflag in contexts where an importing stylesheet would have to redefine whole templates. Most of the revisionflag processing is expected to be done by another stylesheet, for example `changebars.xsl`.

Name

shade.verbatim — Should verbatim environments be shaded?

Synopsis

```
<xsl:param name="shade.verbatim" select="0"></xsl:param>
```

Description

In the FO stylesheet, if this parameter is non-zero then the shade.verbatim.style properties will be applied to verbatim environments.

In the HTML stylesheet, this parameter is now deprecated. Use CSS instead.

Name

shade.verbatim.style — Properties that specify the style of shaded verbatim listings

Synopsis

```
<xsl:attribute-set name="shade.verbatim.style">
  <xsl:attribute name="border">0</xsl:attribute>
  <xsl:attribute name="bgcolor">#E0E0E0</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties that specify the style of shaded verbatim listings. The parameters specified (the border and background color) are added to the styling of the xsl-fo output. A border might be specified as "thin black solid" for example. See xsl-fo[1]

Name

punct.honorific — Punctuation after an honorific in a personal name.

Synopsis

```
<xsl:param name="punct.honorific">.</xsl:param>
```

Description

This parameter specifies the punctuation that should be added after an honorific in a personal name.

Name

tex.math.in.alt — TeX notation used for equations

Synopsis

```
<xsl:param name="tex.math.in.alt"></xsl:param>
```

Description

If you want type math directly in TeX notation in equations, this parameter specifies notation used. Currently are supported two values -- `plain` and `latex`. Empty value means that you are not using TeX math at all.

Preferred way for including TeX alternative of math is inside of `textobject` element. Eg.:

```
<inlineequation>
<inlinemediaobject>
<imageobject>
<imagedata fileref="eq1.gif"/>
</imageobject>
<textobject><phrase>E=mc squared</phrase></textobject>
<textobject role="tex"><phrase>E=mc^2</phrase></textobject>
</inlinemediaobject>
</inlineequation>
```

If you are using `graphic` element, you can store TeX inside `alt` element:

```
<inlineequation>
<alt role="tex">a^2+b^2=c^2</alt>
<graphic fileref="a2b2c2.gif"/>
</inlineequation>
```

[1] http://www.w3.org/TR/2004/WD-xsl11-20041216/#border

If you want use this feature, you should process your FO with PassiveTeX, which only supports TeX math notation. When calling stylsheet, don't forget to specify also passivetex.extensions=1.

If you want equations in HTML, just process generated file `tex-math-equations.tex` by TeX or LaTeX. Then run dvi2bitmap program on result DVI file. You will get images for equations in your document.

Warning

This feature is useful for print/PDF output only if you use the obsolete and now unsupported PassiveTeX XSL-FO engine.

Related Parameters

`tex.math.delims`, *`passivetex.extensions`*, *`tex.math.file`*

More information

For how-to documentation on embedding TeX equations and generating output from them, see DocBook XSL: TCG, DBTeXMath[1].

Name

tex.math.file — Name of temporary file for generating images from equations

Synopsis

```
<xsl:param name="tex.math.file">tex-math-equations.tex</xsl:param>
```

Description

Name of auxiliary file for TeX equations. This file can be processed by dvi2bitmap to get bitmap versions of equations for HTML output.

Related Parameters

`tex.math.in.alt`, *`tex.math.delims`*,

More information

For how-to documentation on embedding TeX equations and generating output from them, see DocBook XSL: TCG, DBTeXMath[1].

Name

tex.math.delims — Should equations output for processing by TeX be surrounded by math mode delimiters?

Synopsis

```
<xsl:param name="tex.math.delims" select="1"></xsl:param>
```

Description

For compatibility with DSSSL based DBTeXMath from Allin Cottrell you should set this parameter to 0.

Related Parameters

`tex.math.in.alt`, *`passivetex.extensions`*

See Also

[1] http://www.sagehill.net/docbookxsl/TexMath.html
[1] http://www.sagehill.net/docbookxsl/TexMath.html

You can also use the `<?dbtex delims?>` processing instruction to control whether delimiters are output.

More information

For how-to documentation on embedding TeX equations and generating output from them, see DocBook XSL: TCG, DBTeXMath[1].

Name

pixels.per.inch — How many pixels are there per inch?

Synopsis

```
<xsl:param name="pixels.per.inch">90</xsl:param>
```

Description

When lengths are converted to pixels, this value is used to determine the size of a pixel. The default value is taken from the XSL Recommendation[1].

Name

points.per.em — Specify the nominal size of an em-space in points

Synopsis

```
<xsl:param name="points.per.em">10</xsl:param>
```

Description

The fixed value used for calculations based upon the size of a character. The assumption made is that ten point font is in use. This assumption may not be valid.

Name

use.svg — Allow SVG in the result tree?

Synopsis

```
<xsl:param name="use.svg" select="1"></xsl:param>
```

Description

If non-zero, SVG will be considered an acceptable image format. SVG is passed through to the result tree, so correct rendering of the resulting diagram depends on the formatter (FO processor or web browser) that is used to process the output from the stylesheet.

Name

menuchoice.separator — Separator between items of a `menuchoice` other than `guimenuitem` and `guisubmenu`

Synopsis

```
<xsl:param name="menuchoice.separator">+</xsl:param>
```

[1] http://www.sagehill.net/docbookxsl/TexMath.html
[1] http://www.w3.org/TR/2004/WD-xsl11-20041216/

Description

Separator used to connect items of a `menuchoice` other than `guimenuitem` and `guisubmenu`. The latter elements are linked with *`menuchoice.menu.separator`*.

Name

menuchoice.menu.separator — Separator between items of a `menuchoice` with `guimenuitem` or `guisubmenu`

Synopsis

```
<xsl:param name="menuchoice.menu.separator">   </xsl:param>
```

Description

Separator used to connect items of a `menuchoice` with `guimenuitem` or `guisubmenu`. Other elements are linked with *`menuchoice.separator`*.

The default value is →, which is the → (right arrow) character entity. The current FOP (0.20.5) requires setting the font-family explicitly.

The default value also includes spaces around the arrow, which will allow a line to break. Replace the spaces with   (nonbreaking space) if you don't want those spaces to break.

Name

default.float.class — Specifies the default float class

Synopsis

```
<xsl:param name="default.float.class">
  <xsl:choose>
    <xsl:when test="contains($stylesheet.result.type,'html')">left</xsl:when>
    <xsl:otherwise>before</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

Selects the direction in which a float should be placed. for xsl-fo this is before, for html it is left. For Western texts, the before direction is the top of the page.

Name

footnote.number.format — Identifies the format used for footnote numbers

Synopsis

```
<xsl:param name="footnote.number.format">1</xsl:param>
```

Description

The *`footnote.number.format`* specifies the format to use for footnote numeration (1, i, I, a, or A).

Name

table.footnote.number.format — Identifies the format used for footnote numbers in tables

Synopsis

```
<xsl:param name="table.footnote.number.format">a</xsl:param>
```

Description

The *table.footnote.number.format* specifies the format to use for footnote numeration (1, i, I, a, or A) in tables.

Name

footnote.number.symbols — Special characters to use as footnote markers

Synopsis

```
<xsl:param name="footnote.number.symbols"></xsl:param>
```

Description

If *footnote.number.symbols* is not the empty string, footnotes will use the characters it contains as footnote symbols. For example, “*†‡◊✠” will identify footnotes with “*”, “†”, “‡”, “◊”, and “✠”. If there are more footnotes than symbols, the stylesheets will fall back to numbered footnotes using *footnote.number.format*.

The use of symbols for footnotes depends on the ability of your processor (or browser) to render the symbols you select. Not all systems are capable of displaying the full range of Unicode characters. If the quoted characters in the preceding paragraph are not displayed properly, that's a good indicator that you may have trouble using those symbols for footnotes.

Name

table.footnote.number.symbols — Special characters to use a footnote markers in tables

Synopsis

```
<xsl:param name="table.footnote.number.symbols"></xsl:param>
```

Description

If *table.footnote.number.symbols* is not the empty string, table footnotes will use the characters it contains as footnote symbols. For example, “*†‡◊✠” will identify footnotes with “*”, “†”, “‡”, “◊”, and “✠”. If there are more footnotes than symbols, the stylesheets will fall back to numbered footnotes using *table.footnote.number.format*.

The use of symbols for footnotes depends on the ability of your processor (or browser) to render the symbols you select. Not all systems are capable of displaying the full range of Unicode characters. If the quoted characters in the preceding paragraph are not displayed properly, that's a good indicator that you may have trouble using those symbols for footnotes.

Name

highlight.source — Should the content of `programlisting` be syntactically highlighted?

Synopsis

```
<xsl:param name="highlight.source" select="0"></xsl:param>
```

Description

When this parameter is non-zero, the stylesheets will try to do syntax highlighting of the content of the `programlisting` element. The highlighting is done by the XSLTHL extension module. This is an external Java library which is not part of the DocBook XSL distribution.

In order to use this extension, you must add `xslthl.jar` to your Java classpath. You can download this software from the XSLT syntax highlighting project[1] at SourceForge.

The configuration of syntax highlighting is stored in `highlighting/xslthl-config.xml`. The Java property `xslthl.config` must point to this file (using URL syntax).

This extension is known to work with Saxon 6.5.x. Here is an example of a modified Saxon command:

```
java -cp c:\batch\;…;c:\path\to\xslthl.jar \
-Dxslthl.config=file:///c:/docbook-xsl/highlighting/xslthl-config.xml … \
com.icl.saxon.StyleSheet …
```

You can specify the language for each programlisting by using the `language` attribute. The *`highlighting.default.language`* parameter can be used for specifying the language to be used for programlistings without a `language` attribute.

Name

highlight.default.language — Default language of programlisting

Synopsis

```
<xsl:param name="highlight.default.language"></xsl:param>
```

Description

This language is used when there is no language attribute on programlisting.

Name

email.delimiters.enabled — Generate delimiters around email addresses?

Synopsis

```
<xsl:param name="email.delimiters.enabled" select="1"></xsl:param>
```

Description

If non-zero, delimiters [1] are generated around e-mail addresses (the output of the `email` element).

[1] http://sourceforge.net/projects/xslthl

[1] For delimiters, the stylesheets are currently hard-coded to output angle brackets.

Annotations

Name

annotation.support — Enable annotations?

Synopsis

```
<xsl:param name="annotation.support" select="0"></xsl:param>
```

Description

If non-zero, the stylesheets will attempt to support annotation elements in HTML by including some JavaScript (see *`annotation.js`*).

Name

annotation.js — URIs identifying JavaScript files with support for annotation popups

Synopsis

```
<xsl:param name="annotation.js">
<xsl:text>http://docbook.sourceforge.net/release/script/AnchorPosition.js \
http://docbook.sourceforge.net/release/script/PopupWindow.js</xsl:text></xsl:param>
```

Description

If annotation.support is enabled and the document contains `annotation`s, then the URIs listed in this parameter will be included. These JavaScript files are required for popup annotation support.

Name

annotation.css — CSS rules for annotations

Synopsis

```
<xsl:param name="annotation.css">
/* ====================================================================
   Annotations
*/

div.annotation-list  { visibility: hidden;
                     }

div.annotation-nocss { position: absolute;
                       visibility: hidden;
                     }

div.annotation-popup { position: absolute;
                       z-index: 4;
                       visibility: hidden;
                       padding: 0px;
                       margin: 2px;
                       border-style: solid;
                       border-width: 1px;
                       width: 200px;
         background-color: white;
                     }

div.annotation-title { padding: 1px;
```

```
                        font-weight: bold;
                        border-bottom-style: solid;
                        border-bottom-width: 1px;
          color: white;
          background-color: black;
                      }

div.annotation-body  { padding: 2px;
                      }

div.annotation-body p { margin-top: 0px;
                        padding-top: 0px;
                       }

div.annotation-close { position: absolute;
                        top: 2px;
                        right: 2px;
                      }
</xsl:param>
```

Description

If *annotation.support* is enabled and the document contains annotations, then the CSS in this parameter will be included in the document.

Name

annotation.graphic.open — Image for identifying a link that opens an annotation popup

Synopsis

```
<xsl:param \
name="annotation.graphic.open">http://docbook.sourceforge.net/release/images/annot-open.png</xsl:param>
```

Description

This image is used inline to identify the location of annotations. It may be replaced by a user provided graphic. The size should be approximately 10x10 pixels.

Name

annotation.graphic.close — Image for identifying a link that closes an annotation popup

Synopsis

```
<xsl:param name="annotation.graphic.close">
http://docbook.sourceforge.net/release/images/annot-close.png</xsl:param>
```

Description

This image is used on popup annotations as the “x” that the user can click to dismiss the popup.

This image is used on popup annotations as the “x” that the user can click to dismiss the popup. It may be replaced by a user provided graphic. The size should be approximately 10x10 pixels.

Graphics

Name

img.src.path — Path to HTML/FO image files

Synopsis

```
<xsl:param name="img.src.path"></xsl:param>
```

Description

Add a path prefix to each HTML `img` or FO `fo:external-graphic` element's `src` attribute. This path could be relative to the directory where the HTML/FO files are created, or it could be an absolute URI. The default value is empty. Be sure to include a trailing slash if needed.

This prefix is not applied to any filerefs that start with "/" or contain "//:".

Name

keep.relative.image.uris — Should image URIs be resolved against xml:base?

Synopsis

```
<xsl:param name="keep.relative.image.uris" select="1"></xsl:param>
```

Description

If non-zero, relative URIs (in, for example `fileref` attributes) will be used in the generated output. Otherwise, the URIs will be made absolute with respect to the base URI.

Note that the stylesheets calculate (and use) the absolute form for some purposes, this only applies to the resulting output.

Name

graphic.default.extension — Default extension for graphic filenames

Synopsis

```
<xsl:param name="graphic.default.extension"></xsl:param>
```

Description

If a `graphic` or `mediaobject` includes a reference to a filename that does not include an extension, and the `format` attribute is *unspecified*, the default extension will be used.

Name

default.image.width — The default width of images

Synopsis

```
<xsl:param name="default.image.width"></xsl:param>
```

Description

If specified, this value will be used for the `width` attribute on images that do not specify any viewport dimensions[1].

Name

nominal.image.width — The nominal image width

Synopsis

```
<xsl:param name="nominal.image.width" select="6 * $pixels.per.inch"></xsl:param>
```

Description

Graphic widths expressed as a percentage are problematic. In the following discussion, we speak of width and contentwidth, but the same issues apply to depth and contentdepth.

A width of 50% means "half of the available space for the image." That's fine. But note that in HTML, this is a dynamic property and the image size will vary if the browser window is resized.

A contentwidth of 50% means "half of the actual image width". But what does that mean if the stylesheets cannot assess the image's actual size? Treating this as a width of 50% is one possibility, but it produces behavior (dynamic scaling) that seems entirely out of character with the meaning.

Instead, the stylesheets define a *`nominal.image.width`* and convert percentages to actual values based on that nominal size.

Name

nominal.image.depth — Nominal image depth

Synopsis

```
<xsl:param name="nominal.image.depth" select="4 * $pixels.per.inch"></xsl:param>
```

Description

See *`nominal.image.width`*.

Name

use.embed.for.svg — Use HTML `embed` for SVG?

Synopsis

```
<xsl:param name="use.embed.for.svg" select="0"></xsl:param>
```

Description

If non-zero, an `embed` element will be created for SVG figures. An `object` is *always* created, this parameter merely controls whether or not an additional `embed` is generated inside the `object`.

On the plus side, this may be more portable among browsers and plug-ins. On the minus side, it isn't valid HTML.

Name

make.graphic.viewport — Use tables in HTML to make viewports for graphics

[1] http://docbook.org/tdg/en/html/imagedata.html#viewport.area

Synopsis

```
<xsl:param name="make.graphic.viewport" select="1"></xsl:param>
```

Description

The HTML `img` element only supports the notion of content-area scaling; it doesn't support the distinction between a content-area and a viewport-area, so we have to make some compromises.

If *`make.graphic.viewport`* is non-zero, a table will be used to frame the image. This creates an effective viewport-area.

Tables and alignment don't work together, so this parameter is ignored if alignment is specified on an image.

Name

preferred.mediaobject.role — Select which mediaobject to use based on this value of an object's `role` attribute.

Synopsis

```
<xsl:param name="preferred.mediaobject.role"></xsl:param>
```

Description

A mediaobject may contain several objects such as imageobjects. If the parameter *`use.role.for.mediaobject`* is non-zero, then the `role` attribute on `imageobject`s and other objects within a `mediaobject` container will be used to select which object will be used. If one of the objects has a role value that matches the *`preferred.mediaobject.role`* parameter, then it has first priority for selection. If more than one has such a role value, the first one is used.

See the *`use.role.for.mediaobject`* parameter for the sequence of selection.

Name

use.role.for.mediaobject — Use `role` attribute value for selecting which of several objects within a mediaobject to use.

Synopsis

```
<xsl:param name="use.role.for.mediaobject" select="1"></xsl:param>
```

Description

If non-zero, the `role` attribute on `imageobject`s or other objects within a `mediaobject` container will be used to select which object will be used.

The order of selection when then parameter is non-zero is:

1. If the stylesheet parameter *`preferred.mediaobject.role`* has a value, then the object whose role equals that value is selected.
2. Else if an object's role attribute has a value of `html` for HTML processing or `fo` for FO output, then the first of such objects is selected.
3. Else the first suitable object is selected.

If the value of *use.role.for.mediaobject* is zero, then role attributes are not considered and the first suitable object with or without a role value is used.

Name

ignore.image.scaling — Tell the stylesheets to ignore the author's image scaling attributes

Synopsis

```
<xsl:param name="ignore.image.scaling" select="0"></xsl:param>
```

Description

If non-zero, the scaling attributes on graphics and media objects are ignored.

Chunking

Name

chunker.output.cdata-section-elements — List of elements to escape with CDATA sections

Synopsis

```
<xsl:param name="chunker.output.cdata-section-elements"></xsl:param>
```

Description

This parameter specifies the list of elements that should be escaped as CDATA sections by the chunking stylesheet. Not all processors support specification of this parameter.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.doctype-public — Public identifer to use in the document type of generated pages

Synopsis

```
<xsl:param name="chunker.output.doctype-public"></xsl:param>
```

Description

This parameter specifies the public identifier that should be used by the chunking stylesheet in the document type declaration of chunked pages. Not all processors support specification of this parameter.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.doctype-system — System identifier to use for the document type in generated pages

Synopsis

```
<xsl:param name="chunker.output.doctype-system"></xsl:param>
```

Description

This parameter specifies the system identifier that should be used by the chunking stylesheet in the document type declaration of chunked pages. Not all processors support specification of this parameter.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.encoding — Encoding used in generated pages

Synopsis

```
<xsl:param name="chunker.output.encoding">ISO-8859-1</xsl:param>
```

Description

This parameter specifies the encoding to be used in files generated by the chunking stylesheet. Not all processors support specification of this parameter.

This parameter used to be named `default.encoding`.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.indent — Specification of indentation on generated pages

Synopsis

```
<xsl:param name="chunker.output.indent">no</xsl:param>
```

Description

This parameter specifies the value of the indent specification for generated pages. Not all processors support specification of this parameter.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.media-type — Media type to use in generated pages

Synopsis

```
<xsl:param name="chunker.output.media-type"></xsl:param>
```

Description

This parameter specifies the media type that should be used by the chunking stylesheet. Not all processors support specification of this parameter.

This parameter specifies the media type that should be used by the chunking stylesheet. This should be one from those defined in [RFC2045][1] and [RFC2046][2]

[1] http://www.ietf.org/rfc/rfc2045.txt
[2] http://www.ietf.org/rfc/rfc2046.txt

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

It must be one from html, xml or text

Name

chunker.output.method — Method used in generated pages

Synopsis

```
<xsl:param name="chunker.output.method">html</xsl:param>
```

Description

This parameter specifies the output method to be used in files generated by the chunking stylesheet.

This parameter used to be named `output.method`.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.omit-xml-declaration — Omit-xml-declaration for generated pages

Synopsis

```
<xsl:param name="chunker.output.omit-xml-declaration">no</xsl:param>
```

Description

This parameter specifies the value of the omit-xml-declaration specification for generated pages. Not all processors support specification of this parameter.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

chunker.output.standalone — Standalone declaration for generated pages

Synopsis

```
<xsl:param name="chunker.output.standalone">no</xsl:param>
```

Description

This parameter specifies the value of the standalone specification for generated pages. It must be either `yes` or `no`. Not all processors support specification of this parameter.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

saxon.character.representation — Saxon character representation used in generated HTML pages

Synopsis

```
<xsl:param name="saxon.character.representation" select="'entity;decimal'"></xsl:param>
```

Description

This parameter has effect only when Saxon 6 is used (version 6.4.2 or later). It sets the character representation in files generated by the chunking stylesheets. If you want to suppress entity references for characters with direct representations in *`chunker.output.encoding`*, set the parameter value to `native`.

Note

This parameter is documented here, but the declaration is actually in the `chunker.xsl` stylesheet module.

Name

html.ext — Identifies the extension of generated HTML files

Synopsis

```
<xsl:param name="html.ext">.html</xsl:param>
```

Description

The extension identified by *`html.ext`* will be used as the filename extension for chunks created by this stylesheet.

Name

use.id.as.filename — Use ID value of chunk elements as the filename?

Synopsis

```
<xsl:param name="use.id.as.filename" select="0"></xsl:param>
```

Description

If *`use.id.as.filename`* is non-zero, the filename of chunk elements that have IDs will be derived from the ID value.

Name

html.extra.head.links — Toggle extra HTML head link information

Synopsis

```
<xsl:param name="html.extra.head.links" select="0"></xsl:param>
```

Description

If non-zero, extra `link` elements will be generated in the `head` of chunked HTML files. These extra links point to chapters, appendixes, sections, etc. as supported by the “Site Navigation Bar” in Mozilla 1.0 (as of CR1, at least).

Name

root.filename — Identifies the name of the root HTML file when chunking

Synopsis

```
<xsl:param name="root.filename">index</xsl:param>
```

Description

The *`root.filename`* is the base filename for the chunk created for the root of each document processed.

Name

base.dir — The base directory of chunks

Synopsis

```
<xsl:param name="base.dir"></xsl:param>
```

Description

If specified, the *`base.dir`* identifies the output directory for chunks. (If not specified, the output directory is system dependent.)

Name

generate.manifest — Generate a manifest file?

Synopsis

```
<xsl:param name="generate.manifest" select="0"></xsl:param>
```

Description

If non-zero, a list of HTML files generated by the stylesheet transformation is written to the file named by the *`manifest`* parameter.

Name

manifest — Name of manifest file

Synopsis

```
    <xsl:param name="manifest">HTML.manifest</xsl:param>
    \
```

Description

The name of the file to which a manifest is written (if the value of the *`generate.manifest`* parameter is non-zero).

Name

manifest.in.base.dir — Should be manifest file written in $base.dir?

Synopsis

```
<xsl:param name="manifest.in.base.dir" select="0"></xsl:param>
```

Description

If non-zero manifest file and project files for HTML Help and Eclipse Help are written into *`base.dir`* instead of current directory.

Name

chunk.toc — An explicit TOC to be used for chunking

Synopsis

```
<xsl:param name="chunk.toc"></xsl:param>
```

Description

The *`chunk.toc`* identifies an explicit TOC that will be used for chunking. This parameter is only used by the `chunktoc.xsl` stylesheet (and customization layers built from it).

Name

chunk.tocs.and.lots — Should ToC and LoTs be in separate chunks?

Synopsis

```
<xsl:param name="chunk.tocs.and.lots" select="0"></xsl:param>
```

Description

If non-zero, ToC and LoT (List of Examples, List of Figures, etc.) will be put in a separate chunk. At the moment, this chunk is not in the normal forward/backward navigation list. Instead, a new link is added to the navigation footer.

This feature is still somewhat experimental. Feedback welcome.

Name

chunk.separate.lots — Should each LoT be in its own separate chunk?

Synopsis

```
<xsl:param name="chunk.separate.lots" select="0"></xsl:param>
```

Description

If non-zero, each of the ToC and LoTs (List of Examples, List of Figures, etc.) will be put in its own separate chunk. The title page includes generated links to each of the separate files.

This feature depends on the `chunk.tocs.and.lots` parameter also being non-zero.

Name

chunk.tocs.and.lots.has.title — Should ToC and LoTs in a separate chunks have title?

Synopsis

```
<xsl:param name="chunk.tocs.and.lots.has.title" select="1"></xsl:param>
```

Description

If non-zero title of document is shown before ToC/LoT in separate chunk.

Name

chunk.section.depth — Depth to which sections should be chunked

Synopsis

```
<xsl:param name="chunk.section.depth" select="1"></xsl:param>
```

Description

This parameter sets the depth of section chunking.

Name

chunk.first.sections — Chunk the first top-level section?

Synopsis

```
<xsl:param name="chunk.first.sections" select="0"></xsl:param>
```

Description

If non-zero, a chunk will be created for the first top-level `sect1` or `section` elements in each component. Otherwise, that section will be part of the chunk for its parent.

Name

chunk.quietly — Omit the chunked filename messages.

Synopsis

```
<xsl:param name="chunk.quietly" select="0"></xsl:param>
```

Description

If zero (the default), the XSL processor emits a message naming each separate chunk filename as it is being output. If nonzero, then the messages are suppressed.

Name

chunk.append — Specifies content to append to chunked HTML output

Synopsis

```
<xsl:param name="chunk.append"></xsl:param>
```

Description

Specifies content to append to the end of HTML files output by the `html/chunk.xsl` stylesheet, after the closing `<html>` tag. You probably don't want to set any value for this parameter; but if you do, the only value it should ever be set to is a newline character: `
` or `
`

Name

navig.graphics — Use graphics in navigational headers and footers?

Synopsis

```
<xsl:param name="navig.graphics" select="0"></xsl:param>
```

Description

If non-zero, the navigational headers and footers in chunked HTML are presented in an alternate style that uses graphical icons for Next, Previous, Up, and Home. Default graphics are provided in the distribution. If zero, text is used instead of graphics.

Name

navig.graphics.extension — Extension for navigational graphics

Synopsis

```
<xsl:param name="navig.graphics.extension">.gif</xsl:param>
```

Description

Sets the filename extension to use on navigational graphics used in the headers and footers of chunked HTML.

Name

navig.graphics.path — Path to navigational graphics

Synopsis

```
<xsl:param name="navig.graphics.path">images/</xsl:param>
```

Description

Sets the path, probably relative to the directory where the HTML files are created, to the navigational graphics used in the headers and footers of chunked HTML.

Name

navig.showtitles — Display titles in HTML headers and footers?

Synopsis

```
<xsl:param name="navig.showtitles">1</xsl:param>
```

Description

If non-zero, the headers and footers of chunked HTML display the titles of the next and previous chunks, along with the words 'Next' and 'Previous' (or the equivalent graphical icons if navig.graphics is true). If false (zero), then only the words 'Next' and 'Previous' (or the icons) are displayed.

Profiling

The following parameters can be used for attribute-based profiling of your document. FIXME: Add link to Bob's book.

Name

profile.arch — Target profile for `arch` attribute

Synopsis

```
<xsl:param name="profile.arch"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.audience — Target profile for `audience` attribute

Synopsis

```
<xsl:param name="profile.audience"></xsl:param>
```

Description

Value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.condition — Target profile for `condition` attribute

Synopsis

```
<xsl:param name="profile.condition"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.conformance — Target profile for `conformance` attribute

Synopsis

```
<xsl:param name="profile.conformance"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.lang — Target profile for `lang` attribute

Synopsis

```
<xsl:param name="profile.lang"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.os — Target profile for `os` attribute

Synopsis

```
<xsl:param name="profile.os"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.revision — Target profile for `revision` attribute

Synopsis

```
<xsl:param name="profile.revision"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.revisionflag — Target profile for `revisionflag` attribute

Synopsis

```
<xsl:param name="profile.revisionflag"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.role — Target profile for `role` attribute

Synopsis

```
<xsl:param name="profile.role"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Warning

Note that `role` is often used for other purposes than profiling. For example it is commonly used to get emphasize in bold font:

```
<emphasis role="bold">very important</emphasis>
```

If you are using `role` for these purposes do not forget to add values like `bold` to value of this parameter. If you forgot you will get document with small pieces missing which are very hard to track.

For this reason it is not recommended to use `role` attribute for profiling. You should rather use profiling specific attributes like `userlevel`, `os`, `arch`, `condition`, etc.

Name

profile.security — Target profile for `security` attribute

Synopsis

```
<xsl:param name="profile.security"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.status — Target profile for `status` attribute

Synopsis

```
<xsl:param name="profile.status"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.userlevel — Target profile for `userlevel` attribute

Synopsis

```
<xsl:param name="profile.userlevel"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.vendor — Target profile for `vendor` attribute

Synopsis

```
<xsl:param name="profile.vendor"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *profile.separator* parameter.

This parameter has effect only when you are using profiling stylesheets (profile-docbook.xsl, profile-chunk.xsl, …) instead of normal ones (docbook.xsl, chunk.xsl, …).

Name

profile.wordsize — Target profile for wordsize attribute

Synopsis

```
<xsl:param name="profile.wordsize"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *profile.separator* parameter.

This parameter has effect only when you are using profiling stylesheets (profile-docbook.xsl, profile-chunk.xsl, …) instead of normal ones (docbook.xsl, chunk.xsl, …).

Name

profile.attribute — Name of user-specified profiling attribute

Synopsis

```
<xsl:param name="profile.attribute"></xsl:param>
```

Description

This parameter is used in conjuction with *profile.value*.

This parameter has effect only when you are using profiling stylesheets (profile-docbook.xsl, profile-chunk.xsl, …) instead of normal ones (docbook.xsl, chunk.xsl, …).

Name

profile.value — Target profile for user-specified attribute

Synopsis

```
<xsl:param name="profile.value"></xsl:param>
```

Description

When you are using this parameter you must also specify name of profiling attribute with parameter *profile.attribute*.

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *profile.separator* parameter.

This parameter has effect only when you are using profiling stylesheets (profile-docbook.xsl, profile-chunk.xsl, …) instead of normal ones (docbook.xsl, chunk.xsl, …).

Name

profile.separator — Separator character for compound profile values

Synopsis

```
<xsl:param name="profile.separator">;</xsl:param>
```

Description

Separator character used for compound profile values. See *`profile.arch`*

HTML Help

Name

htmlhelp.encoding — Character encoding to use in files for HTML Help compiler.

Synopsis

```
<xsl:param name="htmlhelp.encoding">iso-8859-1</xsl:param>
```

Description

HTML Help Compiler is not UTF-8 aware, so you should always use an appropriate single-byte encoding here. Use one from iana[1], the registered charset values.

Name

htmlhelp.autolabel — Should tree-like ToC use autonumbering feature?

Synopsis

```
<xsl:param name="htmlhelp.autolabel" select="0"></xsl:param>
```

Description

Set this to non-zero to include chapter and section numbers into ToC in the left panel.

Name

htmlhelp.chm — Filename of output HTML Help file.

Synopsis

```
<xsl:param name="htmlhelp.chm">htmlhelp.chm</xsl:param>
```

Description

Set the name of resulting CHM file

Name

htmlhelp.default.topic — Name of file with default topic

Synopsis

```
<xsl:param name="htmlhelp.default.topic"></xsl:param>
```

Description

Normally first chunk of document is displayed when you open HTML Help file. If you want to display another topic, simply set its filename by this parameter.

This is useful especially if you don't generate ToC in front of your document and you also hide root element in ToC. E.g.:

[1] ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets

```
<xsl:param name="generate.book.toc" select="0"/>
<xsl:param name="htmlhelp.hhc.show.root" select="0"/>
<xsl:param name="htmlhelp.default.topic">pr01.html</xsl:param>
```

Name

htmlhelp.display.progress — Display compile progress?

Synopsis

```
<xsl:param name="htmlhelp.display.progress" select="1"></xsl:param>
```

Description

Set to non-zero to to display compile progress

Name

htmlhelp.hhp — Filename of project file.

Synopsis

```
<xsl:param name="htmlhelp.hhp">htmlhelp.hhp</xsl:param>
```

Description

Change this parameter if you want different name of project file than htmlhelp.hhp.

Name

htmlhelp.hhc — Filename of TOC file.

Synopsis

```
<xsl:param name="htmlhelp.hhc">toc.hhc</xsl:param>
```

Description

Set the name of the TOC file. The default is `toc.hhc`.

Name

htmlhelp.hhk — Filename of index file.

Synopsis

```
<xsl:param name="htmlhelp.hhk">index.hhk</xsl:param>
```

Description

set the name of the index file. The default is `index.hhk`.

Name

htmlhelp.hhp.tail — Additional content for project file.

Synopsis

```
<xsl:param name="htmlhelp.hhp.tail"></xsl:param>
```

Description

If you want to include some additional parameters into project file, store appropriate part of project file into this parameter.

Name

htmlhelp.hhp.window — Name of default window.

Synopsis

```
<xsl:param name="htmlhelp.hhp.window">Main</xsl:param>
```

Description

Name of default window. If empty no [WINDOWS] section will be added to project file.

Name

htmlhelp.hhp.windows — Definition of additional windows

Synopsis

```
<xsl:param name="htmlhelp.hhp.windows"></xsl:param>
```

Description

Content of this parameter is placed at the end of [WINDOWS] section of project file. You can use it for defining your own addtional windows.

Name

htmlhelp.enhanced.decompilation — Allow enhanced decompilation of CHM?

Synopsis

```
<xsl:param name="htmlhelp.enhanced.decompilation" select="0"></xsl:param>
```

Description

When non-zero this parameter enables enhanced decompilation of CHM.

Name

htmlhelp.enumerate.images — Should the paths to all used images be added to the project file?

Synopsis

```
<xsl:param name="htmlhelp.enumerate.images" select="0"></xsl:param>
```

Description

Set to non-zero if you insert images into your documents as external binary entities or if you are using absolute image paths.

Name

htmlhelp.force.map.and.alias — Should [MAP] and [ALIAS] sections be added to the project file unconditionally?

Synopsis

```
<xsl:param name="htmlhelp.force.map.and.alias" select="0"></xsl:param>
```

Description

Set to non-zero if you have your own `alias.h` and `context.h` files and you want to include references to them in the project file.

Name

htmlhelp.map.file — Filename of map file.

Synopsis

```
<xsl:param name="htmlhelp.map.file">context.h</xsl:param>
```

Description

Set the name of map file. The default is `context.h`. (used for context-sensitive help).

Name

htmlhelp.alias.file — Filename of alias file.

Synopsis

```
<xsl:param name="htmlhelp.alias.file">alias.h</xsl:param>
```

Description

Specifies the filename of the alias file (used for context-sensitive help).

Name

htmlhelp.hhc.section.depth — Depth of TOC for sections in a left pane.

Synopsis

```
<xsl:param name="htmlhelp.hhc.section.depth">5</xsl:param>
```

Description

Set the section depth in the left pane of HTML Help viewer.

Name

htmlhelp.hhc.show.root — Should there be an entry for the root element in the ToC?

Synopsis

```
<xsl:param name="htmlhelp.hhc.show.root" select="1"></xsl:param>
```

Description

If set to zero, there will be no entry for the root element in the ToC. This is useful when you want to provide the user with an expanded ToC as a default.

Name

htmlhelp.hhc.folders.instead.books — Use folder icons in ToC (instead of book icons)?

Synopsis

```
<xsl:param name="htmlhelp.hhc.folders.instead.books" select="1"></xsl:param>
```

Description

Set non-zero for folder-like icons or zero for book-like icons in the TOC ToC. If you want to use folder-like icons you must swith off binary ToC using (xref) *`htmlhelp.hhc.binary`*.

Name

htmlhelp.hhc.binary — Generate binary ToC?

Synopsis

```
<xsl:param name="htmlhelp.hhc.binary" select="1"></xsl:param>
```

Description

Set to non-zero to generate a binary TOC. You must create a binary TOC if you want to add Prev/Next buttons to toolbar (which is default behaviour). Files with binary TOC can't be merged.

Name

htmlhelp.hhc.width — Width of navigation pane

Synopsis

```
<xsl:param name="htmlhelp.hhc.width"></xsl:param>
```

Description

This parameter specifies the width of the navigation pane (containing TOC and other navigation tabs) in pixels.

Name

htmlhelp.title — Title of HTML Help

Synopsis

```
<xsl:param name="htmlhelp.title"></xsl:param>
```

Description

Content of this parameter will be used as a title for generated HTML Help. If empty, title will be automatically taken from document.

Name

htmlhelp.show.menu — Should the menu bar be shown?

Synopsis

```
<xsl:param name="htmlhelp.show.menu" select="0"></xsl:param>
```

Description

Set to non-zero to have an application menu bar in your HTML Help window.

Name

htmlhelp.show.toolbar.text — Show text under toolbar buttons?

Synopsis

```
<xsl:param name="htmlhelp.show.toolbar.text" select="1"></xsl:param>
```

Description

Set to non-zero to display texts under toolbar buttons, zero to switch off displays.

Name

htmlhelp.show.advanced.search — Should advanced search features be available?

Synopsis

```
<xsl:param name="htmlhelp.show.advanced.search" select="0"></xsl:param>
```

Description

If you want advanced search features in your help, turn this parameter to 1.

Name

htmlhelp.show.favorities — Should the Favorites tab be shown?

Synopsis

```
<xsl:param name="htmlhelp.show.favorities" select="0"></xsl:param>
```

Description

Set to non-zero to include a Favorites tab in the navigation pane of the help window.

Name

htmlhelp.button.hideshow — Should the Hide/Show button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.hideshow" select="1"></xsl:param>
```

Description

Set to non-zero to include the Hide/Show button shown on toolbar

Name

htmlhelp.button.back — Should the Back button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.back" select="1"></xsl:param>
```

Description

Set to non-zero to include the Hide/Show button shown on toolbar

Name

htmlhelp.button.forward — Should the Forward button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.forward" select="0"></xsl:param>
```

Description

Set to non-zero to include the Forward button on the toolbar.

Name

htmlhelp.button.stop — Should the Stop button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.stop" select="0"></xsl:param>
```

Description

If you want Stop button shown on toolbar, turn this parameter to 1.

Name

htmlhelp.button.refresh — Should the Refresh button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.refresh" select="0"></xsl:param>
```

Description

Set to non-zero to include the Stop button on the toolbar.

Name

htmlhelp.button.home — Should the Home button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.home" select="0"></xsl:param>
```

Description

Set to non-zero to include the Home button on the toolbar.

Name

htmlhelp.button.home.url — URL address of page accessible by Home button

Synopsis

```
<xsl:param name="htmlhelp.button.home.url"></xsl:param>
```

Description

URL address of page accessible by Home button.

Name

htmlhelp.button.options — Should the Options button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.options" select="1"></xsl:param>
```

Description

If you want Options button shown on toolbar, turn this parameter to 1.

Name

htmlhelp.button.print — Should the Print button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.print" select="1"></xsl:param>
```

Description

Set to non-zero to include the Print button on the toolbar.

Name

htmlhelp.button.locate — Should the Locate button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.locate" select="0"></xsl:param>
```

Description

If you want Locate button shown on toolbar, turn this parameter to 1.

Name

htmlhelp.button.jump1 — Should the Jump1 button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.jump1" select="0"></xsl:param>
```

Description

Set to non-zero to include the `Jump1` button on the toolbar.

Name

htmlhelp.button.jump1.url — URL address of page accessible by Jump1 button

Synopsis

```
<xsl:param name="htmlhelp.button.jump1.url"></xsl:param>
```

Description

URL address of page accessible by Jump1 button.

Name

htmlhelp.button.jump1.title — Title of Jump1 button

Synopsis

```
<xsl:param name="htmlhelp.button.jump1.title">User1</xsl:param>
```

Description

Title of Jump1 button.

Name

htmlhelp.button.jump2 — Should the Jump2 button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.jump2" select="0"></xsl:param>
```

Description

Set to non-zero to include the Jump2 button on the toolbar.

Name

htmlhelp.button.jump2.url — URL address of page accessible by Jump2 button

Synopsis

```
<xsl:param name="htmlhelp.button.jump2.url"></xsl:param>
```

Description

URL address of page accessible by Jump2 button.

Name

htmlhelp.button.jump2.title — Title of Jump2 button

Synopsis

```
<xsl:param name="htmlhelp.button.jump2.title">User2</xsl:param>
```

Description

Title of Jump2 button.

Name

htmlhelp.button.next — Should the Next button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.next" select="1"></xsl:param>
```

Description

Set to non-zero to include the Next button on the toolbar.

Name

htmlhelp.button.prev — Should the Prev button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.prev" select="1"></xsl:param>
```

Description

Set to non-zero to include the Prev button on the toolbar.

Name

htmlhelp.button.zoom — Should the Zoom button be shown?

Synopsis

```
<xsl:param name="htmlhelp.button.zoom" select="0"></xsl:param>
```

Description

Set to non-zero to include the Zoom button on the toolbar.

Name

htmlhelp.remember.window.position — Remember help window position?

Synopsis

```
<xsl:param name="htmlhelp.remember.window.position" select="0"></xsl:param>
```

Description

Set to non-zero to remember help window position between starts.

Name

htmlhelp.window.geometry — Set initial geometry of help window

Synopsis

```
<xsl:param name="htmlhelp.window.geometry"></xsl:param>
```

Description

This parameter specifies initial position of help window. E.g.

```
<xsl:param name="htmlhelp.window.geometry">[160,64,992,704]</xsl:param>
```

Name

htmlhelp.use.hhk — Should the index be built using the HHK file?

Synopsis

```
<xsl:param name="htmlhelp.use.hhk" select="0"></xsl:param>
```

Description

If non-zero, the index is created using the HHK file (instead of using `object` elements in the HTML files). For more information, see DocBook XSL: TCG, Generating an index[1].

Name

htmlhelp.only — Should only project files be generated?

Synopsis

```
<xsl:param name="htmlhelp.only" select="0"></xsl:param>
```

Description

Set to non-zero if you want to play with various HTML Help parameters and you don't need to regenerate all HTML files. This setting will not process whole document, only project files (hhp, hhc, hhk,...) will be generated.

[1] http://www.sagehill.net/docbookxsl/HtmlHelp.html#HHGenIndex

Eclipse Help Platform

Name

eclipse.autolabel — Should tree-like ToC use autonumbering feature?

Synopsis

```
<xsl:param name="eclipse.autolabel" select="0"></xsl:param>
```

Description

If you want to include chapter and section numbers into ToC in the left panel, set this parameter to 1.

Name

eclipse.plugin.name — Eclipse Help plugin name

Synopsis

```
<xsl:param name="eclipse.plugin.name">DocBook Online Help Sample</xsl:param>
```

Description

Eclipse Help plugin name.

Name

eclipse.plugin.id — Eclipse Help plugin id

Synopsis

```
<xsl:param name="eclipse.plugin.id">com.example.help</xsl:param>
```

Description

Eclipse Help plugin id. You should change this id to something unique for each help.

Name

eclipse.plugin.provider — Eclipse Help plugin provider name

Synopsis

```
<xsl:param name="eclipse.plugin.provider">Example provider</xsl:param>
```

Description

Eclipse Help plugin provider name.

JavaHelp

Name

javahelp.encoding — Character encoding to use in control files for JavaHelp.

Synopsis

```
<xsl:param name="javahelp.encoding">iso-8859-1</xsl:param>
```

Description

JavaHelp crashes on some characters when written as character references. In that case you can use this parameter to select an appropriate encoding.

Localization

Name

l10n.gentext.language — Sets the gentext language

Synopsis

```
<xsl:param name="l10n.gentext.language"></xsl:param>
```

Description

If this parameter is set to any value other than the empty string, its value will be used as the value for the language when generating text. Setting *`l10n.gentext.language`* overrides any settings within the document being formatted.

It's much more likely that you might want to set the *`l10n.gentext.default.language`* parameter.

Name

l10n.gentext.default.language — Sets the default language for generated text

Synopsis

```
<xsl:param name="l10n.gentext.default.language">en</xsl:param>
```

Description

The value of the *`l10n.gentext.default.language`* parameter is used as the language for generated text if no setting is provided in the source document.

Name

l10n.gentext.use.xref.language — Use the language of target when generating cross-reference text?

Synopsis

```
<xsl:param name="l10n.gentext.use.xref.language" select="0"></xsl:param>
```

Description

If non-zero, the language of the target will be used when generating cross reference text. Usually, the “current” language is used when generating text (that is, the language of the element that contains the cross-reference element). But setting this parameter allows the language of the element *pointed to* to control the generated text.

Consider the following example:

```
<para lang="en">See also <xref linkend="chap3"/>.</para>
```

Suppose that Chapter 3 happens to be written in German. If *`l10n.gentext.use.xref.language`* is non-zero, the resulting text will be something like this:

> See also Kapital 3.

Where the more traditional rendering would be:

See also Chapter 3.

Name

l10n.lang.value.rfc.compliant — Make value of lang attribute RFC compliant?

Synopsis

```
<xsl:param name="l10n.lang.value.rfc.compliant" select="1"></xsl:param>
```

Description

If non-zero, ensure that the values for all `lang` attributes in HTML output are RFC compliant[1]. by taking any underscore characters in any `lang` values found in source documents, and replacing them with hyphen characters in output HTML files. For example, `zh_CN` in a source document becomes `zh-CN` in the HTML output form that source.

Note

This parameter does not cause any case change in `lang` values, because RFC 1766 explicitly states that all "language tags" (as it calls them) "are to be treated as case insensitive".

[1] Section 8.1.1, Language Codes [http://www.w3.org/TR/REC-html40/struct/dirlang.html#h-8.1.1], in the HTML 4.0 Recommendation states that:

> [RFC1766] defines and explains the language codes that must be used in HTML documents.
>
> Briefly, language codes consist of a primary code and a possibly empty series of subcodes:
>
> ```
> language-code = primary-code ( "-" subcode )*
> ```
>
> And in RFC 1766, Tags for the Identification of Languages [http://www.ietf.org/rfc/rfc1766.txt], the EBNF for "language tag" is given as:
>
> ```
> Language-Tag = Primary-tag *( "-" Subtag )
> Primary-tag = 1*8ALPHA
> Subtag = 1*8ALPHA
> ```

Part II. FO Parameter Reference

This is reference documentation for all user-configurable parameters in the DocBook XSL FO stylesheets (for generating XSL-FO output destined for final print/PDF output).

Admonitions

Name

admon.graphics — Use graphics in admonitions?

Synopsis

```
<xsl:param name="admon.graphics" select="0"></xsl:param>
```

Description

If true (non-zero), admonitions are presented in an alternate style that uses a graphic. Default graphics are provided in the distribution.

Name

admon.graphics.extension — Extension for admonition graphics

Synopsis

```
<xsl:param name="admon.graphics.extension">.png</xsl:param>
```

Description

Sets the extension to use on admonition graphics.

Name

admon.graphics.path — Path to admonition graphics

Synopsis

```
<xsl:param name="admon.graphics.path">images/</xsl:param>
```

Description

Sets the path to the directory containing the admonition graphics (caution.png, important.png etc). This location is normally relative to the output html directory. See *`base.dir`*

Name

admon.textlabel — Use text label in admonitions?

Synopsis

```
<xsl:param name="admon.textlabel" select="1"></xsl:param>
```

Description

If true (non-zero), admonitions are presented with a generated text label such as Note or Warning in the appropriate language. If zero, such labels are turned off, but any title child of the admonition element are still output. The default value is 1.

Name

admonition.title.properties — To set the style for admonitions titles.

Synopsis

```
<xsl:attribute-set name="admonition.title.properties">
  <xsl:attribute name="font-size">14pt</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
</xsl:attribute-set>
```

Description

How do you want admonitions titles styled?

Set the font-size, weight etc to the style required.

Name

admonition.properties — To set the style for admonitions.

Synopsis

```
<xsl:attribute-set name="admonition.properties"></xsl:attribute-set>
```

Description

How do you want admonitions styled?

Set the font-size, weight, etc. to the style required

Name

graphical.admonition.properties — To add properties to the outer block of a graphical admonition.

Synopsis

```
<xsl:attribute-set name="graphical.admonition.properties">
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">1.2em</xsl:attribute>
</xsl:attribute-set>
```

Description

These properties are added to the outer block containing the entire graphical admonition, including its title. It is used when the parameter *`admon.graphics`* is set to nonzero. Use this attribute-set to set the space above and below, and any indent for the whole admonition.

In addition to these properties, a graphical admonition also applies the *`admonition.title.properties`* attribute-set to the title, and applies the *`admonition.properties`* attribute-set to the rest of the content.

Name

nongraphical.admonition.properties — To add properties to the outer block of a nongraphical admonition.

Synopsis

```
<xsl:attribute-set name="nongraphical.admonition.properties">
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
```

```
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="margin-left">0.25in</xsl:attribute>
  <xsl:attribute name="margin-right">0.25in</xsl:attribute>
</xsl:attribute-set>
```

Description

These properties are added to the outer block containing the entire nongraphical admonition, including its title. It is used when the parameter *admon.graphics* is set to zero. Use this attribute-set to set the space above and below, and any indent for the whole admonition.

In addition to these properties, a nongraphical admonition also applies the *admonition.title.properties* attribute-set to the title, and the *admonition.properties* attribute-set to the rest of the content.

Callouts

Name

callout.defaultcolumn — Indicates what column callouts appear in by default

Synopsis

```
<xsl:param name="callout.defaultcolumn">60</xsl:param>
```

Description

If a callout does not identify a column (for example, if it uses the `linerange unit`), it will appear in the default column.

Name

callout.graphics — Use graphics for callouts?

Synopsis

```
<xsl:param name="callout.graphics" select="1"></xsl:param>
```

Description

If non-zero, callouts are presented with graphics (e.g., reverse-video circled numbers instead of "(1)", "(2)", etc.). Default graphics are provided in the distribution.

Name

callout.graphics.extension — File name extension for callout graphics

Synopsis

```
<xsl:param name="callout.graphics.extension">.svg</xsl:param>
```

Description

Sets the extension to use on callout graphics, hence the callout graphic format. The appropriate format (and range used) should be available. svg, png and gif are provided.

Name

callout.graphics.number.limit — Number of the largest callout graphic

Synopsis

```
<xsl:param name="callout.graphics.number.limit">30</xsl:param>
```

Description

If *`callout.graphics`* is non-zero, graphics are used to represent callout numbers instead of plain text. The value of *`callout.graphics.number.limit`* is the largest number for which a graphic exists. If the callout number exceeds this limit, the default presentation "(plain text instead of a graphic)" will be used.

Name

callout.graphics.path — Path to callout graphics

Synopsis

```
<xsl:param name="callout.graphics.path">images/callouts/</xsl:param>
```

Description

Sets the path to the directory holding the callout graphics. his location is normally relative to the output html directory. see base.dir. Always terminate the directory with / since the graphic file is appended to this string, hence needs the separator.

Name

callout.icon.size — Specifies the size of callout marker icons

Synopsis

```
<xsl:param name="callout.icon.size">7pt</xsl:param>
```

Description

Specifies the size of the callout marker icons. The default size is 7 points.

Name

callout.unicode — Use Unicode characters rather than images for callouts.

Synopsis

```
<xsl:param name="callout.unicode" select="0"></xsl:param>
```

Description

The stylesheets can use either an image of the numbers one to ten, or the single Unicode character which represents the numeral, in white on a black background. Use this to select the Unicode character option.

Name

callout.unicode.font — Specify a font for Unicode glyphs

Synopsis

```
<xsl:param name="callout.unicode.font">ZapfDingbats</xsl:param>
```

Description

The name of the font to specify around Unicode callout glyphs. If set to the empty string, no font change will occur.

Name

callout.unicode.number.limit — Number of the largest unicode callout character

Synopsis

```
<xsl:param name="callout.unicode.number.limit">10</xsl:param>
```

Description

If *`callout.unicode`* is non-zero, unicode characters are used to represent callout numbers. The value of *`callout.unicode.number.limit`* is the largest number for which a unicode character exists. If the callout number exceeds this limit, the default presentation "(nnn)" will always be used.

Name

callout.unicode.start.character — First Unicode character to use, decimal value.

Synopsis

```
<xsl:param name="callout.unicode.start.character">10102</xsl:param>
```

Description

If *`callout.graphics`* is zero and *`callout.unicode`* is non-zero, unicode characters are used to represent callout numbers. The value of *`callout.unicode.start.character`* is the decimal unicode value used for callout number one. Currently, only 10102 is supported in the stylesheets for this parameter.

Name

callouts.extension — Enable the callout extension

Synopsis

```
<xsl:param name="callouts.extension" select="1"></xsl:param>
```

Description

The callouts extension processes `areaset` elements in `ProgramListingCO` and other text-based callout elements.

ToC/LoT/Index Generation

Name

autotoc.label.separator — Separator between labels and titles in the ToC

Synopsis

```
<xsl:param name="autotoc.label.separator">. </xsl:param>
```

Description

String to use to seperate labels and title in a table of contents.

Name

process.empty.source.toc — Generate automated TOC if `toc` element occurs in a source document?

Synopsis

```
<xsl:param name="process.empty.source.toc" select="0"></xsl:param>
```

Description

Specifies that if an empty `toc` element is found in a source document, an automated TOC is generated at this point in the document.

Note

Depending on what the value of the *`generate.toc`* parameter is, setting this parameter to `1` could result in generation of duplicate automated TOCs. So the *`process.empty.source.toc`* is primarily useful as an "override": by placing an empty `toc` in your document and setting this parameter to `1`, you can force a TOC to be generated even if `generate.toc` says not to.

Name

process.source.toc — Process a non-empty `toc` element if it occurs in a source document?

Synopsis

```
<xsl:param name="process.source.toc" select="0"></xsl:param>
```

Description

Specifies that the contents of a non-empty "hard-coded" `toc` element in a source document are processed to generate a TOC in output.

Note

This parameter has no effect on automated generation of TOCs. An automated TOC may still be generated along with the "hard-coded" TOC. To suppress automated TOC generation, adjust the value of the *`generate.toc`* paramameter.

The `process.source.toc` parameter also has no effect if the `toc` element is empty; handling for empty `toc` is controlled by the *`process.empty.source.toc`* parameter.

Name

generate.toc — Control generation of ToCs and LoTs

Synopsis

```
<xsl:param name="generate.toc">
/appendix toc,title
article/appendix  nop
/article  toc,title
book      toc,title,figure,table,example,equation
/chapter  toc,title
part      toc,title
/preface  toc,title
reference toc,title
/sect1    toc
/sect2    toc
/sect3    toc
/sect4    toc
/sect5    toc
/section  toc
set       toc,title
</xsl:param>
```

Description

This parameter has a structured value. It is a table of space-delimited path/value pairs. Each path identifies some element in the source document using a restricted subset of XPath (only the implicit child axis, no wildcards, no predicates). Paths can be either relative or absolute.

When processing a particular element, the stylesheets consult this table to determine if a ToC (or LoT(s)) should be generated.

For example, consider the entry:

```
book toc,figure
```

This indicates that whenever a `book` is formatted, a Table Of Contents and a List of Figures should be generated. Similarly,

```
/chapter toc
```

indicates that whenever a document *that has a root of* `chapter` is formatted, a Table of Contents should be generated. The entry `chapter` would match all chapters, but `/chapter` matches only `chapter` document elements.

Generally, the longest match wins. So, for example, if you want to distinguish articles in books from articles in parts, you could use these two entries:

```
book/article toc,figure
part/article toc
```

Note that an article in a part can never match a `book/article`, so if you want nothing to be generated for articles in parts, you can simply leave that rule out.

If you want to leave the rule in, to make it explicit that you're turning something off, use the value “nop”. For example, the following entry disables ToCs and LoTs for articles:

```
article nop
```

Do not simply leave the word “article” in the file without a matching value. That'd be just begging the silly little path/value parser to get confused.

Section ToCs are further controlled by the *generate.section.toc.level* parameter. For a given section level to have a ToC, it must have both an entry in *generate.toc* and be within the range enabled by *generate.section.toc.level*.

Name

generate.index — Do you want an index?

Synopsis

```
<xsl:param name="generate.index" select="1"></xsl:param>
```

Description

Specify if an index should be generated.

Name

make.index.markup — Generate XML index markup in the index?

Synopsis

```
<xsl:param name="make.index.markup" select="0"></xsl:param>
```

Description

This parameter enables a very neat trick for getting properly merged, collated back-of-the-book indexes. G. Ken Holman suggested this trick at Extreme Markup Languages 2002 and I'm indebted to him for it.

Jeni Tennison's excellent code in `autoidx.xsl` does a great job of merging and sorting `indexterm`s in the document and building a back-of-the-book index. However, there's one thing that it cannot reasonably be expected to do: merge page numbers into ranges. (I would not have thought that it could collate and suppress duplicate page numbers, but in fact it appears to manage that task somehow.)

Ken's trick is to produce a document in which the index at the back of the book is “displayed” in XML. Because the index is generated by the FO processor, all of the page numbers have been resolved. It's a bit hard to explain, but what it boils down to is that instead of having an index at the back of the book that looks like this:

A. ap1, 1, 2, 3

you get one that looks like this:

```
<indexdiv>A</indexdiv>
<indexentry>
<primaryie>ap1</primaryie>,
<phrase role="pageno">1</phrase>,
<phrase role="pageno">2</phrase>,
<phrase role="pageno">3</phrase>
</indexentry>
```

After building a PDF file with this sort of odd-looking index, you can extract the text from the PDF file and the result is a proper index expressed in XML.

Now you have data that's amenable to processing and a simple Perl script (such as `fo/pdf2index`) can merge page ranges and generate a proper index.

Finally, reformat your original document using this literal index instead of an automatically generated one and “bingo”!

Name

index.method — Select method used to group index entries in an index

Synopsis

```
<xsl:param name="index.method">basic</xsl:param>
```

Description

This parameter lets you select which method should be used to sort and group index entries in an index. Indexes in latin-based languages that have accented characters typically sort together accented words and unaccented words. Thus “Á” (A acute) would sort together with “A”, so both would appear in the “A” section of the index. Languages using other alphabets (such as Russian cyrillic) and languages using ideographic chararacters (such as Japanese) require grouping specific to the languages and alphabets.

The default indexing method is limited. It can group accented characters in latin-based languages only. It cannot handle non-latin alphabets or ideographic languages. The other indexing methods require extensions of one type or another, and do not work with all XSLT processors, which is why there are not used by default.

The three choices for indexing method are:

`basic`
: (default) Sort and groups words based only on the Latin alphabet. Words with accented latin letters will group and sort with their respective primary letter, but words in non-Latin alphabets will be put in the “Symbols” section of the index.

`kosek`
: Sort and groups words based on letter groups configured in the DocBook locale file for the given language. See, for example, the French locale file `common/fr.xml`. This method requires that the XSLT processor support the EXSLT extensions (most do). It also requires support for using user-defined functions in xsl:key (xsltproc does not).

  This method is suitable for any language for which you can list all the individual characters that should appear in each letter group in an index. It is probably not practical to use it for ideographic languages such as Chinese that have hundreds or thousands of characters.

  To use the kosek method, you must:

  1. Use a processor that supports its extensions, such as Saxon 6 or Xalan (xsltproc and Saxon 8 do not).
  2. Set the index.method parameter's value to “kosek”.
  3. Import the appropriate index extensions stylesheet module `fo/autoidx-kosek.xsl` or `html/autoidx-kosek.xsl` into your customization.

`kimber`
: This method uses extensions to the Saxon processor to implement sophisticated indexing processes. It uses its own configuration file, which can include information for any number of languages. Each language's configuration can group words using one of two processes. In the enumerated process similar to that used in the kosek method, you indicate the groupings character-by-character. In the between-key process, you specify the break-points in the sort order that should start a new group. The latter configuration is useful for ideographic languages such as Chinese, Japanese, and Korean. You can also define your own collation algorithms and how you want mixed Latin-alphabet words sorted.

- For a whitepaper describing the extensions, see: http://www.innodata-isogen.com/knowledge_center/white_papers/back_of_book_for_xsl_fo.pdf.
- To download the extension library, see http://www.innodata-isogen.com/knowledge_center/tools_downloads/i18nsupport.

To use the kimber method, you must:

1. Use Saxon (version 6 or 8) as your XSLT processor.
2. Install and configure the Innodata Isogen library, using the documentation that comes with it.
3. Set the index.method parameter's value to “kimber”.
4. Import the appropriate index extensions stylesheet module `fo/autoidx-kimber.xsl` or `html/autoidx-kimber.xsl` into your customization.

Name

index.on.type — Select indexterms based on `type` attribute value

Synopsis

```
<xsl:param name="index.on.type" select="0"></xsl:param>
```

Description

If non-zero, then an `index` element that has a `type` attribute value will contain only those `indexterm` elements with a matching `type` attribute value. If an `index` has no `type` attribute or it is blank, then the index will contain all `indexterm`s in the current scope.

If `index.on.type` is zero, then the `type` attribute has no effect on selecting indexterms for an index.

For those using DocBook version 4.2 or earlier, the `type` attribute is not available for index terms. However, you can achieve the same effect by using the `role` attribute in the same manner on `indexterm` and `index`, and setting the stylesheet parameter *`index.on.role`* to a nonzero value.

Name

index.on.role — Select indexterms based on `role` value

Synopsis

```
<xsl:param name="index.on.role" select="0"></xsl:param>
```

Description

If non-zero, then an `index` element that has a `role` attribute value will contain only those `indexterm` elements with a matching role value. If an `index` has no `role` attribute or it is blank, then the index will contain all `indexterm`s in the current scope.

If `index.on.role` is zero, then the `role` attribute has no effect on selecting indexterms for an index.

If you are using DocBook version 4.3 or later, you should use the `type` attribute instead of `role` on `indexterm` and `index`, and set the *`index.on.type`* to a nonzero value.

Name

index.preferred.page.properties — Properties used to emphasize page number references for significant index terms

Synopsis

```
<xsl:attribute-set name="index.preferred.page.properties">
  <xsl:attribute name="font-weight">bold</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties used to emphasize page number references for significant index terms (`significance=preffered`). Currently works only with XEP.

Name

index.entry.properties — Properties applied to the formatted entries in an index

Synopsis

```
<xsl:attribute-set name="index.entry.properties">
  <xsl:attribute name="start-indent">0pt</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is applied to the block containing the entries in a letter division in an index. It can be used to set the font-size, font-family, and other inheritable properties that will be applied to all index entries.

Name

index.div.title.properties — Properties associated with the letter headings in an index

Synopsis

```
<xsl:attribute-set name="index.div.title.properties">
  <xsl:attribute name="margin-left">0pt</xsl:attribute>
  <xsl:attribute name="font-size">14.4pt</xsl:attribute>
  <xsl:attribute name="font-family"><xsl:value-of \
select="$title.fontset"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.optimum"><xsl:value-of \
select="concat($body.font.master,'pt')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="space-before.minimum"><xsl:value-of \
select="concat($body.font.master,'pt * 0.8')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="space-before.maximum"><xsl:value-of \
select="concat($body.font.master,'pt * 1.2')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="start-indent">0pt</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is used on the letter headings that separate the divisions in an index.

Name

index.number.separator — Override for punctuation separating page numbers in index

Synopsis

```
<xsl:param name="index.number.separator"></xsl:param>
```

Description

This parameter permits you to override the text to insert between page references in a formatted index entry. Typically that would be a comma and a space.

Because this text may be locale dependent, this parameter's value is normally taken from a gentext template named 'number-separator' in the context 'index' in the stylesheet locale file for the language of the current document. This parameter can be used to override the gentext string, and would typically be used on the command line. This parameter would apply to all languages.

So this text string can be customized in two ways. You can reset the default gentext string using the *`local.l10n.xml`* parameter, or you can override the gentext with the content of this parameter. The content can be a simple string, or it can be something more complex such as a call-template.

In HTML index output, section title references are used instead of page number references. This punctuation appears between such section titles in an HTML index.

Name

index.range.separator — Override for punctuation separating the two numbers in a page range in index

Synopsis

```
<xsl:param name="index.range.separator"></xsl:param>
```

Description

This parameter permits you to override the text to insert between the two numbers of a page range in an index. This parameter is only used by those XSL-FO processors that support an extension for generating such page ranges (such as XEP).

Because this text may be locale dependent, this parameter's value is normally taken from a gentext template named 'range-separator' in the context 'index' in the stylesheet locale file for the language of the current document. This parameter can be used to override the gentext string, and would typically be used on the command line. This parameter would apply to all languages.

So this text string can be customized in two ways. You can reset the default gentext string using the *`local.l10n.xml`* parameter, or you can override the gentext with the content of this parameter. The content can be a simple string, or it can be something more complex such as a call-template.

In HTML index output, section title references are used instead of page number references. So there are no page ranges and this parameter has no effect.

Name

index.term.separator — Override for punctuation separating an index term from its list of page references in an index

Synopsis

```
<xsl:param name="index.term.separator"></xsl:param>
```

Description

This parameter permits you to override the text to insert between the end of an index term and its list of page references. Typically that might be a comma and a space.

Because this text may be locale dependent, this parameter's value is normally taken from a gentext template named 'term-separator' in the context 'index' in the stylesheet locale file for the language of the current document. This parameter can be used to override the gentext string, and would typically be used on the command line. This parameter would apply to all languages.

So this text string can be customized in two ways. You can reset the default gentext string using the *`local.l10n.xml`* parameter, or you can fill in the content for this normally empty override parameter. The content can be a simple string, or it can be something more complex such as a call-template. For fo output, it could be an `fo:leader` element to provide space of a specific length, or a dot leader.

Name

xep.index.item.properties — Properties associated with XEP index-items

Synopsis

```
<xsl:attribute-set name="xep.index.item.properties" \
use-attribute-sets="index.page.number.properties">
  <xsl:attribute name="merge-subsequent-page-numbers">true</xsl:attribute>
  <xsl:attribute name="link-back">true</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties associated with XEP index-items, which generate page numbers in an index processed by XEP. For more info see the XEP documentation section "Indexes" in `http://www.renderx.com/reference.html#Indexes`.

This attribute-set also adds by default any properties from the `index.page.number.properties` attribute-set.

Name

toc.section.depth — How deep should recursive `section`s appear in the TOC?

Synopsis

```
<xsl:param name="toc.section.depth">2</xsl:param>
```

Description

Specifies the depth to which recursive sections should appear in the TOC.

Name

toc.max.depth — How many levels should be created for each TOC?

Synopsis

```
<xsl:param name="toc.max.depth">8</xsl:param>
```

Description

Specifies the maximal depth of TOC on all levels.

Name

toc.indent.width — Amount of indentation for TOC entries

Synopsis

```
<xsl:param name="toc.indent.width">24</xsl:param>
<!-- inconsistant point specification? -->
```

Description

Specifies, in points, the distance by which each level of the TOC is indented from its parent.

This value is expressed in points, without a unit (in other words, it is a bare number). Using a bare number allows the stylesheet to perform calculations that would otherwise have to be performed by the FO processor because not all processors support expressions.

Name

toc.line.properties — Properties for lines in ToC and LoTs

Synopsis

```
<xsl:attribute-set name="toc.line.properties">
  <xsl:attribute name="text-align-last">justify</xsl:attribute>
  <xsl:attribute name="text-align">start</xsl:attribute>
  <xsl:attribute name="end-indent"><xsl:value-of select="concat($toc.indent.width, \
'pt')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="last-line-end-indent"><xsl:value-of select="concat('-', \
$toc.indent.width, 'pt')"></xsl:value-of></xsl:attribute>
</xsl:attribute-set>
```

Description

Properties which are applied to every line in ToC (or LoT). You can modify them in order to change appearance of all, or some lines. For example in order to make lines for chapters in bold specify the following in your customization layer.

```
<xsl:attribute-set name="toc.line.properties">
  <xsl:attribute name="font-weight">
    <xsl:when test="self::chapter | self::preface | self::appendix">bold</xsl:when>
    <xsl:otherwise>normal</xsl:otherwise>
  </xsl:attribute>
</xsl:attribute-set>
```

Name

toc.margin.properties — Margin properties used on Tables of Contents

Synopsis

```
<xsl:attribute-set name="toc.margin.properties">
  <xsl:attribute name="space-before.minimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">2em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">2em</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is used on Tables of Contents. These attributes are set on the wrapper that surrounds the ToC block, not on each individual lines.

Name

bridgehead.in.toc — Should bridgehead elements appear in the TOC?

Synopsis

```
<xsl:param name="bridgehead.in.toc" select="0"></xsl:param>
```

Description

If non-zero, `bridgehead`s appear in the TOC. Note that this option is not fully supported and may be removed in a future version of the stylesheets.

Name

simplesect.in.toc — Should `simplesect` elements appear in the TOC?

Synopsis

```
<xsl:param name="simplesect.in.toc" select="0"></xsl:param>
```

Description

If non-zero, `simplesect`s will be included in the TOC.

Name

generate.section.toc.level — Control depth of TOC generation in sections

Synopsis

```

<xsl:param name="generate.section.toc.level" select="0"></xsl:param>
```

Description

The *`generate.section.toc.level`* parameter controls the depth of `section` in which TOCs will be generated. Note that this is related to, but not the same as *`toc.section.depth`*, which controls the depth to which TOC entries will be generated in a given TOC.

If, for example, *`generate.section.toc.level`* is `3`, TOCs will be generated in first, second, and third level sections, but not in fourth level sections.

Processor Extensions

Name

arbortext.extensions — Enable Arbortext extensions?

Synopsis

```
<xsl:param name="arbortext.extensions" select="0"></xsl:param>
```

Description

If non-zero, Arbortext[1] extensions will be used.

This parameter can also affect which graphics file formats are supported

Name

axf.extensions — Enable XSL Formatter extensions?

Synopsis

```
<xsl:param name="axf.extensions" select="0"></xsl:param>
```

Description

If non-zero, XSL Formatter[1] extensions will be used. XSL Formatter extensions consists of PDF bookmarks, document information and better index processing.

This parameter can also affect which graphics file formats are supported

Name

fop.extensions — Enable FOP extensions for version 0.20.5 and earlier

Synopsis

```
<xsl:param name="fop.extensions" select="0"></xsl:param>
```

Description

If non-zero, extensions intended for FOP[1] version 0.20.5 and earlier will be used. At present, this consists of PDF bookmarks.

This parameter can also affect which graphics file formats are supported

If you are using a version of FOP beyond version 0.20.5, then use the *`fop1.extensions`* instead.

Name

fop1.extensions — Enable extensions for FOP version 1 and later

Synopsis

```
<xsl:param name="fop1.extensions" select="0"></xsl:param>
```

Description

[1] http://www.arbortext.com/
[1] http://www.antennahouse.com/
[1] http://xml.apache.org/fop/

If non-zero, extensions for FOP[1] version 1 and later will be used.

This parameter can also affect which graphics file formats are supported

The original *`fop.extensions`* should still be used for FOP version 0.20.5 and earlier.

Name

passivetex.extensions — Enable PassiveTeX extensions?

Synopsis

```
<xsl:param name="passivetex.extensions" select="0"></xsl:param>
```

Description

If non-zero, PassiveTeX[1] extensions will be used. At present, this consists of PDF bookmarks and sorted index terms.

This parameter can also affect which graphics file formats are supported

Note

PassiveTeX is incomplete and development has ceased. In most cases, another XSL-FO engine is probably a better choice.

Name

tex.math.in.alt — TeX notation used for equations

Synopsis

```
<xsl:param name="tex.math.in.alt"></xsl:param>
```

Description

If you want type math directly in TeX notation in equations, this parameter specifies notation used. Currently are supported two values -- `plain` and `latex`. Empty value means that you are not using TeX math at all.

Preferred way for including TeX alternative of math is inside of `textobject` element. Eg.:

```
<inlineequation>
<inlinemediaobject>
<imageobject>
<imagedata fileref="eq1.gif"/>
</imageobject>
<textobject><phrase>E=mc squared</phrase></textobject>
<textobject role="tex"><phrase>E=mc^2</phrase></textobject>
</inlinemediaobject>
</inlineequation>
```

If you are using `graphic` element, you can store TeX inside `alt` element:

```
<inlineequation>
<alt role="tex">a^2+b^2=c^2</alt>
<graphic fileref="a2b2c2.gif"/>
</inlineequation>
```

[1] http://xml.apache.org/fop/
[1] http://www.tei-c.org.uk/Software/passivetex/

If you want use this feature, you should process your FO with PassiveTeX, which only supports TeX math notation. When calling stylsheet, don't forget to specify also passivetex.extensions=1.

If you want equations in HTML, just process generated file `tex-math-equations.tex` by TeX or LaTeX. Then run dvi2bitmap program on result DVI file. You will get images for equations in your document.

Warning

This feature is useful for print/PDF output only if you use the obsolete and now unsupported PassiveTeX XSL-FO engine.

Related Parameters

`tex.math.delims`, *`passivetex.extensions`*, *`tex.math.file`*

More information

For how-to documentation on embedding TeX equations and generating output from them, see DocBook XSL: TCG, DBTeXMath[1].

Name

tex.math.delims — Should equations output for processing by TeX be surrounded by math mode delimiters?

Synopsis

```
<xsl:param name="tex.math.delims" select="1"></xsl:param>
```

Description

For compatibility with DSSSL based DBTeXMath from Allin Cottrell you should set this parameter to 0.

Related Parameters

`tex.math.in.alt`, *`passivetex.extensions`*

See Also

You can also use the `<?dbtex delims?>` processing instruction to control whether delimiters are output.

More information

For how-to documentation on embedding TeX equations and generating output from them, see DocBook XSL: TCG, DBTeXMath[1].

Name

xep.extensions — Enable XEP extensions?

Synopsis

```
<xsl:param name="xep.extensions" select="0"></xsl:param>
```

Description

If non-zero, XEP[1] extensions will be used. XEP extensions consists of PDF bookmarks, document information and better index processing.

[1] http://www.sagehill.net/docbookxsl/TexMath.html
[1] http://www.sagehill.net/docbookxsl/TexMath.html
[1] http://www.renderx.com/

This parameter can also affect which graphics file formats are supported

Stylesheet Extensions

Name

linenumbering.everyNth — Indicate which lines should be numbered

Synopsis

```
<xsl:param name="linenumbering.everyNth">5</xsl:param>
```

Description

If line numbering is enabled, everyNth line will be numbered. Note that numbering is one based, not zero based.

Name

linenumbering.extension — Enable the line numbering extension

Synopsis

```
<xsl:param name="linenumbering.extension" select="1"></xsl:param>
```

Description

If non-zero, verbatim environments (elements that have the format='linespecific' notation attribute: address, literallayout, programlisting, screen, synopsis) that specify line numbering will have, line numbers.

Name

linenumbering.separator — Specify a separator between line numbers and lines

Synopsis

```
<xsl:param name="linenumbering.separator"><xsl:text> </xsl:text></xsl:param>
```

Description

The separator is inserted between line numbers and lines in the verbatim environment. The default value is a single white space. Note the interaction with *`linenumbering.width`*

Name

linenumbering.width — Indicates the width of line numbers

Synopsis

```
<xsl:param name="linenumbering.width">3</xsl:param>
```

Description

If line numbering is enabled, line numbers will appear right justified in a field "width" characters wide.

Name

tablecolumns.extension — Enable the table columns extension function

Synopsis

```
<xsl:param name="tablecolumns.extension" select="1"></xsl:param>
```

Description

The table columns extension function adjusts the widths of table columns in the HTML result to more accurately reflect the specifications in the CALS table.

Name

textinsert.extension — Enables the textinsert extension element

Synopsis

```
<xsl:param name="textinsert.extension" select="1"></xsl:param>
```

Description

The textinsert extension element inserts the contents of a file into the result tree (as text).

Note

To use the textinsert extension element, you must use either Saxon or Xalan as your XSLT processor (it doesn't work with xsltproc), along with either the DocBook Saxon extensions or DocBook Xalan extensions (for more information about those extensions, see DocBook XSL: TCG, DocBook Saxon Extensions[1] and DocBook XSL: TCG, DocBook Xalan Extensions[2]), and you must set both the *`use.extensions`* and *`textinsert.extension`* parameters to `1`.

As an alternative to using the textinsert element, consider using an Xinclude element with the `parse="text"` attribute and value specified, as detailed in DocBook XSL: TCG, Using XInclude for text inclusions[3].

See Also

You can also use the `<?dbhtml-include href?>` processing instruction to insert external files — both files containing plain text and files with markup content (including HTML content).

More information

For how-to documentation on inserting contents of external code files and other text files into output, see DocBook XSL: TCG, External code files[4].

For guidelines on inserting contents of HTML files into output, see DocBook XSL: TCG, Inserting external HTML code[5].

Name

textdata.default.encoding — Default encoding of external text files which are included using textdata element

Synopsis

```
<xsl:param name="textdata.default.encoding"></xsl:param>
```

[1] http://www.sagehill.net/docbookxsl/InstallingAProcessor.html#SaxonExtensions
[2] http://www.sagehill.net/docbookxsl/InstallingAProcessor.html#XalanExtensions
[3] http://www.sagehill.net/docbookxsl/ExternalCode.html#XIncludeCode
[4] http://www.sagehill.net/docbookxsl/ExternalCode.html
[5] http://www.sagehill.net/docbookxsl/InsertExtHtml.html

Description

Specifies the encoding of any external text files included using `textdata` element. This value is used only when you do not specify encoding by the appropriate attribute directly on textdata. An empty string is interpreted as the system default encoding.

Name

use.extensions — Enable extensions

Synopsis

```
<xsl:param name="use.extensions" select="0"></xsl:param>
```

Description

If non-zero, extensions may be used. Each extension is further controlled by its own parameter. But if *`use.extensions`* is zero, no extensions will be used.

Automatic labelling

Name

appendix.autolabel — Specifies the labeling format for Appendix titles

Synopsis

```
<xsl:param name="appendix.autolabel">A</xsl:param>
```

Description

If non-zero, then appendices will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (upperalpha).

Name

chapter.autolabel — Specifies the labeling format for Chapter titles

Synopsis

```
<xsl:param name="chapter.autolabel" select="1"></xsl:param>
```

Description

If non-zero, then chapters will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (arabic).

Name

part.autolabel — Specifies the labeling format for Part titles

Synopsis

```
<xsl:param name="part.autolabel">I</xsl:param>
```

Description

If non-zero, then parts will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (upperroman).

Name

reference.autolabel — Specifies the labeling format for Reference titles

Synopsis

```
<xsl:param name="reference.autolabel">I</xsl:param>
```

Description

If non-zero, references will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any non-zero value other than the above will generate the default number format (upperroman).

Name

preface.autolabel — Specifices the labeling format for Preface titles

Synopsis

```
<xsl:param name="preface.autolabel" select="0"></xsl:param>
```

Description

If non-zero then prefaces will be numbered using the parameter value as the number format if the value matches one of the following:

1 or arabic
: Arabic numeration (1, 2, 3 ...).

A or upperalpha
: Uppercase letter numeration (A, B, C ...).

a or loweralpha
: Lowercase letter numeration (a, b, c ...).

I or upperroman
: Uppercase roman numeration (I, II, III ...).

i or lowerroman
: Lowercase roman letter numeration (i, ii, iii ...).

Any nonzero value other than the above will generate the default number format (arabic).

Name

section.autolabel — Are sections enumerated?

Synopsis

```
<xsl:param name="section.autolabel" select="0"></xsl:param>
```

Description

If true (non-zero), unlabeled sections will be enumerated.

Name

section.autolabel.max.depth — The deepest level of sections that are numbered.

Synopsis

```
<xsl:param name="section.autolabel.max.depth">8</xsl:param>
```

Description

When section numbering is turned on by the *`section.autolabel`* parameter, then this parameter controls the depth of `section` nesting that is numbered. Sections nested to a level deeper than this value will not be numbered.

Name

section.label.includes.component.label — Do section labels include the component label?

Synopsis

```
<xsl:param name="section.label.includes.component.label" select="0"></xsl:param>
```

Description

If non-zero, section labels are prefixed with the label of the component that contains them.

Name

label.from.part — Renumber components in each part?

Synopsis

```
<xsl:param name="label.from.part" select="0"></xsl:param>
```

Description

If *`label.from.part`* is non-zero, then numbering of components — `preface`, `chapter`, `appendix`, and `reference` (when `reference` occurs at the component level) — is re-started within each `part`.

If *`label.from.part`* is zero (the default), numbering of components is *not* re-started within each `part`; instead, components are numbered sequentially throughout each `book`, regardless of whether or not they occur within `part` instances.

Name

component.label.includes.part.label — Do component labels include the part label?

Synopsis

```
<xsl:param name="component.label.includes.part.label" select="0"></xsl:param>
```

Description

If non-zero, number labels for `chapter`, `appendix`, and other component elements are prefixed with the label of the part element that contains them. So you might see Chapter II.3 instead of Chapter 3. Also, the labels for formal elements such as `table` and `figure` will include the part label. If there is no part element container, then no prefix is generated.

This feature is most useful when the *`label.from.part`* parameter is turned on. In that case, there would be more than one `chapter` “1”, and the extra part label prefix will identify each chapter unambiguously.

XSLT Processing

Name

rootid — Specify the root element to format

Synopsis

```
<xsl:param name="rootid"></xsl:param>
```

Description

If *`rootid`* is not empty, it must be the value of an ID that occurs in the document being formatted. The entire document will be loaded and parsed, but formatting will begin at the element identified, rather than at the root. For example, this allows you to process only `chapter` 4 of a `book`.

Because the entire document is available to the processor, automatic numbering, cross references, and other dependencies are correctly resolved.

Meta/*Info

Name

make.single.year.ranges — Print single-year ranges (e.g., 1998-1999)

Synopsis

```
<xsl:param name="make.single.year.ranges" select="0"></xsl:param>
```

Description

If non-zero, year ranges that span a single year will be printed in range notation (1998-1999) instead of discrete notation (1998, 1999).

Name

make.year.ranges — Collate copyright years into ranges?

Synopsis

```
<xsl:param name="make.year.ranges" select="0"></xsl:param>
```

Description

If non-zero, multiple copyright `year` elements will be collated into ranges. This works only if each year number is put into a separate year element. The copyright element permits multiple year elements. The stylesheet will not successfully parse a complex year element such as `<year>2001,2002,2003</year>` into a range.

Name

author.othername.in.middle — Is `othername` in `author` a middle name?

Synopsis

```
<xsl:param name="author.othername.in.middle" select="1"></xsl:param>
```

Description

If non-zero, the `othername` of an `author` appears between the `firstname` and `surname`. Otherwise, `othername` is suppressed.

Reference Pages

Name

funcsynopsis.decoration — Decorate elements of a `funcsynopsis`?

Synopsis

```
<xsl:param name="funcsynopsis.decoration" select="1"></xsl:param>
```

Description

If non-zero, elements of the `funcsynopsis` will be decorated (e.g. rendered as bold or italic text). The decoration is controlled by templates that can be redefined in a customization layer.

Name

funcsynopsis.style — What style of `funcsynopsis` should be generated?

Synopsis

```
<xsl:param name="funcsynopsis.style">kr</xsl:param>
```

Description

If *`funcsynopsis.style`* is `ansi`, ANSI-style function synopses are generated for a `funcsynopsis`, otherwise K&R-style function synopses are generated.

Name

function.parens — Generate parens after a function?

Synopsis

```
<xsl:param name="function.parens" select="0"></xsl:param>
```

Description

If non-zero, the formatting of a `function` element will include generated parentheses.

Name

refentry.generate.name — Output NAME header before 'RefName'(s)?

Synopsis

```
<xsl:param name="refentry.generate.name" select="1"></xsl:param>
```

Description

If non-zero, a "NAME" section title is output before the list of 'RefName's. This parameter and *`refentry.generate.title`* are mutually exclusive. This means that if you change this parameter to zero, you should set *`refentry.generate.title`* to non-zero unless you want get quite strange output.

Name

refentry.generate.title — Output title before 'RefName'(s)?

Synopsis

```
<xsl:param name="refentry.generate.title" select="0"></xsl:param>
```

Description

If non-zero, the reference page title or first name is output before the list of 'RefName's. This parameter and *refentry.generate.name* are mutually exclusive. This means that if you change this parameter to non-zero, you should set *refentry.generate.name* to zero unless you want get quite strange output.

Name

refentry.pagebreak — Start each refentry on a new page

Synopsis

```
<xsl:param name="refentry.pagebreak" select="1"></xsl:param>
```

Description

If non-zero (the default), each `refentry` element will start on a new page. If zero, a page break will not be generated between `refentry` elements. The exception is when the `refentry` elements are children of a `part` element, in which case the page breaks are always retained. That is because a `part` element does not generate a page-sequence for its children, so each `refentry` must start its own page-sequence.

Name

refentry.title.properties — Title properties for a refentry title

Synopsis

```
<xsl:attribute-set name="refentry.title.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$title.font.family"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="font-size">18pt</xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="space-after">1em</xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1.0em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.4em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">0.6em</xsl:attribute>
  <xsl:attribute name="start-indent"><xsl:value-of \
select="$title.margin.left"></xsl:value-of></xsl:attribute>
</xsl:attribute-set>
```

Description

Formatting properties applied to the title generated for the `refnamediv` part of output for `refentry` when the value of the *refentry.generate.title* parameter is non-zero. The font size is supplied by the appropriate *section.levelX.title.properties* attribute-set, computed from the location of the `refentry` in the section hierarchy.

Note

This parameter has no effect on the the title generated for the `refnamediv` part of output for `refentry` when the value of the *`refentry.generate.name`* parameter is non-zero. By default, that title is formatted with the same properties as the titles for all other first-level children of `refentry`.

Name

refentry.xref.manvolnum — Output `manvolnum` as part of `refentry` cross-reference?

Synopsis

```
<xsl:param name="refentry.xref.manvolnum" select="1"></xsl:param>
```

Description

if non-zero, the `manvolnum` is used when cross-referencing `refentry`s, either with `xref` or `citerefentry`.

Name

refclass.suppress — Suppress display of refclass contents?

Synopsis

```
<xsl:param name="refclass.suppress" select="0"></xsl:param>
```

Description

If the value of *`refclass.suppress`* is non-zero, then display of `refclass` contents is suppressed in output.

Tables

Name

default.table.width — The default width of tables

Synopsis

```
<xsl:param name="default.table.width"></xsl:param>
```

Description

If non-zero, this value will be used for the `width` attribute on `table`s that do not specify an alternate width (with the dbhtml processing instruction).

Name

nominal.table.width — The (absolute) nominal width of tables

Synopsis

```
<xsl:param name="nominal.table.width">6in</xsl:param>
```

Description

In order to convert CALS column widths into HTML column widths, it is sometimes necessary to have an absolute table width to use for conversion of mixed absolute and relative widths. This value must be an absolute length (not a percentag).

Name

default.table.frame — The default framing of tables

Synopsis

```
<xsl:param name="default.table.frame">all</xsl:param>
```

Description

This value will be used when there is no frame attribute on the table.

Name

table.cell.padding

Synopsis

```
<xsl:attribute-set name="table.cell.padding">
  <xsl:attribute name="padding-left">2pt</xsl:attribute>
  <xsl:attribute name="padding-right">2pt</xsl:attribute>
  <xsl:attribute name="padding-top">2pt</xsl:attribute>
  <xsl:attribute name="padding-bottom">2pt</xsl:attribute>
</xsl:attribute-set>
```

Description

FIXME:

Name

table.frame.border.thickness — Specifies the thickness of the frame border

Synopsis

```
<xsl:param name="table.frame.border.thickness">0.5pt</xsl:param>
```

Description

Specifies the thickness of the border on the table's frame.

Name

table.frame.border.style

Synopsis

```
<xsl:param name="table.frame.border.style">solid</xsl:param>
```

Description

FIXME:

Name

table.frame.border.color

Synopsis

```
<xsl:param name="table.frame.border.color">black</xsl:param>
```

Description

FIXME:

Name

table.cell.border.thickness

Synopsis

```
<xsl:param name="table.cell.border.thickness">0.5pt</xsl:param>
```

Description

If non-zero, specifies the thickness of borders on table cells. The units are points. See CSS[1]

Name

table.cell.border.style

Synopsis

```
<xsl:param name="table.cell.border.style">solid</xsl:param>
```

[1] http://www.w3.org/TR/CSS21/box.html#border-width-properties

Description

FIXME:

Name

table.cell.border.color

Synopsis

```
<xsl:param name="table.cell.border.color">black</xsl:param>
```

Description

Set the color of table borders. If non-zero, the value is used for the border coloration. See CSS[1]. A `color` is either a keyword or a numerical RGB specification. Keywords are aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, and yellow.

Name

table.table.properties — Properties associated with a table

Synopsis

```
<xsl:attribute-set name="table.table.properties">
  <xsl:attribute name="border-before-width.conditionality">retain</xsl:attribute>
  <xsl:attribute name="border-collapse">collapse</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for tables. This parameter should really have been called `table.properties`, but that parameter name was inadvertantly established for the block-level properties of the table as a whole.

See also *`table.properties`*.

[1] http://www.w3.org/TR/CSS21/syndata.html#value-def-color

Linking

Name

current.docid — targetdoc identifier for the document being processed

Synopsis

```
<xsl:param name="current.docid"></xsl:param>
```

Description

When olinks between documents are resolved for HTML output, the stylesheet can compute the relative path between the current document and the target document. The stylesheet needs to know the `targetdoc` identifiers for both documents, as they appear in the *`target.database.document`* database file. This parameter passes to the stylesheet the targetdoc identifier of the current document, since that identifier does not appear in the document itself.

This parameter can also be used for print output. If an olink's `targetdoc` id differs from the `current.docid`, then the stylesheet can append the target document's title to the generated olink text. That identifies to the reader that the link is to a different document, not the current document. See also *`olink.doctitle`* to enable that feature.

Name

collect.xref.targets — Controls whether cross reference data is collected

Synopsis

```
<xsl:param name="collect.xref.targets">no</xsl:param>
```

Description

In order to resolve olinks efficiently, the stylesheets can generate an external data file containing information about all potential cross reference endpoints in a document. This parameter determines whether the collection process is run when the document is processed by the stylesheet. The default value is `no`, which means the data file is not generated during processing. The other choices are `yes`, which means the data file is created and the document is processed for output, and `only`, which means the data file is created but the document is not processed for output. See also *`targets.filename`*.

Name

insert.olink.page.number — Turns page numbers in olinks on and off

Synopsis

```
<xsl:param name="insert.olink.page.number">no</xsl:param>
```

Description

The value of this parameter determines if cross references made between documents with `olink` will include page number citations. In most cases this is only applicable to references in printed output.

The parameter has three possible values.

no
: No page number references will be generated for olinks.

yes
: Page number references will be generated for all `olink` references. The style of page reference may be changed if an `xrefstyle` attribute is used.

maybe
: Page number references will not be generated for an `olink` element unless it has an `xrefstyle` attribute whose value specifies a page reference.

Olinks that point to targets within the same document are treated as `xref`s, and controlled by the *`insert.xref.page.number`* parameter.

Page number references for olinks to external documents can only be inserted if the information exists in the olink database. This means each olink target element (`div` or `obj`) must have a `page` attribute whose value is its page number in the target document. The XSL stylesheets are not able to extract that information during processing because pages have not yet been created in XSLT transformation. Only the XSL-FO processor knows what page each element is placed on. Therefore some postprocessing must take place to populate page numbers in the olink database.

Name

insert.olink.pdf.frag — Add fragment identifiers for links into PDF files

Synopsis

```
<xsl:param name="insert.olink.pdf.frag" select="0"></xsl:param>
```

Description

The value of this parameter determines whether the cross reference URIs to PDF documents made with `olink` will include fragment identifiers.

When forming a URI to link to a PDF document, a fragment identifier (typically a '#' followed by an id value) appended to the PDF filename can be used by the PDF viewer to open the PDF file to a location within the document instead of the first page. However, not all PDF files have id values embedded in them, and not all PDF viewers can handle fragment identifiers.

If *`insert.olink.pdf.frag`* is set to a non-zero value, then any olink targeting a PDF file will have the fragment identifier appended to the URI. The URI is formed by concatenating the value of the *`olink.base.uri`* parameter, the value of the `baseuri` attribute from the `document` element in the olink database with the matching `targetdoc` value, and the value of the `href` attribute for the targeted element in the olink database. The `href` attribute contains the fragment identifier.

If *`insert.olink.pdf.frag`* is set to zero (the default value), then the `href` attribute from the olink database is not appended to PDF olinks, so the fragment identifier is left off. A PDF olink is any olink for which the `baseuri` attribute from the matching `document` element in the olink database ends with '.pdf'. Any other olinks will still have the fragment identifier added.

Name

olink.base.uri — Base URI used in olink hrefs

Synopsis

```
<xsl:param name="olink.base.uri"></xsl:param>
```

Description

When cross reference data is collected for resolving olinks, it may be necessary to prepend a base URI to each target's href. This parameter lets you set that base URI when cross reference data is collected.

This feature is needed when you want to link to a document that is processed without chunking. The output filename for such a document is not known to the XSL stylesheet; the only target information consists of fragment identifiers such as `#idref`. To enable the resolution of olinks between documents, you should pass the name of the HTML output file as the value of this parameter. Then the hrefs recorded in the cross reference data collection look like `outfile.html#idref`, which can be reached as links from other documents.

Name

olink.debug — Turn on debugging messages for olinks

Synopsis

```
<xsl:param name="olink.debug" select="0"></xsl:param>
```

Description

If non-zero, then each olink will generate several messages about how it is being resolved during processing. This is useful when an olink does not resolve properly and the standard error messages are not sufficient to find the problem.

You may need to read through the olink XSL templates to understand the context for some of the debug messages.

Name

olink.doctitle — show the document title for external olinks?

Synopsis

```
<xsl:param name="olink.doctitle">no</xsl:param>
```

Description

When olinks between documents are resolved, the generated text may not make it clear that the reference is to another document. It is possible for the stylesheets to append the other document's title to external olinks. For this to happen, two parameters must be set.

- This *`olink.doctitle`* parameter should be set to either `yes` or `maybe` to enable this feature.
- And you should also set the *`current.docid`* parameter to the document id for the document currently being processed for output.

Then if an olink's `targetdoc` id differs from the `current.docid` value, the stylesheet knows that it is a reference to another document and can append the target document's title to the generated olink text.

The text for the target document's title is copied from the olink database from the `ttl` element of the top-level `div` for that document. If that `ttl` element is missing or empty, no title is output.

The supported values for *`olink.doctitle`* are:

`yes`
: Always insert the title to the target document if it is not the current document.

`no`
: Never insert the title to the target document, even if requested in an `xrefstyle` attribute.

`maybe`
: Only insert the title to the target document, if requested in an `xrefstyle` attribute.

An `xrefstyle` attribute may override the global setting for individual olinks. The following values are supported in an `xrefstyle` attribute using the `select:` syntax:

`docname`
: Insert the target document name for this olink using the `docname` gentext template, but only if the value of *`olink.doctitle`* is `no`.

`docnamelong`
: Insert the target document name for this olink using the `docnamelong` gentext template, but only if the value of *`olink.doctitle`* is `no`.

`nodocname`
: Omit the target document name even if the value of *`olink.doctitle`* is `yes`.

Another way of inserting the target document name for a single olink is to employ an `xrefstyle` attribute using the `template:` syntax. The `%o` placeholder (the letter o, not zero) in such a template will be filled in with the target document's title when it is processed. This will occur regardless of the value of *`olink.doctitle`*.

Note that prior to version 1.66 of the XSL stylesheets, the allowed values for this parameter were 0 and 1. Those values are still supported and mapped to 'no' and 'yes', respectively.

Name

olink.lang.fallback.sequence — look up translated documents if olink not found?

Synopsis

```
<xsl:param name="olink.lang.fallback.sequence"></xsl:param>
```

Description

This parameter defines a list of lang values to search among to resolve olinks.

Normally an olink tries to resolve to a document in the same language as the olink itself. The language of an olink is determined by its nearest ancestor element with a `lang` attribute, otherwise the value of the *`l10n.gentext.default.lang`* parameter.

An olink database can contain target data for the same document in multiple languages. Each set of data has the same value for the `targetdoc` attribute in the `document` element in the database, but with a different `lang` attribute value.

When an olink is being resolved, the target is first sought in the document with the same language as the olink. If no match is found there, then this parameter is consulted for additional languages to try.

The *`olink.lang.fallback.sequence`* must be a whitespace separated list of lang values to try. The first one with a match in the olink database is used. The default value is empty.

For example, a document might be written in German and contain an olink with `targetdoc="adminguide"`. When the document is processed, the processor first looks for a target dataset in the olink database starting with:

```
<document targetdoc="adminguide" lang="de">.
```

If there is no such element, then the *`olink.lang.fallback.sequence`* parameter is consulted. If its value is, for example, “fr en”, then the processor next looks for `targetdoc="adminguide" lang="fr"`, and then for `targetdoc="adminguide" lang="en"`. If there is still no match, it looks for `targetdoc="adminguide"` with no lang attribute.

This parameter is useful when a set of documents is only partially translated, or is in the process of being translated. If a target of an olink has not yet been translated, then this parameter permits the processor to look for the document in other languages. This assumes the reader would rather have a link to a document in a different language than to have a broken link.

Name

olink.properties — Properties associated with the cross-reference text of an olink.

Synopsis

```
<xsl:attribute-set name="olink.properties">
  <xsl:attribute name="show-destination">replace</xsl:attribute>
</xsl:attribute-set>
```

Description

This `attribute set` is applied to the `fo:basic-link` element of an olink. It is not applied to the optional page number or optional title of the external document.

Name

prefer.internal.olink — Prefer a local olink reference to an external reference

Synopsis

```
<xsl:param name="prefer.internal.olink" select="0"></xsl:param>
```

Description

If you are re-using XML content modules in multiple documents, you may want to redirect some of your olinks. This parameter permits you to redirect an olink to the current document.

For example: you are writing documentation for a product, which includes 3 manuals: a little installation booklet (booklet.xml), a user guide (user.xml), and a reference manual (reference.xml). All 3 documents begin with the same introduction section (intro.xml) that contains a reference to the customization section (custom.xml) which is included in both user.xml and reference.xml documents.

How do you write the link to custom.xml in intro.xml so that it is interpreted correctly in all 3 documents?

- If you use xref, it will fail in user.xml.
- If you use olink (pointing to reference.xml), the reference in user.xml will point to the customization section of the reference manual, while it is actually available in user.xml.

If you set the *`prefer.internal.olink`* parameter to a non-zero value, then the processor will first look in the olink database for the olink's `targetptr` attribute value in document matching the *`current.docid`* parameter value. If it isn't found there, then it tries the document in the database with the `targetdoc` value that matches the olink's `targetdoc` attribute.

This feature permits an olink reference to resolve to the current document if there is an element with an id matching the olink's `targetptr` value. The current document's olink data must be included in the target database for this to work.

Caution

There is a potential for incorrect links if the same `id` attribute value is used for different content in different documents. Some of your olinks may be redirected to the current document when they shouldn't be. It is not possible to control individual olink instances.

Name

target.database.document — Name of master database file for resolving olinks

Synopsis

```
<xsl:param name="target.database.document">olinkdb.xml</xsl:param>
```

Description

To resolve olinks between documents, the stylesheets use a master database document that identifies the target datafiles for all the documents within the scope of the olinks. This parameter value is the URI of the master document to be read during processing to resolve olinks. The default value is `olinkdb.xml`.

The data structure of the file is defined in the `targetdatabase.dtd` DTD. The database file provides the high level elements to record the identifiers, locations, and relationships of documents. The cross reference data for individual documents is generally pulled into the database using system entity references or XIncludes. See also *`targets.filename`*.

Name

targets.filename — Name of cross reference targets data file

Synopsis

```
<xsl:param name="targets.filename">target.db</xsl:param>
```

Description

In order to resolve olinks efficiently, the stylesheets can generate an external data file containing information about all potential cross reference endpoints in a document. This parameter lets you change the name of the generated file from the default name `target.db`. The name must agree with that used in the target database used to resolve olinks during processing. See also *`target.database.document`*.

Name

use.local.olink.style — Process olinks using xref style of current document

Synopsis

```
<xsl:param name="use.local.olink.style" select="0"></xsl:param> \
```

Description

When cross reference data is collected for use by olinks, the data for each potential target includes one field containing a completely assembled cross reference string, as if it were an xref generated in that document. Other fields record the separate title, number, and element name of each target. When an olink is formed to a target from another document, the olink resolves to that preassembled string by default. If the *`use.local.olink.style`* parameter is set to non-zero, then instead the cross reference string is formed again from the target title, number, and element name, using the stylesheet processing the targeting document. Then olinks will match the xref style in the targeting document rather than in the target document. If both documents are processed with the same stylesheet, then the results will be the same.

Cross References

Name

insert.xref.page.number — Turns page numbers in xrefs on and off

Synopsis

```
<xsl:param name="insert.xref.page.number">no</xsl:param>
```

Description

The value of this parameter determines if cross references (`xref`s) in printed output will include page number citations. It has three possible values.

no
: No page number references will be generated.

yes
: Page number references will be generated for all `xref` elements. The style of page reference may be changed if an `xrefstyle` attribute is used.

maybe
: Page number references will not be generated for an `xref` element unless it has an `xrefstyle` attribute whose value specifies a page reference.

Name

xref.properties — Properties associated with cross-reference text

Synopsis

```
<xsl:attribute-set name="xref.properties">
</xsl:attribute-set>
```

Description

This attribute set is used to set properties on cross reference text.

Name

xref.label-title.separator — Punctuation or space separating label from title in xref

Synopsis

```
<xsl:param name="xref.label-title.separator">: </xsl:param>
```

Description

This parameter allows you to control the punctuation of certain types of generated cross reference text. When cross reference text is generated for an `xref` or `olink` element using an `xrefstyle` attribute that makes use of the `select:` feature, and the selected components include both label and title, then the value of this parameter is inserted between label and title in the output.

Name

xref.label-page.separator — Punctuation or space separating label from page number in xref

Synopsis

```
<xsl:param name="xref.label-page.separator"><xsl:text> </xsl:text></xsl:param>
```

Description

This parameter allows you to control the punctuation of certain types of generated cross reference text. When cross reference text is generated for an `xref` or `olink` element using an `xrefstyle` attribute that makes use of the `select:` feature, and the selected components include both label and page but no title, then the value of this parameter is inserted between label and page number in the output. If a title is included, then other separators are used.

Name

xref.title-page.separator — Punctuation or space separating title from page number in xref

Synopsis

```
<xsl:param name="xref.title-page.separator"><xsl:text> </xsl:text></xsl:param>
```

Description

This parameter allows you to control the punctuation of certain types of generated cross reference text. When cross reference text is generated for an `xref` or `olink` element using an `xrefstyle` attribute that makes use of the `select:` feature, and the selected components include both title and page number, then the value of this parameter is inserted between title and page number in the output.

Name

insert.link.page.number — Turns page numbers in link elements on and off

Synopsis

```
<xsl:param name="insert.link.page.number">no</xsl:param>
```

Description

The value of this parameter determines if cross references using the `link` element in printed output will include standard page number citations. It has three possible values.

no
: No page number references will be generated.

yes
: Page number references will be generated for all `link` elements. The style of page reference may be changed if an `xrefstyle` attribute is used.

maybe
: Page number references will not be generated for a `link` element unless it has an `xrefstyle` attribute whose value specifies a page reference.

Although the `xrefstyle` attribute can be used to turn the page reference on or off, it cannot be used to control the formatting of the page number as it can in `xref`. In `link` it will always format with the style established by the gentext template with `name="page.citation"` in the `l:context name="xref"`.

Lists

Name

compact.list.item.spacing — What space do you want between list items (when spacing="compact")?

Synopsis

```
<xsl:attribute-set name="compact.list.item.spacing">
  <xsl:attribute name="space-before.optimum">0em</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">0.2em</xsl:attribute>
</xsl:attribute-set>
```

Description

Specify what spacing you want between each list item when `spacing` is “`compact`”.

Name

itemizedlist.properties — Properties that apply to each list-block generated by itemizedlist.

Synopsis

```
<xsl:attribute-set name="itemizedlist.properties" \
use-attribute-sets="list.block.properties">
</xsl:attribute-set>
```

Description

Properties that apply to each fo:list-block generated by itemizedlist.

Name

itemizedlist.label.properties — Properties that apply to each label inside itemized list.

Synopsis

```
<xsl:attribute-set name="itemizedlist.label.properties">
</xsl:attribute-set>
```

Description

Properties that apply to each label inside itemized list. E.g.:

```
<xsl:attribute-set name="itemizedlist.label.properties">
  <xsl:attribute name="text-align">right</xsl:attribute>
</xsl:attribute-set>
```

Name

itemizedlist.label.width — The default width of the label (bullet) in an itemized list.

Synopsis

```
    <xsl:param name="itemizedlist.label.width">1.0em</xsl:param>
  \
```

Description

Specifies the default width of the label (usually a bullet or other symbol) in an itemized list. You can override the default value on any particular list with the “dbfo” processing instruction using the “label-width” pseudoattribute.

Name

list.block.properties — Properties that apply to each list-block generated by list.

Synopsis

```
<xsl:attribute-set name="list.block.properties">
  <xsl:attribute name="provisional-label-separation">0.2em</xsl:attribute>
  <xsl:attribute name="provisional-distance-between-starts">1.5em</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties that apply to each fo:list-block generated by itemizedlist/orderedlist.

Name

list.block.spacing — What spacing do you want before and after lists?

Synopsis

```
<xsl:attribute-set name="list.block.spacing">
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">1.2em</xsl:attribute>
</xsl:attribute-set>
```

Description

Specify the spacing required before and after a list. It is necessary to specify the space after a list block because lists can come inside of paras.

Name

list.item.spacing — What space do you want between list items?

Synopsis

```
<xsl:attribute-set name="list.item.spacing">
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
</xsl:attribute-set>
```

Description

Specify what spacing you want between each list item.

Name

orderedlist.properties — Properties that apply to each list-block generated by orderedlist.

Synopsis

```
<xsl:attribute-set name="orderedlist.properties" \
use-attribute-sets="list.block.properties">
  <xsl:attribute name="provisional-distance-between-starts">2em</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties that apply to each fo:list-block generated by orderedlist.

Name

orderedlist.label.properties — Properties that apply to each label inside ordered list.

Synopsis

```
<xsl:attribute-set name="orderedlist.label.properties">
</xsl:attribute-set>
```

Description

Properties that apply to each label inside ordered list. E.g.:

```
<xsl:attribute-set name="orderedlist.label.properties">
  <xsl:attribute name="text-align">right</xsl:attribute>
</xsl:attribute-set>
```

Name

orderedlist.label.width — The default width of the label (number) in an ordered list.

Synopsis

```
<xsl:param name="orderedlist.label.width">1.2em</xsl:param>
```

Description

Specifies the default width of the label (usually a number or sequence of numbers) in an ordered list. You can override the default value on any particular list with the “dbfo” processing instruction using the “label-width” pseudoattribute.

Name

variablelist.max.termlength — Specifies the longest term in variablelists

Synopsis

```
<xsl:param name="variablelist.max.termlength">24</xsl:param>
```

Description

In variablelists, the `listitem` is indented to leave room for the `term` elements. That indent may be computed if it is not specified with a `termlength` attribute on the `variablelist` element.

The computation counts characters in the `term` elements in the list to find the longest term. However, some terms are very long and would produce extreme indents. This parameter lets you set a maximum character count. Any terms longer than the maximum would line wrap. The default value is 24.

The character counts are converted to physical widths by multiplying by 0.50em. There will be some variability in how many actual characters fit in the space since some characters are wider than others.

Name

variablelist.term.separator — Text to separate `term`s within a multi-term `varlistentry`

Synopsis

```
<xsl:param name="variablelist.term.separator">, </xsl:param>
```

Description

When a `varlistentry` contains multiple `term` elements, the string specified in the value of the *`variablelist.term.separator`* parameter is placed after each `term` except the last.

Note

To generate a line break between multiple `term`s in a `varlistentry`, set a non-zero value for the *`variablelist.term.break.after`* parameter. If you do so, you may also want to set the value of the *`variablelist.term.separator`* parameter to an empty string (to suppress rendering of the default comma and space after each `term`).

Name

variablelist.term.break.after — Generate line break after each `term` within a multi-term `varlistentry`?

Synopsis

```
<xsl:param name="variablelist.term.break.after">0</xsl:param>
```

Description

Set a non-zero value for the *`variablelist.term.break.after`* parameter to generate a line break between `term`s in a multi-term `varlistentry`.

Note

If you set a non-zero value for *`variablelist.term.break.after`*, you may also want to set the value of the *`variablelist.term.separator`* parameter to an empty string (to suppress rendering of the default comma and space after each `term`).

QAndASet

Name

qandadiv.autolabel — Are divisions in QAndASets enumerated?

Synopsis

```
<xsl:param name="qandadiv.autolabel" select="1"></xsl:param>
```

Description

If non-zero, unlabeled qandadivs will be enumerated.

Name

qanda.inherit.numeration — Does enumeration of QandASet components inherit the numeration of parent elements?

Synopsis

```
<xsl:param name="qanda.inherit.numeration" select="1"></xsl:param>
```

Description

If non-zero, numbered `qandadiv` elements and `question` and `answer` inherit the enumeration of the ancestors of the `qandaset`.

Name

qanda.defaultlabel — Sets the default for defaultlabel on QandASet.

Synopsis

```
<xsl:param name="qanda.defaultlabel">number</xsl:param>
```

Description

If no `defaultlabel` attribute is specified on a `qandaset`, this value is used. It must be one of the legal values for the defaultlabel attribute, one from none, number or qanda. The default value is 'number'.

Meaning

qanda - `question`s are labeled “Q:” and `answer`s are labeled “A:”.

number - The entries are enumerated.

none - No distinguishing label precedes Questions or Answers.

Name

qanda.in.toc — Should qandaentry questions appear in the document table of contents?

Synopsis

```
<xsl:param name="qanda.in.toc" select="0"></xsl:param>
```

Description

If true (non-zero), then the generated table of contents for a document will include `qandaset` titles, `qandadiv` titles, and `question` elements. The default value (zero) excludes them from the TOC.

This parameter does not affect any tables of contents that may be generated inside a qandaset or qandadiv.

Name

qanda.nested.in.toc — Should nested answer/qandaentry instances appear in TOC?

Synopsis

```
<xsl:param name="qanda.nested.in.toc" select="0"></xsl:param>
```

Description

If non-zero, instances of `qandaentry` that are children of `answer` elements are shown in the TOC.

Bibliography

Name

bibliography.style — Style used for formatting of biblioentries.

Synopsis

```
<xsl:param name="bibliography.style">normal</xsl:param>
```

Description

Currently only `normal` and `iso690` styles are supported.

In order to use ISO690 style to the full extent you might need to use additional markup described on the following WiKi page[1].

Name

biblioentry.item.separator — Text to separate bibliography entries

Synopsis

```
<xsl:param name="biblioentry.item.separator">. </xsl:param>
```

Description

Text to separate bibliography entries

Name

bibliography.collection — Name of the bibliography collection file

Synopsis

```
<xsl:param \
name="bibliography.collection">http://docbook.sourceforge.net/release/bibliography/bibliography.xml</xsl:param>
```

Description

Maintaining bibliography entries across a set of documents is tedious, time consuming, and error prone. It makes much more sense, usually, to store all of the bibliography entries in a single place and simply “extract” the ones you need in each document.

That's the purpose of the *`bibliography.collection`* parameter. To setup a global bibliography “database”, follow these steps:

First, create a stand-alone bibliography document that contains all of the documents that you wish to reference. Make sure that each bibliography entry (whether you use `biblioentry` or `bibliomixed`) has an ID.

My global bibliography, `~/bibliography.xml` begins like this:

```
<!DOCTYPE bibliography
  PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
```

[1] http://wiki.docbook.org/topic/ISO690Bibliography

```
  "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
<bibliography><title>References</title>

<bibliomixed id="xml-rec"><abbrev>XML 1.0</abbrev>Tim Bray,
Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler, editors.
<citetitle><ulink url="http://www.w3.org/TR/REC-xml">Extensible Markup
Language (XML) 1.0 Second Edition</ulink></citetitle>.
World Wide Web Consortium, 2000.
</bibliomixed>

<bibliomixed id="xml-names"><abbrev>Namespaces</abbrev>Tim Bray,
Dave Hollander,
and Andrew Layman, editors.
<citetitle><ulink url="http://www.w3.org/TR/REC-xml-names/">Namespaces in
XML</ulink></citetitle>.
World Wide Web Consortium, 1999.
</bibliomixed>

<!-- ... -->
</bibliography>
```

When you create a bibliography in your document, simply provide *empty* `bibliomixed` entries for each document that you wish to cite. Make sure that these elements have the same ID as the corresponding “real” entry in your global bibliography.

For example:

```
<bibliography><title>Bibliography</title>

<bibliomixed id="xml-rec"/>
<bibliomixed id="xml-names"/>
<bibliomixed id="DKnuth86">Donald E. Knuth. <citetitle>Computers and
Typesetting: Volume B, TeX: The Program</citetitle>. Addison-Wesley,
1986.  ISBN 0-201-13437-3.
</bibliomixed>
<bibliomixed id="relaxng"/>

</bibliography>
```

Note that it's perfectly acceptable to mix entries from your global bibliography with “normal” entries. You can use `xref` or other elements to cross-reference your bibliography entries in exactly the same way you do now.

Finally, when you are ready to format your document, simply set the *`bibliography.collection`* parameter (in either a customization layer or directly through your processor's interface) to point to your global bibliography.

The stylesheets will format the bibliography in your document as if all of the entries referenced appeared there literally.

Name

bibliography.numbered — Should bibliography entries be numbered?

Synopsis

```
<xsl:param name="bibliography.numbered" select="0"></xsl:param>
```

Description

If non-zero bibliography entries will be numbered

Name

biblioentry.properties — To set the style for biblioentry.

Synopsis

```
<xsl:attribute-set name="biblioentry.properties" \
use-attribute-sets="normal.para.spacing">
  <xsl:attribute name="start-indent">0.5in</xsl:attribute>
  <xsl:attribute name="text-indent">-0.5in</xsl:attribute>
</xsl:attribute-set>
```

Description

How do you want biblioentry styled?

Set the font-size, weight, space-above and space-below, indents, etc. to the style required

Glossary

Name

glossterm.auto.link — Generate links from glossterm to glossentry automaticaly?

Synopsis

```
<xsl:param name="glossterm.auto.link" select="0"></xsl:param>
```

Description

If true, a link will be automatically created from glossterm to glossentry for that glossary term. This is usefull when your glossterm names are consistent and you don't want to add links manually.

If there is `linkend` on `glossterm` then is used instead of autogeneration of link.

Name

firstterm.only.link — Does automatic glossterm linking only apply to firstterms?

Synopsis

```
<xsl:param name="firstterm.only.link" select="0"></xsl:param>
```

Description

If non-zero, only `firstterm`s will be automatically linked to the glossary. If glossary linking is not enabled, this parameter has no effect.

Name

glossary.collection — Name of the glossary collection file

Synopsis

```
<xsl:param name="glossary.collection"></xsl:param>
```

Description

Glossaries maintained independently across a set of documents are likely to become inconsistent unless considerable effort is expended to keep them in sync. It makes much more sense, usually, to store all of the glossary entries in a single place and simply “extract” the ones you need in each document.

That's the purpose of the *`glossary.collection`* parameter. To setup a global glossary “database”, follow these steps:

Setting Up the Glossary Database

First, create a stand-alone glossary document that contains all of the entries that you wish to reference. Make sure that each glossary entry has an ID.

Here's an example glossary:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE glossary
  PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
```

```
<glossary>
<glossaryinfo>
<editor><firstname>Eric</firstname><surname>Raymond</surname></editor>
<title>Jargon File 4.2.3 (abridged)</title>
<releaseinfo>Just some test data</releaseinfo>
</glossaryinfo>

<glossdiv><title>0</title>

<glossentry>
<glossterm>0</glossterm>
<glossdef>
<para>Numeric zero, as opposed to the letter `O' (the 15th letter of
the English alphabet). In their unmodified forms they look a lot
alike, and various kluges invented to make them visually distinct have
compounded the confusion. If your zero is center-dotted and letter-O
is not, or if letter-O looks almost rectangular but zero looks more
like an American football stood on end (or the reverse), you're
probably looking at a modern character display (though the dotted zero
seems to have originated as an option on IBM 3270 controllers). If
your zero is slashed but letter-O is not, you're probably looking at
an old-style ASCII graphic set descended from the default typewheel on
the venerable ASR-33 Teletype (Scandinavians, for whom /O is a letter,
curse this arrangement). (Interestingly, the slashed zero long
predates computers; Florian Cajori's monumental "A History of
Mathematical Notations" notes that it was used in the twelfth and
thirteenth centuries.) If letter-O has a slash across it and the zero
does not, your display is tuned for a very old convention used at IBM
and a few other early mainframe makers (Scandinavians curse <emphasis>this</emphasis>
arrangement even more, because it means two of their letters collide).
Some Burroughs/Unisys equipment displays a zero with a <emphasis>reversed</emphasis>
slash. Old CDC computers rendered letter O as an unbroken oval and 0
as an oval broken at upper right and lower left. And yet another
convention common on early line printers left zero unornamented but
added a tail or hook to the letter-O so that it resembled an inverted
Q or cursive capital letter-O (this was endorsed by a draft ANSI
standard for how to draw ASCII characters, but the final standard
changed the distinguisher to a tick-mark in the upper-left corner).
Are we sufficiently confused yet?</para>
</glossdef>
</glossentry>

<glossentry>
<glossterm>1TBS</glossterm>
<glossdef>
<para role="accidence">
<phrase role="pronounce"></phrase>
<phrase role="partsofspeach">n</phrase>
</para>
<para>The "One True Brace Style"</para>
<glossseealso>indent style</glossseealso>
</glossdef>
</glossentry>

<!-- ... -->

</glossdiv>

<!-- ... -->

</glossary>
```

Marking Up Glossary Terms

That takes care of the glossary database, now you have to get the entries into your document. Unlike bibliography entries, which can be empty, creating “placeholder” glossary entries would be very tedious. So instead, support for *`glossary.collection`* relies on implicit linking.

In your source document, simply use `firstterm` and `glossterm` to identify the terms you wish to have included in the glossary. The stylesheets assume that you will either set the `baseform` attribute correctly, or that the content of the element exactly matches a term in your glossary.

If you're using a *`glossary.collection`*, don't make explicit links on the terms in your document.

So, in your document, you might write things like this:

```
<para>This is dummy text, without any real meaning.
The point is simply to reference glossary terms like <glossterm>0</glossterm>
and the <firstterm baseform="1TBS">One True Brace Style (1TBS)</firstterm>.
The <glossterm>1TBS</glossterm>, as you can probably imagine, is a nearly
religious issue.</para>
```

If you set the *`firstterm.only.link`* parameter, only the terms marked with `firstterm` will be links. Otherwise, all the terms will be linked.

Marking Up the Glossary

The glossary itself has to be identified for the stylesheets. For lack of a better choice, the `role` is used. To identify the glossary as the target for automatic processing, set the role to "`auto`". The title of this glossary (and any other information from the `glossaryinfo` that's rendered by your stylesheet) will be displayed, but the entries will come from the database.

Unfortunately, the glossary can't be empty, so you must put in at least one `glossentry`. The content of this entry is irrelevant, it will not be rendered:

```
<glossary role="auto">
<glossentry>
<glossterm>Irrelevant</glossterm>
<glossdef>
<para>If you can see this, the document was processed incorrectly. Use
the <parameter>glossary.collection</parameter> parameter.</para>
</glossdef>
</glossentry>
</glossary>
```

What about glossary divisions? If your glossary database has glossary divisions *and* your automatic glossary contains at least one `glossdiv`, the automic glossary will have divisions. If the `glossdiv` is missing from either location, no divisions will be rendered.

Glossary entries (and divisions, if appropriate) in the glossary will occur in precisely the order they occur in your database.

Formatting the Document

Finally, when you are ready to format your document, simply set the *`glossary.collection`* parameter (in either a customization layer or directly through your processor's interface) to point to your global glossary.

The stylesheets will format the glossary in your document as if all of the entries implicilty referenced appeared there literally.

Limitations

Glossary cross-references *within the glossary* are not supported. For example, this *will not* work:

```
<glossentry>
<glossterm>gloss-1</glossterm>
<glossdef><para>A description that references <glossterm>gloss-2</glossterm>.</para>
<glossseealso>gloss-2</glossseealso>
</glossdef>
</glossentry>
```

If you put glossary cross-references in your glossary that way, you'll get the cryptic error: `Warning: glossary.collection specified, but there are 0 automatic glossaries`.

Instead, you must do two things:

1. Markup your glossary using `glossseealso`:

```
<glossentry>
<glossterm>gloss-1</glossterm>
<glossdef><para>A description that references <glossterm>gloss-2</glossterm>.</para>
<glossseealso>gloss-2</glossseealso>
</glossdef>
</glossentry>
```

2. Make sure there is at least one `glossterm` reference to *gloss-2 in your document*. The easiest way to do that is probably within a `remark` in your automatic glossary:

```
<glossary role="auto">
<remark>Make sure there's a reference to <glossterm>gloss-2</glossterm>.</remark>
<glossentry>
<glossterm>Irrelevant</glossterm>
<glossdef>
<para>If you can see this, the document was processed incorrectly. Use
the <parameter>glossary.collection</parameter> parameter.</para>
</glossdef>
</glossentry>
</glossary>
```

Name

glossterm.separation — Separation between glossary terms and descriptions in list mode

Synopsis

```
<xsl:param name="glossterm.separation">0.25in</xsl:param>
```

Description

Specifies the separation between glossary terms and descriptions when glossarys are presented using lists.

Name

glossterm.width — Width of glossterm in list presentation mode

Synopsis

```
<xsl:param name="glossterm.width">2in</xsl:param>
```

Description

This parameter specifies the width reserved for glossary terms when a list presentation is used.

Name

glossary.as.blocks — Present glossarys using blocks instead of lists?

Synopsis

```
<xsl:param name="glossary.as.blocks" select="0"></xsl:param>
```

Description

If non-zero, `glossary`s will be formatted as blocks.

If you have long `glossterm`s, proper list markup in the FO case may produce unattractive lists. By setting this parameter, you can force the stylesheets to produce block markup instead of proper lists.

You can override this setting with a processing instruction as the child of `glossary`: `<?dbfo glossary-presentation="blocks"?>` or `<?dbfo glossary-presentation="list"?>`

Name

glosslist.as.blocks — Use blocks for glosslists?

Synopsis

```
<xsl:param name="glosslist.as.blocks" select="0"></xsl:param>
```

Description

See *`glossary.as.blocks`*.

Name

glossentry.show.acronym — Display `glossentry` acronyms?

Synopsis

```
<xsl:param name="glossentry.show.acronym">no</xsl:param>
```

Description

A setting of “yes” means they should be displayed; “no” means they shouldn't. If “primary” is used, then they are shown as the primary text for the entry.

Note

This setting controls both `acronym` and `abbrev` elements in the `glossentry`.

Name

glossary.sort — Sort glossentry elements?

Synopsis

```
<xsl:param name="glossary.sort" select="0"></xsl:param>
```

Description

If non-zero, then the glossentry elements within a glossary, glossdiv, or glosslist are sorted on the glossterm, using the current lang setting. If zero (the default), then glossentry elements are not sorted and are presented in document order.

Miscellaneous

Name

formal.procedures — Selects formal or informal procedures

Synopsis

```
<xsl:param name="formal.procedures" select="1"></xsl:param>
```

Description

Formal procedures are numbered and always have a title.

Name

formal.title.placement — Specifies where formal object titles should occur

Synopsis

```
<xsl:param name="formal.title.placement">
figure before
example before
equation before
table before
procedure before
task before
</xsl:param>
```

Description

Specifies where formal object titles should occur. For each formal object type (`figure`, `example`, `equation`, `table`, and `procedure`) you can specify either the keyword "`before`" or "`after`".

Name

runinhead.default.title.end.punct — Default punctuation character on a run-in-head

Synopsis

```
<xsl:param name="runinhead.default.title.end.punct">.</xsl:param>
```

Description

If non-zero, For a `formalpara`, use the specified string as the separator between the title and following text. The period is the default value.

Name

runinhead.title.end.punct — Characters that count as punctuation on a run-in-head

Synopsis

```
<xsl:param name="runinhead.title.end.punct">.!?:</xsl:param>
```

Description

Specify which characters are to be counted as punctuation. These characters are checked for a match with the last character of the title. If no match is found, the

runinhead.default.title.end.punct contents are inserted. This is to avoid duplicated punctuation in the output.

Name

show.comments — Display `remark` elements?

Synopsis

```
<xsl:param name="show.comments" select="1"></xsl:param>
```

Description

If non-zero, comments will be displayed, otherwise they are suppressed. Comments here refers to the `remark` element (which was called `comment` prior to DocBook 4.0), not XML comments (<-- like this -->) which are unavailable.

Name

punct.honorific — Punctuation after an honorific in a personal name.

Synopsis

```
<xsl:param name="punct.honorific">.</xsl:param>
```

Description

This parameter specifies the punctuation that should be added after an honorific in a personal name.

Name

segmentedlist.as.table — Format segmented lists as tables?

Synopsis

```
<xsl:param name="segmentedlist.as.table" select="0"></xsl:param>
```

Description

If non-zero, `segmentedlist`s will be formatted as tables.

Name

variablelist.as.blocks — Format `variablelist`s lists as blocks?

Synopsis

```
<xsl:param name="variablelist.as.blocks" select="0"></xsl:param>
```

Description

If non-zero, `variablelist`s will be formatted as blocks.

If you have long terms, proper list markup in the FO case may produce unattractive lists. By setting this parameter, you can force the stylesheets to produce block markup instead of proper lists.

You can override this setting with a processing instruction as the child of `variablelist`: `<?dbfo list-presentation="blocks"?>` or `<?dbfo list-presentation="list"?>`.

When using `list-presentation="list"`, you can also control the amount of space used for the `term`s with the `<?dbfo term-width=".25in"?>` processing instruction, the `termlength` attribute on `variablelist`, or allow the stylesheets to attempt to calculate the amount of space to leave based on the number of letters in the longest term.

```
  <variablelist>
        <?dbfo list-presentation="list"?>
        <?dbfo term-width="1.5in"?>
        <?dbhtml list-presentation="table"?>
        <?dbhtml term-width="1.5in"?>
        <varlistentry>
          <term>list</term>
          <listitem>
                <para>
                  Formatted as a list even if variablelist.as.blocks is set to 1.
                </para>
          </listitem>
        </varlistentry>
  </variablelist>
```

Name

blockquote.properties — To set the style for block quotations.

Synopsis

```
<xsl:attribute-set name="blockquote.properties">
<xsl:attribute name="margin-left">0.5in</xsl:attribute>
<xsl:attribute name="margin-right">0.5in</xsl:attribute>
<xsl:attribute name="space-after.minimum">0.5em</xsl:attribute>
<xsl:attribute name="space-after.optimum">1em</xsl:attribute>
<xsl:attribute name="space-after.maximum">2em</xsl:attribute>
</xsl:attribute-set>
```

Description

The *`blockquote.properties`* attribute set specifies the formating properties of block quotations.

Name

ulink.show — Display URLs after `ulink`s?

Synopsis

```
<xsl:param name="ulink.show" select="1"></xsl:param>
```

Description

If non-zero, the URL of each `ulink` will appear after the text of the link. If the text of the link and the URL are identical, the URL is suppressed.

Name

ulink.footnotes — Generate footnotes for `ULink`s?

Synopsis

```
<xsl:param name="ulink.footnotes" select="0"></xsl:param>
```

Description

If non-zero, the URL of each `ULink` will appear as a footnote.

Name

ulink.hyphenate — Allow URLs to be automatically hyphenated

Synopsis

```
<xsl:param name="ulink.hyphenate"></xsl:param>
```

Description

If not empty, the specified character (or more generally, content) is added to URLs after every character included in the string in the *`ulink.hyphenate.chars`* parameter (default is "/"). If the character in this parameter is a Unicode soft hyphen (0x00AD) or Unicode zero-width space (0x200B), some FO processors will be able to reasonably hyphenate long URLs.

As of 28 Jan 2002, discretionary hyphens are more widely and correctly supported than zero-width spaces for this purpose.

Name

ulink.hyphenate.chars — List of characters to allow ulink URLs to be automatically hyphenated on

Synopsis

```
<xsl:param name="ulink.hyphenate.chars">/</xsl:param>
```

Description

If the *`ulink.hyphenate`* is not empty, then hyphenation of ulinks is turned on, and any character contained in this parameter is treated as an allowable hyphenation point.

The default value is "/", but the parameter could be customized to contain other URL characters, as for example:

```
<xsl:param name="ulink.hyphenate.chars">:/@&?.#</xsl:param>
```

Name

shade.verbatim — Should verbatim environments be shaded?

Synopsis

```
<xsl:param name="shade.verbatim" select="0"></xsl:param>
```

Description

In the FO stylesheet, if this parameter is non-zero then the shade.verbatim.style properties will be applied to verbatim environments.

In the HTML stylesheet, this parameter is now deprecated. Use CSS instead.

Name

shade.verbatim.style — Properties that specify the style of shaded verbatim listings

Synopsis

```
<xsl:attribute-set name="shade.verbatim.style">
```

```
  <xsl:attribute name="background-color">#E0E0E0</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties that specify the style of shaded verbatim listings. The parameters specified (the border and background color) are added to the styling of the xsl-fo output. A border might be specified as "thin black solid" for example. See xsl-fo[1]

Name

hyphenate.verbatim — Should verbatim environments be hyphenated on space characters?

Synopsis

```
<xsl:param name="hyphenate.verbatim" select="0"></xsl:param>
```

Description

If the lines of program listing are too long to fit into one line it is quite common to split them at space and indicite by hook arrow that code continues on the next line. You can turn on this behaviour for `programlisting`, `screen` and `synopsis` elements by using this parameter.

Note that you must also enable line wrapping for verbatim environments and select appropriate hyphenation character (e.g. hook arrow). This can be done using *`monospace.verbatim.properties`* attribute set:

```
<xsl:attribute-set name="monospace.verbatim.properties"
                   use-attribute-sets="verbatim.properties monospace.properties">
  <xsl:attribute name="wrap-option">wrap</xsl:attribute>
  <xsl:attribute name="hyphenation-character">&#x25BA;</xsl:attribute>
</xsl:attribute-set>
```

For a list of arrows available in Unicode see `http://www.unicode.org/charts/PDF/U2190.pdf` and `http://www.unicode.org/charts/PDF/U2900.pdf` and make sure that selected character is available in the font you are using for verbatim environments.

Name

hyphenate.verbatim.characters — List of characters after which line break can occur in listings

Synopsis

```
<xsl:param name="hyphenate.verbatim.characters"></xsl:param>
```

Description

If you enable *`hyphenate.verbatim`* line breaks are allowed only on space characters. If this is not enough for your document, you can specify list of additional characters after which line break is allowed in this parameter.

Name

use.svg — Allow SVG in the result tree?

[1] http://www.w3.org/TR/2004/WD-xsl11-20041216/#border

Synopsis

```
<xsl:param name="use.svg" select="1"></xsl:param>
```

Description

If non-zero, SVG will be considered an acceptable image format. SVG is passed through to the result tree, so correct rendering of the resulting diagram depends on the formatter (FO processor or web browser) that is used to process the output from the stylesheet.

Name

use.role.as.xrefstyle — Use `role` attribute for `xrefstyle` on `xref`?

Synopsis

```
<xsl:param name="use.role.as.xrefstyle" select="1"></xsl:param>
```

Description

If non-zero, the `role` attribute on `xref` will be used to select the cross reference style. The DocBook Technical Committee[1] recently added an `xrefstyle` attribute for this purpose. If the `xrefstyle` attribute is present, `role` will be ignored, regardless of this setting.

Until an official DocBook release that includes the new attribute, this flag allows `role` to serve that purpose.

Example

The following small stylesheet shows how to configure the stylesheets to make use of the cross reference style:

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

<xsl:import href="../xsl/html/docbook.xsl"/>

<xsl:output method="html"/>

<xsl:param name="local.l10n.xml" select="document('')"/>
<l:i18n xmlns:l="http://docbook.sourceforge.net/xmlns/l10n/1.0">
  <l:l10n xmlns:l="http://docbook.sourceforge.net/xmlns/l10n/1.0" language="en">
   <l:context name="xref">
      <l:template name="chapter" style="title" text="Chapter %n, %t"/>
      <l:template name="chapter" text="Chapter %n"/>
    </l:context>
  </l:l10n>
</l:i18n>

</xsl:stylesheet>
```

With this stylesheet, the cross references in the following document:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
                  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book id="book"><title>Book</title>

<preface>
<title>Preface</title>
```

[1] http://www.oasis-open.org/docbook/

```
<para>Normal: <xref linkend="ch1"/>.</para>
<para>Title: <xref xrefstyle="title" linkend="ch1"/>.</para>

</preface>

<chapter id="ch1">
<title>First Chapter</title>

<para>Irrelevant.</para>

</chapter>
</book>
```

will appear as:

Normal: Chapter 1.

Title: Chapter 1, *First Chapter*.

Name

menuchoice.separator — Separator between items of a `menuchoice` other than `guimenuitem` and `guisubmenu`

Synopsis

```
<xsl:param name="menuchoice.separator">+</xsl:param>
```

Description

Separator used to connect items of a `menuchoice` other than `guimenuitem` and `guisubmenu`. The latter elements are linked with *`menuchoice.menu.separator`*.

Name

menuchoice.menu.separator — Separator between items of a `menuchoice` with `guimenuitem` or `guisubmenu`

Synopsis

```
<xsl:param name="menuchoice.menu.separator">   </xsl:param>
```

Description

Separator used to connect items of a `menuchoice` with `guimenuitem` or `guisubmenu`. Other elements are linked with *`menuchoice.separator`*.

The default value is →, which is the → (right arrow) character entity. The current FOP (0.20.5) requires setting the font-family explicitly.

The default value also includes spaces around the arrow, which will allow a line to break. Replace the spaces with   (nonbreaking space) if you don't want those spaces to break.

Name

default.float.class — Specifies the default float class

Synopsis

```
<xsl:param name="default.float.class">
  <xsl:choose>
```

```
    <xsl:when test="contains($stylesheet.result.type,'html')">left</xsl:when>
    <xsl:otherwise>before</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

Selects the direction in which a float should be placed. for xsl-fo this is before, for html it is left. For Western texts, the before direction is the top of the page.

Name

footnote.number.format — Identifies the format used for footnote numbers

Synopsis

```
<xsl:param name="footnote.number.format">1</xsl:param>
```

Description

The *`footnote.number.format`* specifies the format to use for footnote numeration (1, i, I, a, or A).

Name

table.footnote.number.format — Identifies the format used for footnote numbers in tables

Synopsis

```
<xsl:param name="table.footnote.number.format">a</xsl:param>
```

Description

The *`table.footnote.number.format`* specifies the format to use for footnote numeration (1, i, I, a, or A) in tables.

Name

footnote.number.symbols — Special characters to use as footnote markers

Synopsis

```
<xsl:param name="footnote.number.symbols"></xsl:param>
```

Description

If *`footnote.number.symbols`* is not the empty string, footnotes will use the characters it contains as footnote symbols. For example, “*†‡◊✠” will identify footnotes with “*”, “†”, “‡”, “◊”, and “✠”. If there are more footnotes than symbols, the stylesheets will fall back to numbered footnotes using *`footnote.number.format`*.

The use of symbols for footnotes depends on the ability of your processor (or browser) to render the symbols you select. Not all systems are capable of displaying the full range of Unicode characters. If the quoted characters in the preceding paragraph are not displayed properly, that's a good indicator that you may have trouble using those symbols for footnotes.

Name

table.footnote.number.symbols — Special characters to use a footnote markers in tables

Synopsis

```
<xsl:param name="table.footnote.number.symbols"></xsl:param>
```

Description

If *table.footnote.number.symbols* is not the empty string, table footnotes will use the characters it contains as footnote symbols. For example, “*†‡◊✠” will identify footnotes with “*”, “†”, “‡”, “◊”, and “✠”. If there are more footnotes than symbols, the stylesheets will fall back to numbered footnotes using *table.footnote.number.format*.

The use of symbols for footnotes depends on the ability of your processor (or browser) to render the symbols you select. Not all systems are capable of displaying the full range of Unicode characters. If the quoted characters in the preceding paragraph are not displayed properly, that's a good indicator that you may have trouble using those symbols for footnotes.

Name

footnote.properties — Properties applied to each footnote body

Synopsis

```
<xsl:attribute-set name="footnote.properties">
  <xsl:attribute name="font-family"><xsl:value-of \
select="$body.fontset"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="font-size"><xsl:value-of \
select="$footnote.font.size"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="font-weight">normal</xsl:attribute>
  <xsl:attribute name="font-style">normal</xsl:attribute>
  <xsl:attribute name="text-align"><xsl:value-of \
select="$alignment"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="start-indent">0pt</xsl:attribute>
  <xsl:attribute name="text-indent">0pt</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is applied to the footnote-block for each footnote. It can be used to set the font-size, font-family, and other inheritable properties that will be applied to all footnotes.

Name

table.footnote.properties — Properties applied to each table footnote body

Synopsis

```
<xsl:attribute-set name="table.footnote.properties">
  <xsl:attribute name="font-family"><xsl:value-of \
select="$body.fontset"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="font-size"><xsl:value-of \
select="$footnote.font.size"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="font-weight">normal</xsl:attribute>
  <xsl:attribute name="font-style">normal</xsl:attribute>
  <xsl:attribute name="space-before">2pt</xsl:attribute>
  <xsl:attribute name="text-align"><xsl:value-of \
select="$alignment"></xsl:value-of></xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is applied to the footnote-block for each table footnote. It can be used to set the font-size, font-family, and other inheritable properties that will be applied to all table footnotes.

Name

footnote.mark.properties — Properties applied to each footnote mark

Synopsis

```
<xsl:attribute-set name="footnote.mark.properties">
  <xsl:attribute name="font-size">75%</xsl:attribute>
  <xsl:attribute name="font-weight">normal</xsl:attribute>
  <xsl:attribute name="font-style">normal</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is applied to the footnote mark used for each footnote. It should contain only inline properties.

The property to make the mark a superscript is contained in the footnote template itself, because the current version of FOP reports an error if baseline-shift is used.

Name

footnote.sep.leader.properties — Properties associated with a procedure

Synopsis

```
<xsl:attribute-set name="footnote.sep.leader.properties">
  <xsl:attribute name="color">black</xsl:attribute>
  <xsl:attribute name="leader-pattern">rule</xsl:attribute>
  <xsl:attribute name="leader-length">1in</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for the rule line that separates the footnotes from the body text. These are properties applied to the fo:leader used as the separator.

If you want to do more than just set properties on the leader element, then you can customize the template named `footnote.separator` in `fo/pagesetup.xsl`.

Name

xref.with.number.and.title — Use number and title in cross references

Synopsis

```
<xsl:param name="xref.with.number.and.title" select="1"></xsl:param>
```

Description

A cross reference may include the number (for example, the number of an example or figure) and the `title` which is a required child of some targets. This parameter inserts both the relevant number as well as the title into the link.

Name

superscript.properties — Properties associated with superscripts

Synopsis

```
<xsl:attribute-set name="superscript.properties">
  <xsl:attribute name="font-size">75%</xsl:attribute>
</xsl:attribute-set>
```

Description

Specifies styling properties for superscripts.

Name

subscript.properties — Properties associated with subscripts

Synopsis

```
<xsl:attribute-set name="subscript.properties">
  <xsl:attribute name="font-size">75%</xsl:attribute>
</xsl:attribute-set>
```

Description

Specifies styling properties for subscripts.

Name

pgwide.properties — Properties to make a figure or table page wide.

Synopsis

```
<xsl:attribute-set name="pgwide.properties">
  <xsl:attribute name="start-indent">0pt</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is used to set the properties that make a figure or table "page wide" in fo output. It comes into effect when an attribute `pgwide="1"` is used.

By default, it sets *`start-indent`* to `0pt`. In a stylesheet that sets the parameter *`body.start.indent`* to a non-zero value in order to indent body text, this attribute set can be used to outdent pgwide figures to the left margin.

If a document uses a multi-column page layout, then this attribute set could try setting *`span`* to a value of `all`. However, this may not work with some processors because a span property must be on an fo:block that is a direct child of fo:flow. It may work in some processors anyway.

Name

highlight.source — Should the content of `programlisting` be syntactically highlighted?

Synopsis

```
<xsl:param name="highlight.source" select="0"></xsl:param>
```

Description

When this parameter is non-zero, the stylesheets will try to do syntax highlighting of the content of the `programlisting` element. The highlighting is done by the XSLTHL extension module. This is an external Java library which is not part of the DocBook XSL distribution.

In order to use this extension, you must add `xslthl.jar` to your Java classpath. You can download this software from the XSLT syntax highlighting project[1] at SourceForge.

The configuration of syntax highlighting is stored in `highlighting/xslthl-config.xml`. The Java property `xslthl.config` must point to this file (using URL syntax).

This extension is known to work with Saxon 6.5.x. Here is an example of a modified Saxon command:

```
java -cp c:\batch\;…;c:\path\to\xslthl.jar \
-Dxslthl.config=file:///c:/docbook-xsl/highlighting/xslthl-config.xml … \
com.icl.saxon.StyleSheet …
```

You can specify the language for each programlisting by using the `language` attribute. The *`highlighting.default.language`* parameter can be used for specifying the language to be used for programlistings without a `language` attribute.

Name

highlight.default.language — Default language of programlisting

Synopsis

```
<xsl:param name="highlight.default.language"></xsl:param>
```

Description

This language is used when there is no language attribute on programlisting.

Name

email.delimiters.enabled — Generate delimiters around email addresses?

Synopsis

```
<xsl:param name="email.delimiters.enabled" select="1"></xsl:param>
```

Description

If non-zero, delimiters [1] are generated around e-mail addresses (the output of the `email` element).

Name

section.container.element — Select XSL-FO element name to contain sections

Synopsis

```
<xsl:param name="section.container.element">block</xsl:param>
```

Description

Selects the element name for outer container of each section. The choices are `block` (default) or `wrapper`. The `fo:` namespace prefix is added by the stylesheet to form the full element name.

[1] http://sourceforge.net/projects/xslthl

[1] For delimiters, the stylesheets are currently hard-coded to output angle brackets.

This element receives the section `id` attribute and the appropriate section level attribute-set.

Changing this parameter to `wrapper` is only necessary when producing multi-column output that contains page-wide spans. Using `fo:wrapper` avoids the nesting of `fo:block` elements that prevents spans from working (the standard says a span must be on a block that is a direct child of `fo:flow`).

If set to `wrapper`, the section attribute-sets only support properties that are inheritable. That's because there is no block to apply them to. Properties such as font-family are inheritable, but properties such as border are not.

Only some XSL-FO processors need to use this parameter. The Antenna House processor, for example, will handle spans in nested blocks without changing the element name. The RenderX XEP product and FOP follow the XSL-FO standard and need to use `wrapper`.

Graphics

Name

graphic.default.extension — Default extension for graphic filenames

Synopsis

```
<xsl:param name="graphic.default.extension"></xsl:param>
```

Description

If a `graphic` or `mediaobject` includes a reference to a filename that does not include an extension, and the `format` attribute is *unspecified*, the default extension will be used.

Name

default.image.width — The default width of images

Synopsis

```
<xsl:param name="default.image.width"></xsl:param>
```

Description

If specified, this value will be used for the `width` attribute on images that do not specify any viewport dimensions[1].

Name

preferred.mediaobject.role — Select which mediaobject to use based on this value of an object's `role` attribute.

Synopsis

```
<xsl:param name="preferred.mediaobject.role"></xsl:param>
```

Description

A mediaobject may contain several objects such as imageobjects. If the parameter *`use.role.for.mediaobject`* is non-zero, then the `role` attribute on `imageobject`s and other objects within a `mediaobject` container will be used to select which object will be used. If one of the objects has a role value that matches the *`preferred.mediaobject.role`* parameter, then it has first priority for selection. If more than one has such a role value, the first one is used.

See the *`use.role.for.mediaobject`* parameter for the sequence of selection.

Name

use.role.for.mediaobject — Use `role` attribute value for selecting which of several objects within a mediaobject to use.

Synopsis

```
<xsl:param name="use.role.for.mediaobject" select="1"></xsl:param>
```

[1] http://docbook.org/tdg/en/html/imagedata.html#viewport.area

Description

If non-zero, the `role` attribute on `imageobject`s or other objects within a `mediaobject` container will be used to select which object will be used.

The order of selection when then parameter is non-zero is:

1. If the stylesheet parameter *`preferred.mediaobject.role`* has a value, then the object whose role equals that value is selected.
2. Else if an object's role attribute has a value of `html` for HTML processing or `fo` for FO output, then the first of such objects is selected.
3. Else the first suitable object is selected.

If the value of *`use.role.for.mediaobject`* is zero, then role attributes are not considered and the first suitable object with or without a role value is used.

Name

ignore.image.scaling — Tell the stylesheets to ignore the author's image scaling attributes

Synopsis

```
<xsl:param name="ignore.image.scaling" select="0"></xsl:param>
```

Description

If non-zero, the scaling attributes on graphics and media objects are ignored.

Name

img.src.path — Path to HTML/FO image files

Synopsis

```
<xsl:param name="img.src.path"></xsl:param>
```

Description

Add a path prefix to each HTML `img` or FO `fo:external-graphic` element's `src` attribute. This path could be relative to the directory where the HTML/FO files are created, or it could be an absolute URI. The default value is empty. Be sure to include a trailing slash if needed.

This prefix is not applied to any filerefs that start with "/" or contain "//:".

Name

keep.relative.image.uris — Should image URIs be resolved against xml:base?

Synopsis

```
<xsl:param name="keep.relative.image.uris" select="0"></xsl:param>
```

Description

If non-zero, relative URIs (in, for example `fileref` attributes) will be used in the generated output. Otherwise, the URIs will be made absolute with respect to the base URI.

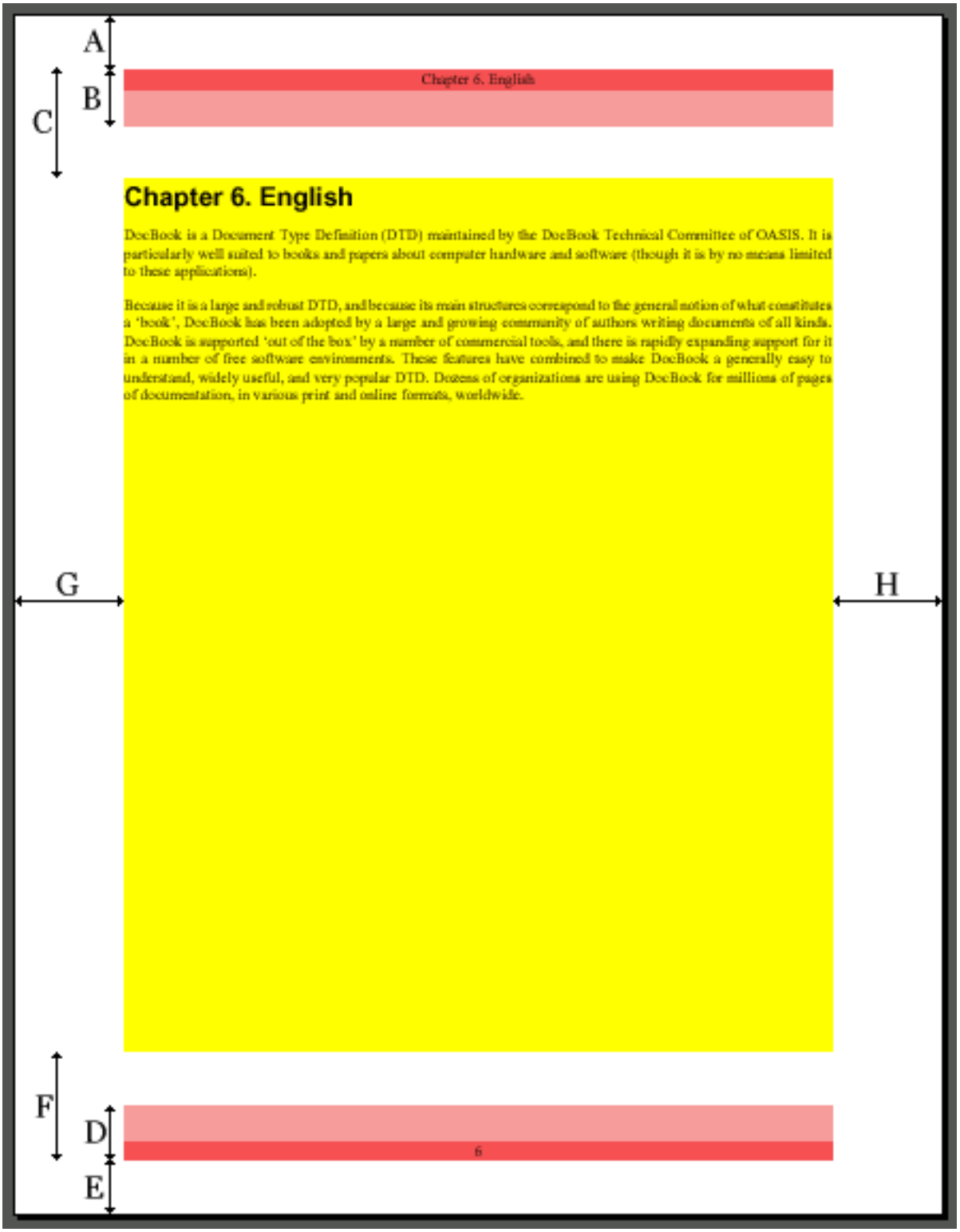
Note that the stylesheets calculate (and use) the absolute form for some purposes, this only applies to the resulting output.

Pagination and General Styles

Understanding XSL FO Margins

To make sense of the parameters in this section, it's useful to consider Figure 1, “Page Model”.

Figure 1. Page Model



First, let's consider the regions on the page.

The white region is the physical page. Its dimensions are determined by the *`page.height`* and *`page.width`* parameters.

The yellow region is the region-body. The size and placement of the region body is constrained by the dimensions labelled in the figure.

The pink region at the top of the page is the region-before. The darker area inside the region-before is the header text. In XSL, the default display alignment for a region is `before`, but the DocBook stylesheets still explicitly make it `before`. That's why the darker area is at the top.

The pink region at the bottom of the page is the region-after. The darker area is the footer text. In XSL, the default display alignment for a region is `before`, but the DocBook stylesheets explicitly make it `after`. That's why the darker area is at the bottom.

The dimensions in the figure are:

A. The page-master margin-top.

B. The region-before extent.

C. The region-body margin-top.

D. The region-after extent.

E. The page-master margin-bottom.

F. The region-body margin-bottom.

G. The sum of the page-master margin-left and the region-body margin-left. In DocBook, the region-body margin-left is zero by default, so this is simply the page-master region-left.

H. The sum of the page-master margin-right and the region-body margin-right. In DocBook, the region-body margin-right is zero by default, so this is simply the page-master region-left.

Name

page.height — The height of the physical page

Synopsis

```
<xsl:param name="page.height">
  <xsl:choose>
    <xsl:when test="$page.orientation = 'portrait'">
      <xsl:value-of select="$page.height.portrait"></xsl:value-of>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$page.width.portrait"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

The page height is generally calculated from the `paper.type` and `page.orientation`.

Name

page.height.portrait — Specify the physical size of the long edge of the page

Synopsis

```
<xsl:param name="page.height.portrait">
  <xsl:choose>
    <xsl:when test="$paper.type = 'A4landscape'">210mm</xsl:when>
    <xsl:when test="$paper.type = 'USletter'">11in</xsl:when>
    <xsl:when test="$paper.type = 'USlandscape'">8.5in</xsl:when>
    <xsl:when test="$paper.type = '4A0'">2378mm</xsl:when>
    <xsl:when test="$paper.type = '2A0'">1682mm</xsl:when>
    <xsl:when test="$paper.type = 'A0'">1189mm</xsl:when>
    <xsl:when test="$paper.type = 'A1'">841mm</xsl:when>
    <xsl:when test="$paper.type = 'A2'">594mm</xsl:when>
    <xsl:when test="$paper.type = 'A3'">420mm</xsl:when>
    <xsl:when test="$paper.type = 'A4'">297mm</xsl:when>
    <xsl:when test="$paper.type = 'A5'">210mm</xsl:when>
    <xsl:when test="$paper.type = 'A6'">148mm</xsl:when>
    <xsl:when test="$paper.type = 'A7'">105mm</xsl:when>
    <xsl:when test="$paper.type = 'A8'">74mm</xsl:when>
    <xsl:when test="$paper.type = 'A9'">52mm</xsl:when>
    <xsl:when test="$paper.type = 'A10'">37mm</xsl:when>
    <xsl:when test="$paper.type = 'B0'">1414mm</xsl:when>
    <xsl:when test="$paper.type = 'B1'">1000mm</xsl:when>
    <xsl:when test="$paper.type = 'B2'">707mm</xsl:when>
    <xsl:when test="$paper.type = 'B3'">500mm</xsl:when>
    <xsl:when test="$paper.type = 'B4'">353mm</xsl:when>
    <xsl:when test="$paper.type = 'B5'">250mm</xsl:when>
    <xsl:when test="$paper.type = 'B6'">176mm</xsl:when>
    <xsl:when test="$paper.type = 'B7'">125mm</xsl:when>
    <xsl:when test="$paper.type = 'B8'">88mm</xsl:when>
    <xsl:when test="$paper.type = 'B9'">62mm</xsl:when>
    <xsl:when test="$paper.type = 'B10'">44mm</xsl:when>
    <xsl:when test="$paper.type = 'C0'">1297mm</xsl:when>
    <xsl:when test="$paper.type = 'C1'">917mm</xsl:when>
    <xsl:when test="$paper.type = 'C2'">648mm</xsl:when>
    <xsl:when test="$paper.type = 'C3'">458mm</xsl:when>
    <xsl:when test="$paper.type = 'C4'">324mm</xsl:when>
    <xsl:when test="$paper.type = 'C5'">229mm</xsl:when>
    <xsl:when test="$paper.type = 'C6'">162mm</xsl:when>
    <xsl:when test="$paper.type = 'C7'">114mm</xsl:when>
    <xsl:when test="$paper.type = 'C8'">81mm</xsl:when>
    <xsl:when test="$paper.type = 'C9'">57mm</xsl:when>
    <xsl:when test="$paper.type = 'C10'">40mm</xsl:when>
    <xsl:otherwise>11in</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

The portrait page height is the length of the long edge of the physical page.

Name

page.margin.bottom — The bottom margin of the page

Synopsis

```
<xsl:param name="page.margin.bottom">0.5in</xsl:param>
```

Description

The bottom page margin is the distance from the bottom of the region-after to the physical bottom of the page.

Name

page.margin.inner — The inner page margin

Synopsis

```
<xsl:param name="page.margin.inner">
  <xsl:choose>
    <xsl:when test="$double.sided != 0">1.25in</xsl:when>
    <xsl:otherwise>1in</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

The inner page margin is the distance from binding edge of the page to the first column of text. In the left-to-right, top-to-bottom writing direction, this is the left margin of recto pages.

The inner and outer margins are usually the same unless the output is double-sided.

Name

page.margin.outer — The outer page margin

Synopsis

```
<xsl:param name="page.margin.outer">
  <xsl:choose>
    <xsl:when test="$double.sided != 0">0.75in</xsl:when>
    <xsl:otherwise>1in</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

The outer page margin is the distance from non-binding edge of the page to the last column of text. In the left-to-right, top-to-bottom writing direction, this is the right margin of recto pages.

The inner and outer margins are usually the same unless the output is double-sided.

Name

page.margin.top — The top margin of the page

Synopsis

```
<xsl:param name="page.margin.top">0.5in</xsl:param>
```

Description

The top page margin is the distance from the physical top of the page to the top of the region-before.

Name

page.orientation — Select the page orientation

Synopsis

```
<xsl:param name="page.orientation">portrait</xsl:param>
```

Description

Select one from portrait or landscape. In portrait orientation, the short edge is horizontal; in landscape orientation, it is vertical.

Name

page.width — The width of the physical page

Synopsis

```
<xsl:param name="page.width">
  <xsl:choose>
    <xsl:when test="$page.orientation = 'portrait'">
      <xsl:value-of select="$page.width.portrait"></xsl:value-of>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$page.height.portrait"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

The page width is generally calculated from the `paper.type` and `page.orientation`.

Name

page.width.portrait — Specify the physical size of the short edge of the page

Synopsis

```
<xsl:param name="page.width.portrait">
  <xsl:choose>
    <xsl:when test="$paper.type = 'USletter'">8.5in</xsl:when>
    <xsl:when test="$paper.type = '4A0'">1682mm</xsl:when>
    <xsl:when test="$paper.type = '2A0'">1189mm</xsl:when>
    <xsl:when test="$paper.type = 'A0'">841mm</xsl:when>
    <xsl:when test="$paper.type = 'A1'">594mm</xsl:when>
    <xsl:when test="$paper.type = 'A2'">420mm</xsl:when>
    <xsl:when test="$paper.type = 'A3'">297mm</xsl:when>
    <xsl:when test="$paper.type = 'A4'">210mm</xsl:when>
    <xsl:when test="$paper.type = 'A5'">148mm</xsl:when>
    <xsl:when test="$paper.type = 'A6'">105mm</xsl:when>
    <xsl:when test="$paper.type = 'A7'">74mm</xsl:when>
    <xsl:when test="$paper.type = 'A8'">52mm</xsl:when>
    <xsl:when test="$paper.type = 'A9'">37mm</xsl:when>
    <xsl:when test="$paper.type = 'A10'">26mm</xsl:when>
    <xsl:when test="$paper.type = 'B0'">1000mm</xsl:when>
    <xsl:when test="$paper.type = 'B1'">707mm</xsl:when>
    <xsl:when test="$paper.type = 'B2'">500mm</xsl:when>
    <xsl:when test="$paper.type = 'B3'">353mm</xsl:when>
    <xsl:when test="$paper.type = 'B4'">250mm</xsl:when>
    <xsl:when test="$paper.type = 'B5'">176mm</xsl:when>
    <xsl:when test="$paper.type = 'B6'">125mm</xsl:when>
    <xsl:when test="$paper.type = 'B7'">88mm</xsl:when>
    <xsl:when test="$paper.type = 'B8'">62mm</xsl:when>
    <xsl:when test="$paper.type = 'B9'">44mm</xsl:when>
    <xsl:when test="$paper.type = 'B10'">31mm</xsl:when>
    <xsl:when test="$paper.type = 'C0'">917mm</xsl:when>
    <xsl:when test="$paper.type = 'C1'">648mm</xsl:when>
    <xsl:when test="$paper.type = 'C2'">458mm</xsl:when>
    <xsl:when test="$paper.type = 'C3'">324mm</xsl:when>
    <xsl:when test="$paper.type = 'C4'">229mm</xsl:when>
    <xsl:when test="$paper.type = 'C5'">162mm</xsl:when>
    <xsl:when test="$paper.type = 'C6'">114mm</xsl:when>
    <xsl:when test="$paper.type = 'C7'">81mm</xsl:when>
    <xsl:when test="$paper.type = 'C8'">57mm</xsl:when>
    <xsl:when test="$paper.type = 'C9'">40mm</xsl:when>
    <xsl:when test="$paper.type = 'C10'">28mm</xsl:when>
    <xsl:otherwise>8.5in</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

The portrait page width is the length of the short edge of the physical page.

Name

paper.type — Select the paper type

Synopsis

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

Description

The paper type is a convenient way to specify the paper size. The list of known paper sizes includes USletter and most of the A, B, and C sizes. See *`page.width.portrait`*, for example.

Name

double.sided — Is the document to be printed double sided?

Synopsis

```
<xsl:param name="double.sided" select="0"></xsl:param>
```

Description

Double-sided documents are printed with a slightly wider margin on the binding edge of the page.

FIXME: The current set of parameters does not take writing direction into account.

Name

body.margin.bottom — The bottom margin of the body text

Synopsis

```
<xsl:param name="body.margin.bottom">0.5in</xsl:param>
```

Description

The body bottom margin is the distance from the last line of text in the page body to the bottom of the region-after.

Name

body.margin.top — To specify the size of the top margin of a page

Synopsis

```
<xsl:param name="body.margin.top">0.5in</xsl:param>
```

Description

The body top margin is the distance from the top of the region-before to the first line of text in the page body.

Name

body.start.indent — The start-indent for the body text

Synopsis

```
<xsl:param name="body.start.indent">
  <xsl:choose>
    <xsl:when test="$fop.extensions != 0">0pt</xsl:when>
    <xsl:when test="$passivetex.extensions != 0">0pt</xsl:when>
    <xsl:otherwise>4pc</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

This parameter provides one means of indenting the body text relative to the left page margin. It is used in place of the *`title.margin.left`* for all XSL-FO processors except FOP. It enables support for side floats to appear in the indented margin area.

This start-indent property is added to the fo:flow for certain page sequences. Which page-sequences it is applied to is determined by the template named `set.flow.properties`. By default, that template adds it to the flow for page-sequences using the “body” master-reference, as well as appendixes and prefaces.

If this parameter is used, section titles should have a start-indent value of 0pt if they are to be outdented relative to the body text.

If you are using FOP, then set this parameter to a zero width value and set the *`title.margin.left`* parameter to the negative value of the desired indent.

See also *`body.end.indent`* and *`title.margin.left`*.

Name

body.end.indent — The end-indent for the body text

Synopsis

```
<xsl:param name="body.end.indent">0pt</xsl:param>
```

Description

This end-indent property is added to the fo:flow for certain page sequences. Which page-sequences it is applied to is determined by the template named `set.flow.properties`. By default, that template adds it to the flow for page-sequences using the “body” master-reference, as well as appendixes and prefaces.

See also *`body.start.indent`*.

Name

alignment — Specify the default text alignment

Synopsis

```
<xsl:param name="alignment">justify</xsl:param>
```

Description

The default text alignment is used for most body text.

Name

hyphenate — Specify hyphenation behavior

Synopsis

```
<xsl:param name="hyphenate">true</xsl:param>
```

Description

If true, words may be hyphenated. Otherwise, they may not.

Name

line-height — Specify the line-height property

Synopsis

```
<xsl:param name="line-height">normal</xsl:param>
```

Description

Sets the line-height property.

Name

column.count.back — Number of columns on back matter pages

Synopsis

```
<xsl:param name="column.count.back" select="1"></xsl:param>
```

Description

Number of columns on back matter (appendix, glossary, etc.) pages.

Name

column.count.body — Number of columns on body pages

Synopsis

```
<xsl:param name="column.count.body" select="1"></xsl:param>
```

Description

Number of columns on body pages.

Name

column.count.front — Number of columns on front matter pages

Synopsis

```
<xsl:param name="column.count.front" select="1"></xsl:param>
```

Description

Number of columns on front matter (dedication, preface, etc.) pages.

Name

column.count.index — Number of columns on index pages

Synopsis

```
<xsl:param name="column.count.index">2</xsl:param>
```

Description

Number of columns on index pages.

Name

column.count.lot — Number of columns on a 'List-of-Titles' page

Synopsis

```
<xsl:param name="column.count.lot" select="1"></xsl:param>
```

Description

Number of columns on a page sequence containing the Table of Contents, List of Figures, etc.

Name

column.count.titlepage — Number of columns on a title page

Synopsis

```
<xsl:param name="column.count.titlepage" select="1"></xsl:param>
```

Description

Number of columns on a title page

Name

column.gap.back — Gap between columns in back matter

Synopsis

```
<xsl:param name="column.gap.back">12pt</xsl:param>
```

Description

Specifies the gap between columns in back matter (if `column.count.back` is greater than one).

Name

column.gap.body — Gap between columns in the body

Synopsis

```
<xsl:param name="column.gap.body">12pt</xsl:param>
```

Description

Specifies the gap between columns in body matter (if `column.count.body` is greater than one).

Name

column.gap.front — Gap between columns in the front matter

Synopsis

```
<xsl:param name="column.gap.front">12pt</xsl:param>
```

Description

Specifies the gap between columns in front matter (if *column.count.front* is greater than one).

Name

column.gap.index — Gap between columns in the index

Synopsis

```
<xsl:param name="column.gap.index">12pt</xsl:param>
```

Description

Specifies the gap between columns in indexes (if *column.count.index* is greater than one).

Name

column.gap.lot — Gap between columns on a 'List-of-Titles' page

Synopsis

```
<xsl:param name="column.gap.lot">12pt</xsl:param>
```

Description

Specifies the gap between columns on 'List-of-Titles' pages (if *column.count.lot* is greater than one).

Name

column.gap.titlepage — Gap between columns on title pages

Synopsis

```
<xsl:param name="column.gap.titlepage">12pt</xsl:param>
```

Description

Specifies the gap between columns on title pages (if *column.count.titlepage* is greater than one).

Name

region.after.extent — Specifies the height of the footer.

Synopsis

```
<xsl:param name="region.after.extent">0.4in</xsl:param>
```

Description

The region after extent is the height of the area where footers are printed.

Name

region.before.extent — Specifies the height of the header

Synopsis

```
<xsl:param name="region.before.extent">0.4in</xsl:param>
```

Description

The region before extent is the height of the area where headers are printed.

Name

default.units — Default units for an unqualified dimension

Synopsis

```
<xsl:param name="default.units">pt</xsl:param>
```

Description

If an unqualified dimension is encountered (for example, in a graphic width), the *`default-units`* will be used for the units. Unqualified dimensions are not allowed in XSL Formatting Objects.

Name

normal.para.spacing — What space do you want between normal paragraphs

Synopsis

```
<xsl:attribute-set name="normal.para.spacing">
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
</xsl:attribute-set>
```

Description

Specify the spacing required between normal paragraphs

Name

body.font.master — Specifies the default point size for body text

Synopsis

```
<xsl:param name="body.font.master">10</xsl:param>
```

Description

The body font size is specified in two parameters (`body.font.master` and `body.font.size`) so that math can be performed on the font size by XSLT.

Name

body.font.size — Specifies the default font size for body text

Synopsis

```
<xsl:param name="body.font.size">
 <xsl:value-of select="$body.font.master"></xsl:value-of><xsl:text>pt</xsl:text>
</xsl:param>
```

Description

The body font size is specified in two parameters (`body.font.master` and `body.font.size`) so that math can be performed on the font size by XSLT.

Name

footnote.font.size — The font size for footnotes

Synopsis

```
<xsl:param name="footnote.font.size">
 <xsl:value-of select="$body.font.master * 0.8"></xsl:value-of><xsl:text>pt</xsl:text>
</xsl:param>
```

Description

The footnote font size is used for...footnotes!

Name

title.margin.left — Adjust the left margin for titles

Synopsis

```
<xsl:param name="title.margin.left">
  <xsl:choose>
    <xsl:when test="$fop.extensions != 0">-4pc</xsl:when>
    <xsl:when test="$passivetex.extensions != 0">0pt</xsl:when>
    <xsl:otherwise>0pt</xsl:otherwise>
  </xsl:choose>
</xsl:param>
```

Description

This parameter provides one means of adjusting the left margin for titles. The left margin of the body region is calculated to include this space, and titles are outdented to the left by this amount, effectively leaving titles at the left margin and the body text indented. Currently this method is only used for FOP because it cannot properly use the *`body.start.indent`* parameter. the relative

The default value for FOP is -4pc, which means the body text is indented 4 picas relative to the titles. The default value for other processors is 0pt, and the body indent is provided by the *`body.start.indent`* parameter.

If you set the value to zero, be sure to still include a unit indicator such as `0pt`, or the FO processor will report errors.

This parameter must be set to 0pt if the *`passivetex.extensions`* parameter is nonzero because PassiveTeX cannot handle the math expression with negative values used to calculate the indents.

Name

draft.mode — Select draft mode

Synopsis

```
<xsl:param name="draft.mode">maybe</xsl:param>
```

Description

Selects draft mode. If *`draft.mode`* is “`yes`”, the entire document will be treated as a draft. If it is “`no`”, the entire document will be treated as a final copy. If it is “`maybe`”, individual sections will be treated as draft or final independently, depending on how their `status` attribute is set.

Name

draft.watermark.image — The URI of the image to be used for draft watermarks

Synopsis

```
<xsl:param \
name="draft.watermark.image">http://docbook.sourceforge.net/release/images/draft.png</xsl:param>
```

Description

The image to be used for draft watermarks.

Name

headers.on.blank.pages — Put headers on blank pages?

Synopsis

```
<xsl:param name="headers.on.blank.pages" select="1"></xsl:param>
```

Description

If non-zero, headers will be placed on blank pages.

Name

footers.on.blank.pages — Put footers on blank pages?

Synopsis

```
<xsl:param name="footers.on.blank.pages" select="1"></xsl:param>
```

Description

If non-zero, footers will be placed on blank pages.

Name

header.rule — Rule under headers?

Synopsis

```
<xsl:param name="header.rule" select="1"></xsl:param>
```

Description

If non-zero, a rule will be drawn below the page headers.

Name

footer.rule — Rule over footers?

Synopsis

```
<xsl:param name="footer.rule" select="1"></xsl:param>
```

Description

If non-zero, a rule will be drawn above the page footers.

Name

header.column.widths — Specify relative widths of header areas

Synopsis

```
<xsl:param name="header.column.widths">1 1 1</xsl:param>
```

Description

Page headers in print output use a three column table to position text at the left, center, and right side of the header on the page. This parameter lets you specify the relative sizes of the three columns. The default value is "1 1 1".

The parameter value must be three numbers, separated by white space. The first number represents the relative width of the left header for single-sided output, or the inside header for double-sided output. The second number is the relative width of the center header. The third number is the relative width of the right header for single-sided output, or the outside header for double-sided output.

The numbers are used to specify the column widths for the table that makes up the header area. In the FO output, this looks like:

```
<fo:table-column column-number="1"
    column-width="proportional-column-width(1)"/>
```

The `proportional-column-width()` function computes a column width by dividing its argument by the total of the arguments for all the columns, and then multiplying the result by the width of the whole table (assuming all the column specs use the function). Its argument can be any positive integer or floating point number. Zero is an acceptable value, although some FO processors may warn about it, in which case using a very small number might be more satisfactory.

For example, the value "1 2 1" means the center header should have twice the width of the other areas. A value of "0 0 1" means the entire header area is reserved for the right (or outside) header text. Note that to keep the center area centered on the page, the left and right values must be the same. A specification like "1 2 3" means the center area is no longer centered on the page since the right area is three times the width of the left area.

Name

footer.column.widths — Specify relative widths of footer areas

Synopsis

```
<xsl:param name="footer.column.widths">1 1 1</xsl:param>
```

Description

Page footers in print output use a three column table to position text at the left, center, and right side of the footer on the page. This parameter lets you specify the relative sizes of the three columns. The default value is "1 1 1".

The parameter value must be three numbers, separated by white space. The first number represents the relative width of the left footer for single-sided output, or the inside footer for double-sided output. The second number is the relative width of the center footer. The third number is the relative width of the right footer for single-sided output, or the outside footer for double-sided output.

The numbers are used to specify the column widths for the table that makes up the footer area. In the FO output, this looks like:

```
<fo:table-column column-number="1"
    column-width="proportional-column-width(1)"/>
```

The `proportional-column-width()` function computes a column width by dividing its argument by the total of the arguments for all the columns, and then multiplying the result by the width of the whole table (assuming all the column specs use the function). Its argument can be any positive integer or floating point number. Zero is an acceptable value, although some FO processors may warn about it, in which case using a very small number might be more satisfactory.

For example, the value "1 2 1" means the center footer should have twice the width of the other areas. A value of "0 0 1" means the entire footer area is reserved for the right (or outside) footer text. Note that to keep the center area centered on the page, the left and right values must be the same. A specification like "1 2 3" means the center area is no longer centered on the page since the right area is three times the width of the left area.

Name

header.table.properties — Apply properties to the header layout table

Synopsis

```
<xsl:attribute-set name="header.table.properties">
  <xsl:attribute name="table-layout">fixed</xsl:attribute>
  <xsl:attribute name="width">100%</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties applied to the table that lays out the page header.

Name

header.table.height — Specify the minimum height of the table containing the running page headers

Synopsis

```
<xsl:param name="header.table.height">14pt</xsl:param>
```

Description

Page headers in print output use a three column table to position text at the left, center, and right side of the header on the page. This parameter lets you specify the minimum height of the single row in the table. Since this specifies only the minimum height, the table should automatically grow to fit taller content. The default value is "14pt".

Name

footer.table.properties — Apply properties to the footer layout table

Synopsis

```
<xsl:attribute-set name="footer.table.properties">
  <xsl:attribute name="table-layout">fixed</xsl:attribute>
  <xsl:attribute name="width">100%</xsl:attribute>
</xsl:attribute-set>
```

Description

Properties applied to the table that lays out the page footer.

Name

footer.table.height — Specify the minimum height of the table containing the running page footers

Synopsis

```
<xsl:param name="footer.table.height">14pt</xsl:param>
```

Description

Page footers in print output use a three column table to position text at the left, center, and right side of the footer on the page. This parameter lets you specify the minimum height of the single row in the table. Since this specifies only the minimum height, the table should automatically grow to fit taller content. The default value is "14pt".

Name

header.content.properties

Synopsis

```
<xsl:attribute-set name="header.content.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$body.fontset"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="margin-left">
    <xsl:value-of select="$title.margin.left"></xsl:value-of>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

Properties of page header content.

Name

footer.content.properties

Synopsis

```
<xsl:attribute-set name="footer.content.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$body.fontset"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="margin-left">
    <xsl:value-of select="$title.margin.left"></xsl:value-of>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

Properties of page footer content.

Name

marker.section.level — Control depth of sections shown in running headers or footers

Synopsis

```
<xsl:param name="marker.section.level">2</xsl:param>
```

Description

The *marker.section.level* parameter controls the depth of section levels that may be displayed in running headers and footers. For example, if the value is 2 (the default), then titles from `sect1` and `sect2` or equivalent `section` elements are candidates for use in running headers and footers.

Each candidate title is marked in the FO output with a `<fo:marker marker-class-name="section.head.marker">` element.

In order for such titles to appear in headers or footers, the `header.content` or `footer.content` template must be customized to retrieve the marker using an output element such as:

```
<fo:retrieve-marker retrieve-class-name="section.head.marker"
                       retrieve-position="first-including-carryover"
                       retrieve-boundary="page-sequence"/>
```

Font Families

Name

body.font.family — The default font family for body text

Synopsis

```
<xsl:param name="body.font.family">serif</xsl:param>
```

Description

The body font family is the default font used for text in the page body.

Name

dingbat.font.family — The font family for copyright, quotes, and other symbols

Synopsis

```
<xsl:param name="dingbat.font.family">serif</xsl:param>
```

Description

The dingbat font family is used for dingbats. If it is defined as the empty string, no font change is effected around dingbats.

Name

monospace.font.family — The default font family for monospace environments

Synopsis

```
<xsl:param name="monospace.font.family">monospace</xsl:param>
```

Description

The monospace font family is used for verbatim environments (program listings, screens, etc.).

Name

sans.font.family — The default sans-serif font family

Synopsis

```
<xsl:param name="sans.font.family">sans-serif</xsl:param>
```

Description

The default sans-serif font family. At the present, this isn't actually used by the stylesheets.

Name

title.font.family — The default font family for titles

Synopsis

```
<xsl:param name="title.font.family">sans-serif</xsl:param>
```

Description

The title font family is used for titles (chapter, section, figure, etc.)

Name

symbol.font.family — The font families to be searched for symbols outside of the body font

Synopsis

```
<xsl:param name="symbol.font.family">Symbol,ZapfDingbats</xsl:param>
```

Description

A typical body or title font does not contain all the character glyphs that DocBook supports. This parameter specifies additional fonts that should be searched for special characters not in the normal font. These symbol font names are automatically appended to the body or title font family name when fonts are specified in a `font-family` property in the FO output.

The symbol font names should be entered as a comma-separated list. The default value is `Symbol,ZapfDingbats`.

Property Sets

Name

formal.object.properties — Properties associated with a formal object such as a figure, or other component that has a title

Synopsis

```
<xsl:attribute-set name="formal.object.properties">
  <xsl:attribute name="space-before.minimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">2em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">2em</xsl:attribute>
  <xsl:attribute name="keep-together.within-column">always</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for formal objects in docbook. Specify the spacing before and after the object.

Name

formal.title.properties — Style the title element of formal object such as a figure.

Synopsis

```
<xsl:attribute-set name="formal.title.properties" \
use-attribute-sets="normal.para.spacing">
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.2"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.4em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">0.6em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">0.8em</xsl:attribute>
</xsl:attribute-set>
```

Description

Specify how the title should be styled. Specify the font size and weight of the title of the formal object.

Name

informal.object.properties — Properties associated with a formal object such as a figure, or other component that has a title

Synopsis

```
<xsl:attribute-set name="informal.object.properties">
  <xsl:attribute name="space-before.minimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">2em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.5em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">2em</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for informal objects in docbook. Specify the spacing before and after the object.

Name

monospace.properties — Properties of monospaced content

Synopsis

```
<xsl:attribute-set name="monospace.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$monospace.font.family"></xsl:value-of>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

Specifies the font name for monospaced output. This property set used to set the font-size as well, but that doesn't work very well when different fonts are used (as they are in titles and paragraphs, for example).

If you want to set the font-size in a customization layer, it's probably going to be more appropriate to set font-size-adjust, if your formatter supports it.

Name

verbatim.properties — Properties associated with verbatim text

Synopsis

```
<xsl:attribute-set name="verbatim.properties">
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="space-after.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-after.optimum">1em</xsl:attribute>
  <xsl:attribute name="space-after.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="wrap-option">no-wrap</xsl:attribute>
  <xsl:attribute name="white-space-collapse">false</xsl:attribute>
  <xsl:attribute name="white-space-treatment">preserve</xsl:attribute>
  <xsl:attribute name="linefeed-treatment">preserve</xsl:attribute>
  <xsl:attribute name="text-align">start</xsl:attribute>
</xsl:attribute-set>
```

Description

This attribute set is used on all verbatim environments.

Name

monospace.verbatim.properties — What font and size do you want for monospaced content?

Synopsis

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

Description

Specify the font name and size you want for monospaced output

Name

sidebar.properties — Attribute set for sidebar properties

Synopsis

```
<xsl:attribute-set name="sidebar.properties" \
use-attribute-sets="formal.object.properties">
  <xsl:attribute name="border-style">solid</xsl:attribute>
  <xsl:attribute name="border-width">1pt</xsl:attribute>
  <xsl:attribute name="border-color">black</xsl:attribute>
  <xsl:attribute name="background-color">#DDDDDD</xsl:attribute>
  <xsl:attribute name="padding-left">12pt</xsl:attribute>
  <xsl:attribute name="padding-right">12pt</xsl:attribute>
  <xsl:attribute name="padding-top">6pt</xsl:attribute>
  <xsl:attribute name="padding-bottom">6pt</xsl:attribute>
  <xsl:attribute name="margin-left">0pt</xsl:attribute>
  <xsl:attribute name="margin-right">0pt</xsl:attribute>
<!--
  <xsl:attribute name="margin-top">6pt</xsl:attribute>
  <xsl:attribute name="margin-bottom">6pt</xsl:attribute>
-->
</xsl:attribute-set>
```

Description

The styling for sidebars.

Name

sidebar.title.properties — Attribute set for sidebar titles

Synopsis

```
<xsl:attribute-set name="sidebar.title.properties">
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="text-align">start</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for sidebars titles.

Name

sidebar.float.type — Select type of float for sidebar elements

Synopsis

```
<xsl:param name="sidebar.float.type">none</xsl:param>
```

Description

Selects the type of float for sidebar elements.

- If *`sidebar.float.type`* is “`none`”, then no float is used.
- If *`sidebar.float.type`* is “`before`”, then the float appears at the top of the page. On some processors, that may be the next page rather than the current page.
- If *`sidebar.float.type`* is “`left`” or “`start`”, then a left side float is used.

- If *sidebar.float.type* is “right” or “end”, then a right side float is used.
- If your XSL-FO processor supports floats positioned on the “inside” or “outside” of double-sided pages, then you have those two options for side floats as well.

Name

sidebar.float.width — Set the default width for sidebars

Synopsis

```
<xsl:param name="sidebar.float.width">1in</xsl:param>
```

Description

Sets the default width for sidebars when used as a side float. The width determines the degree to which the sidebar block intrudes into the text area.

If *sidebar.float.type* is “before” or “none”, then this parameter is ignored.

Name

margin.note.properties — Attribute set for margin.note properties

Synopsis

```
<xsl:attribute-set name="margin.note.properties">
  <xsl:attribute name="font-size">90%</xsl:attribute>
  <xsl:attribute name="text-align">start</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for margin notes. By default, margin notes are not implemented for any element. A stylesheet customization is needed to make use of this attribute-set.

You can use a template named “floater” to create the customization. That template can create side floats by specifying the content and characteristics as template parameters.

For example:

```
<xsl:template match="para[@role='marginnote']">
  <xsl:call-template name="floater">
    <xsl:with-param name="position">
      <xsl:value-of select="$margin.note.float.type"/>
    </xsl:with-param>
    <xsl:with-param name="width">
      <xsl:value-of select="$margin.note.width"/>
    </xsl:with-param>
    <xsl:with-param name="content">
      <xsl:apply-imports/>
    </xsl:with-param>
  </xsl:call-template>
</xsl:template>
```

Name

margin.note.title.properties — Attribute set for margin note titles

Synopsis

```
<xsl:attribute-set name="margin.note.title.properties">
```

```
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="text-align">start</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for margin note titles.

Name

margin.note.float.type — Select type of float for margin note customizations

Synopsis

```
<xsl:param name="margin.note.float.type">none</xsl:param>
```

Description

Selects the type of float for margin notes. DocBook does not define a margin note element, so this feature must be implemented as a customization of the stylesheet. See *`margin.note.properties`* for an example.

- If *`margin.note.float.type`* is "`none`", then no float is used.
- If *`margin.note.float.type`* is "`before`", then the float appears at the top of the page. On some processors, that may be the next page rather than the current page.
- If *`margin.note.float.type`* is "`left`" or "`start`", then a left side float is used.
- If *`margin.note.float.type`* is "`right`" or "`end`", then a right side float is used.
- If your XSL-FO processor supports floats positioned on the "`inside`" or "`outside`" of double-sided pages, then you have those two options for side floats as well.

Name

margin.note.width — Set the default width for margin notes

Synopsis

```
<xsl:param name="margin.note.width">1in</xsl:param>
```

Description

Sets the default width for margin notes when used as a side float. The width determines the degree to which the margin note block intrudes into the text area.

If *`margin.note.float.type`* is "`before`" or "`none`", then this parameter is ignored.

Name

component.title.properties — Properties for component titles

Synopsis

```
<xsl:attribute-set name="component.title.properties">
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.optimum"><xsl:value-of \
```

```
select="concat($body.font.master, 'pt')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="space-before.minimum"><xsl:value-of \
select="concat($body.font.master, 'pt * 0.8')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="space-before.maximum"><xsl:value-of \
select="concat($body.font.master, 'pt * 1.2')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="text-align">
    <xsl:choose>
      <xsl:when test="((parent::article | parent::articleinfo | \
parent::info/parent::article) and not(ancestor::book) and not(self::bibliography))  \
       or (parent::slides | parent::slidesinfo)">center</xsl:when>
      <xsl:otherwise>left</xsl:otherwise>
    </xsl:choose>
  </xsl:attribute>
  <xsl:attribute name="start-indent"><xsl:value-of \
select="$title.margin.left"></xsl:value-of></xsl:attribute>
</xsl:attribute-set>
```

Description

The properties common to all component titles.

Name

component.titlepage.properties — Properties for component titlepages

Synopsis

```
<xsl:attribute-set name="component.titlepage.properties">
</xsl:attribute-set>
```

Description

The properties that are applied to the outer block containing all the component title page information. Its main use is to set a `span="all"` property on the block that is a direct child of the flow.

This attribute-set is empty by default.

Name

section.title.properties — Properties for section titles

Synopsis

```
<xsl:attribute-set name="section.title.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$title.font.family"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <!-- font size is calculated dynamically by section.heading template -->
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
  <xsl:attribute name="space-before.optimum">1.0em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
  <xsl:attribute name="text-align">left</xsl:attribute>
  <xsl:attribute name="start-indent"><xsl:value-of \
select="$title.margin.left"></xsl:value-of></xsl:attribute>
</xsl:attribute-set>
```

Description

The properties common to all section titles.

Name

section.title.level1.properties — Properties for level-1 section titles

Synopsis

```
<xsl:attribute-set name="section.title.level1.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 2.0736"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-1 section titles.

Name

section.title.level2.properties — Properties for level-2 section titles

Synopsis

```
<xsl:attribute-set name="section.title.level2.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.728"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-2 section titles.

Name

section.title.level3.properties — Properties for level-3 section titles

Synopsis

```
<xsl:attribute-set name="section.title.level3.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.44"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-3 section titles.

Name

section.title.level4.properties — Properties for level-4 section titles

Synopsis

```
<xsl:attribute-set name="section.title.level4.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.2"></xsl:value-of>
    <xsl:text>pt</xsl:text>
```

```
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-4 section titles.

Name

section.title.level5.properties — Properties for level-5 section titles

Synopsis

```
<xsl:attribute-set name="section.title.level5.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-5 section titles.

Name

section.title.level6.properties — Properties for level-6 section titles

Synopsis

```
<xsl:attribute-set name="section.title.level6.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-6 section titles. This property set is actually used for all titles below level 5.

Name

section.properties — Properties for all section levels

Synopsis

```
<xsl:attribute-set name="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of all section levels, and therefore apply to the whole section. This attribute set is inherited by the more specific attribute sets such as `section.level1.properties`. The default is empty.

Name

section.level1.properties — Properties for level-1 sections

Synopsis

```
<xsl:attribute-set name="section.level1.properties" \
use-attribute-sets="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of a level-1 section, and therefore apply to the whole section. This includes `sect1` elements and `section` elements at level 1.

For example, you could start each level-1 section on a new page by using:

```
<xsl:attribute-set name="section.level1.properties">
  <xsl:attribute name="break-before">page</xsl:attribute>
</xsl:attribute-set>
```

This attribute set inherits attributes from the general `section.properties` attribute set.

Name

section.level2.properties — Properties for level-2 sections

Synopsis

```
<xsl:attribute-set name="section.level2.properties" \
use-attribute-sets="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of a level-2 section, and therefore apply to the whole section. This includes `sect2` elements and `section` elements at level 2.

For example, you could start each level-2 section on a new page by using:

```
<xsl:attribute-set name="section.level2.properties">
  <xsl:attribute name="break-before">page</xsl:attribute>
</xsl:attribute-set>
```

This attribute set inherits attributes from the general `section.properties` attribute set.

Name

section.level3.properties — Properties for level-3 sections

Synopsis

```
<xsl:attribute-set name="section.level3.properties" \
use-attribute-sets="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of a level-3 section, and therefore apply to the whole section. This includes `sect3` elements and `section` elements at level 3.

For example, you could start each level-3 section on a new page by using:

```
<xsl:attribute-set name="section.level3.properties">
  <xsl:attribute name="break-before">page</xsl:attribute>
</xsl:attribute-set>
```

This attribute set inherits attributes from the general `section.properties` attribute set.

Name

section.level4.properties — Properties for level-4 sections

Synopsis

```
<xsl:attribute-set name="section.level4.properties" \
use-attribute-sets="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of a level-4 section, and therefore apply to the whole section. This includes `sect4` elements and `section` elements at level 4.

For example, you could start each level-4 section on a new page by using:

```
<xsl:attribute-set name="section.level4.properties">
  <xsl:attribute name="break-before">page</xsl:attribute>
</xsl:attribute-set>
```

This attribute set inherits attributes from the general `section.properties` attribute set.

Name

section.level5.properties — Properties for level-5 sections

Synopsis

```
<xsl:attribute-set name="section.level5.properties" \
use-attribute-sets="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of a level-5 section, and therefore apply to the whole section. This includes `sect5` elements and `section` elements at level 5.

For example, you could start each level-5 section on a new page by using:

```
<xsl:attribute-set name="section.level5.properties">
  <xsl:attribute name="break-before">page</xsl:attribute>
</xsl:attribute-set>
```

This attribute set inherits attributes from the general `section.properties` attribute set.

Name

section.level6.properties — Properties for level-6 sections

Synopsis

```
<xsl:attribute-set name="section.level6.properties" \
use-attribute-sets="section.properties">
</xsl:attribute-set>
```

Description

The properties that apply to the containing block of a level 6 or lower section, and therefore apply to the whole section. This includes `section` elements at level 6 and lower.

For example, you could start each level-6 section on a new page by using:

```
<xsl:attribute-set name="section.level6.properties">
  <xsl:attribute name="break-before">page</xsl:attribute>
</xsl:attribute-set>
```

This attribute set inherits attributes from the general `section.properties` attribute set.

Name

figure.properties — Properties associated with a figure

Synopsis

```
<xsl:attribute-set name="figure.properties" \
use-attribute-sets="formal.object.properties"></xsl:attribute-set>
```

Description

The styling for figures.

Name

example.properties — Properties associated with a example

Synopsis

```
<xsl:attribute-set name="example.properties" \
use-attribute-sets="formal.object.properties"></xsl:attribute-set>
```

Description

The styling for examples.

Name

equation.properties — Properties associated with a equation

Synopsis

```
<xsl:attribute-set name="equation.properties" \
use-attribute-sets="formal.object.properties"></xsl:attribute-set>
```

Description

The styling for equations.

Name

table.properties — Properties associated with the block surrounding a table

Synopsis

```
<xsl:attribute-set name="table.properties" \
use-attribute-sets="formal.object.properties"></xsl:attribute-set>
```

Description

Block styling properties for tables. This parameter should really have been called `table.block.properties` or something like that, but we're leaving it to avoid backwards-compatibility problems.

See also *`table.table.properties`*.

Name

informalfigure.properties — Properties associated with an informalfigure

Synopsis

```
<xsl:attribute-set name="informalfigure.properties" \
use-attribute-sets="informal.object.properties"></xsl:attribute-set>
```

Description

The styling for informalfigures.

Name

informalexample.properties — Properties associated with an informalexample

Synopsis

```
<xsl:attribute-set name="informalexample.properties" \
use-attribute-sets="informal.object.properties"></xsl:attribute-set>
```

Description

The styling for informalexamples.

Name

informalequation.properties — Properties associated with a informalequation

Synopsis

```
<xsl:attribute-set name="informalequation.properties" \
use-attribute-sets="informal.object.properties"></xsl:attribute-set>
```

Description

The styling for informalequations.

Name

informaltable.properties — Properties associated with the block surrounding an informaltable

Synopsis

```
<xsl:attribute-set name="informaltable.properties" \
use-attribute-sets="informal.object.properties"></xsl:attribute-set>
```

Description

Block styling properties for informaltables. This parameter should really have been called `informaltable.block.properties` or something like that, but we're leaving it to avoid backwards-compatibility problems.

See also *`table.table.properties`*.

Name

procedure.properties — Properties associated with a procedure

Synopsis

```
<xsl:attribute-set name="procedure.properties" \
use-attribute-sets="formal.object.properties">
  <xsl:attribute name="keep-together.within-column">auto</xsl:attribute>
</xsl:attribute-set>
```

Description

The styling for procedures.

Name

root.properties — The properties of the fo:root element

Synopsis

```
<xsl:attribute-set name="root.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$body.fontset"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.size"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="text-align">
    <xsl:value-of select="$alignment"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="line-height">
    <xsl:value-of select="$line-height"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="font-selection-strategy">character-by-character</xsl:attribute>
  <xsl:attribute name="line-height-shift-adjustment">disregard-shifts</xsl:attribute>
</xsl:attribute-set>
```

Description

This property set is used on the `fo:root` element of an FO file. It defines a set of default, global parameters.

Name

qanda.title.properties — Properties for qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.properties">
  <xsl:attribute name="font-family">
    <xsl:value-of select="$title.font.family"></xsl:value-of>
  </xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <!-- font size is calculated dynamically by qanda.heading template -->
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.minimum">0.8em</xsl:attribute>
```

```
  <xsl:attribute name="space-before.optimum">1.0em</xsl:attribute>
  <xsl:attribute name="space-before.maximum">1.2em</xsl:attribute>
</xsl:attribute-set>
```

Description

The properties common to all qanda set titles.

Name

qanda.title.level1.properties — Properties for level-1 qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.level1.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 2.0736"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-1 qanda set titles.

Name

qanda.title.level2.properties — Properties for level-2 qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.level2.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.728"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-2 qanda set titles.

Name

qanda.title.level3.properties — Properties for level-3 qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.level3.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.44"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-3 qanda set titles.

Name

qanda.title.level4.properties — Properties for level-4 qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.level4.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master * 1.2"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-4 qanda set titles.

Name

qanda.title.level5.properties — Properties for level-5 qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.level5.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-5 qanda set titles.

Name

qanda.title.level6.properties — Properties for level-6 qanda set titles

Synopsis

```
<xsl:attribute-set name="qanda.title.level6.properties">
  <xsl:attribute name="font-size">
    <xsl:value-of select="$body.font.master"></xsl:value-of>
    <xsl:text>pt</xsl:text>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties of level-6 qanda set titles. This property set is actually used for all titles below level 5.

Name

article.appendix.title.properties — Properties for appendix titles that appear in an article

Synopsis

```
<xsl:attribute-set name="article.appendix.title.properties" \
use-attribute-sets="section.title.properties                          \
section.title.level1.properties">
  <xsl:attribute name="margin-left">
    <xsl:value-of select="$title.margin.left"></xsl:value-of>
  </xsl:attribute>
</xsl:attribute-set>
```

Description

The properties for the title of an appendix that appears inside an article. The default is to use the properties of sect1 titles.

Name

abstract.properties — Properties associated with the block surrounding an abstract

Synopsis

```
<xsl:attribute-set name="abstract.properties">
  <xsl:attribute name="start-indent">0.0in</xsl:attribute>
  <xsl:attribute name="end-indent">0.0in</xsl:attribute>
</xsl:attribute-set>
```

Description

Block styling properties for abstract.

See also *`abstract.title.properties`*.

Name

abstract.title.properties — Properties for abstract titles

Synopsis

```
<xsl:attribute-set name="abstract.title.properties">
  <xsl:attribute name="font-family"><xsl:value-of \
select="$title.fontset"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="keep-with-next.within-column">always</xsl:attribute>
  <xsl:attribute name="space-before.optimum"><xsl:value-of \
select="concat($body.font.master, 'pt')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="space-before.minimum"><xsl:value-of \
select="concat($body.font.master, 'pt * 0.8')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="space-before.maximum"><xsl:value-of \
select="concat($body.font.master, 'pt * 1.2')"></xsl:value-of></xsl:attribute>
  <xsl:attribute name="hyphenate">false</xsl:attribute>
  <xsl:attribute name="text-align">center</xsl:attribute>
</xsl:attribute-set>
```

Description

The properties for abstract titles.

See also *`abstract.properties`*.

Name

index.page.number.properties — Properties associated with index page numbers

Synopsis

```
<xsl:attribute-set name="index.page.number.properties">
</xsl:attribute-set>
```

Description

Properties associated with page numbers in indexes. Changing color to indicate the page number is a link is one possibility.

Name

revhistory.table.properties — The properties of table used for formatting revhistory

Synopsis

```
<xsl:attribute-set name="revhistory.table.properties">
</xsl:attribute-set>
```

Description

This property set defines appearance of revhistory table.

Name

revhistory.table.cell.properties — The properties of table cells used for formatting revhistory

Synopsis

```
<xsl:attribute-set name="revhistory.table.cell.properties">
</xsl:attribute-set>
```

Description

This property set defines appearance of individual cells in revhistory table.

Name

revhistory.title.properties — The properties of revhistory title

Synopsis

```
<xsl:attribute-set name="revhistory.title.properties">
</xsl:attribute-set>
```

Description

This property set defines appearance of revhistory title.

Profiling

The following parameters can be used for attribute-based profiling of your document. FIXME: add link to profiling section in Bob's book.

Name

profile.arch — Target profile for `arch` attribute

Synopsis

```
<xsl:param name="profile.arch"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.audience — Target profile for `audience` attribute

Synopsis

```
<xsl:param name="profile.audience"></xsl:param>
```

Description

Value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.condition — Target profile for `condition` attribute

Synopsis

```
<xsl:param name="profile.condition"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.conformance — Target profile for `conformance` attribute

Synopsis

```
<xsl:param name="profile.conformance"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.lang — Target profile for `lang` attribute

Synopsis

```
<xsl:param name="profile.lang"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.os — Target profile for `os` attribute

Synopsis

```
<xsl:param name="profile.os"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.revision — Target profile for `revision` attribute

Synopsis

```
<xsl:param name="profile.revision"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by `profile.separator` parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.revisionflag — Target profile for `revisionflag` attribute

Synopsis

```
<xsl:param name="profile.revisionflag"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by `profile.separator` parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.role — Target profile for `role` attribute

Synopsis

```
<xsl:param name="profile.role"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by `profile.separator` parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Warning

Note that `role` is often used for other purposes than profiling. For example it is commonly used to get emphasize in bold font:

```
<emphasis role="bold">very important</emphasis>
```

If you are using `role` for these purposes do not forget to add values like `bold` to value of this parameter. If you forgot you will get document with small pieces missing which are very hard to track.

For this reason it is not recommended to use `role` attribute for profiling. You should rather use profiling specific attributes like `userlevel`, `os`, `arch`, `condition`, etc.

Name

profile.security — Target profile for `security` attribute

Synopsis

```
<xsl:param name="profile.security"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.status — Target profile for `status` attribute

Synopsis

```
<xsl:param name="profile.status"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.userlevel — Target profile for `userlevel` attribute

Synopsis

```
<xsl:param name="profile.userlevel"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.vendor — Target profile for `vendor` attribute

Synopsis

```
<xsl:param name="profile.vendor"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.wordsize — Target profile for `wordsize` attribute

Synopsis

```
<xsl:param name="profile.wordsize"></xsl:param>
```

Description

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.attribute — Name of user-specified profiling attribute

Synopsis

```
<xsl:param name="profile.attribute"></xsl:param>
```

Description

This parameter is used in conjuction with *`profile.value`*.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.value — Target profile for user-specified attribute

Synopsis

```
<xsl:param name="profile.value"></xsl:param>
```

Description

When you are using this parameter you must also specify name of profiling attribute with parameter *`profile.attribute`*.

The value of this parameter specifies profiles which should be included in the output. You can specify multiple profiles by separating them by semicolon. You can change separator character by *`profile.separator`* parameter.

This parameter has effect only when you are using profiling stylesheets (`profile-docbook.xsl`, `profile-chunk.xsl`, …) instead of normal ones (`docbook.xsl`, `chunk.xsl`, …).

Name

profile.separator — Separator character for compound profile values

Synopsis

```
<xsl:param name="profile.separator">;</xsl:param>
```

Description

Separator character used for compound profile values. See *`profile.arch`*

Localization

Name

l10n.gentext.language — Sets the gentext language

Synopsis

```
<xsl:param name="l10n.gentext.language"></xsl:param>
```

Description

If this parameter is set to any value other than the empty string, its value will be used as the value for the language when generating text. Setting `l10n.gentext.language` overrides any settings within the document being formatted.

It's much more likely that you might want to set the `l10n.gentext.default.language` parameter.

Name

l10n.gentext.default.language — Sets the default language for generated text

Synopsis

```
<xsl:param name="l10n.gentext.default.language">en</xsl:param>
```

Description

The value of the `l10n.gentext.default.language` parameter is used as the language for generated text if no setting is provided in the source document.

Name

l10n.gentext.use.xref.language — Use the language of target when generating cross-reference text?

Synopsis

```
<xsl:param name="l10n.gentext.use.xref.language" select="0"></xsl:param>
```

Description

If non-zero, the language of the target will be used when generating cross reference text. Usually, the “current” language is used when generating text (that is, the language of the element that contains the cross-reference element). But setting this parameter allows the language of the element *pointed to* to control the generated text.

Consider the following example:

```
<para lang="en">See also <xref linkend="chap3"/>.</para>
```

Suppose that Chapter 3 happens to be written in German. If `l10n.gentext.use.xref.language` is non-zero, the resulting text will be something like this:

> See also Kapital 3.

Where the more traditional rendering would be:

See also Chapter 3.

Name

l10n.lang.value.rfc.compliant — Make value of lang attribute RFC compliant?

Synopsis

```
<xsl:param name="l10n.lang.value.rfc.compliant" select="1"></xsl:param>
```

Description

If non-zero, ensure that the values for all `lang` attributes in HTML output are RFC compliant[1]. by taking any underscore characters in any `lang` values found in source documents, and replacing them with hyphen characters in output HTML files. For example, `zh_CN` in a source document becomes `zh-CN` in the HTML output form that source.

Note

This parameter does not cause any case change in `lang` values, because RFC 1766 explicitly states that all "language tags" (as it calls them) "are to be treated as case insensitive".

[1] Section 8.1.1, Language Codes [http://www.w3.org/TR/REC-html40/struct/dirlang.html#h-8.1.1], in the HTML 4.0 Recommendation states that:

> [RFC1766] defines and explains the language codes that must be used in HTML documents.
>
> Briefly, language codes consist of a primary code and a possibly empty series of subcodes:
>
> ```
> language-code = primary-code ( "-" subcode )*
> ```
>
> And in RFC 1766, Tags for the Identification of Languages [http://www.ietf.org/rfc/rfc1766.txt], the EBNF for "language tag" is given as:
>
> ```
> Language-Tag = Primary-tag *( "-" Subtag )
> Primary-tag = 1*8ALPHA
> Subtag = 1*8ALPHA
> ```

EBNF

Name

ebnf.assignment — The EBNF production assignment operator

Synopsis

```
<xsl:param name="ebnf.assignment">
  <fo:inline font-family="{$monospace.font.family}">
    <xsl:text>::=</xsl:text>
  </fo:inline>
</xsl:param>
```

Description

The *`ebnf.assignment`* parameter determines what text is used to show “assignment” in `production`s in `productionset`s.

While “`::=`” is common, so are several other operators.

Name

ebnf.statement.terminator — Punctuation that ends an EBNF statement.

Synopsis

```
<xsl:param name="ebnf.statement.terminator"></xsl:param>
```

Description

The *`ebnf.statement.terminator`* parameter determines what text is used to terminate each `production` in `productionset`.

Some notations end each statement with a period.

Prepress

Name

crop.marks — Output crop marks?

Synopsis

```
<xsl:param name="crop.marks" select="0"></xsl:param>
```

Description

If non-zero, crop marks will be added to each page. Currently this works only with XEP if you have *`xep.extensions`* set.

Name

crop.mark.width — Width of crop marks.

Synopsis

```
<xsl:param name="crop.mark.width">0.5pt</xsl:param>
```

Description

Width of crop marks. Crop marks are controlled by *`crop.marks`* parameter.

Name

crop.mark.offset — Length of crop marks.

Synopsis

```
<xsl:param name="crop.mark.offset">24pt</xsl:param>
```

Description

Length of crop marks. Crop marks are controlled by *`crop.marks`* parameter.

Name

crop.mark.bleed — Length of invisible part of crop marks.

Synopsis

```
<xsl:param name="crop.mark.bleed">6pt</xsl:param>
```

Description

Length of invisible part of crop marks. Crop marks are controlled by *`crop.marks`* parameter.

Part III. Manpages Parameter Reference

This is reference documentation for all user-configurable parameters in the DocBook XSL "manpages" stylesheet (for generating groff/nroff output). Note that the manpages stylesheet is a customization layer of the DocBook XSL HTML stylesheet. Therefore, you can also use a number of HTML stylesheet parameters[1] to control manpages output (in addition to the manpages-specific parameters listed in this section).

[1] ../html/

Hyphenation, justification, and breaking

Name

man.hyphenate — Enable hyphenation?

Synopsis

```
<xsl:param name="man.hyphenate">0</xsl:param>
```

Description

If non-zero, hyphenation is enabled.

Note

The default value for this parameter is zero because groff is not particularly smart about how it does hyphenation; it can end up hyphenating a lot of things that you don't want hyphenated. To mitigate that, the default behavior of the stylesheets is to suppress hyphenation of computer inlines, filenames, and URLs. (You can override the default behavior by setting non-zero values for the *`man.hyphenate.urls`*, *`man.hyphenate.filenames`*, and *`man.hyphenate.computer.inlines`* parameters.) But the best way is still to just globally disable hyphenation, as the stylesheets do by default.

The only good reason to enabled hyphenation is if you have also enabled justification (which is disabled by default). The reason is that justified text can look very bad unless you also hyphenate it; to quote the “Hypenation” node from the groff info page:

> *Since the odds are not great for finding a set of words, for every output line, which fit nicely on a line without inserting excessive amounts of space between words, 'gtroff' hyphenates words so that it can justify lines without inserting too much space between words.*

So, if you set a non-zero value for the *`man.justify`* parameter (to enable justification), then you should probably also set a non-zero value for *`man.hyphenate`* (to enable hyphenation).

Name

man.hyphenate.urls — Hyphenate URLs?

Synopsis

```
<xsl:param name="man.hyphenate.urls">0</xsl:param>
```

Description

If zero (the default), hyphenation is suppressed for output of the `ulink` `url` attribute.

Note

If hyphenation is already turned off globally (that is, if *`man.hyphenate`* is zero, setting *`man.hyphenate.urls`* is not necessary.

If *man.hyphenate.urls* is non-zero, URLs will not be treated specially and are subject to hyphenation just like other words.

Note

If you are thinking about setting a non-zero value for *man.hyphenate.urls* in order to make long URLs break across lines, you'd probably be better off experimenting with setting the *man.break.after.slash* parameter first. That will cause long URLs to be broken after slashes.

Name

man.hyphenate.filenames — Hyphenate filenames?

Synopsis

```
<xsl:param name="man.hyphenate.filenames">0</xsl:param>
```

Description

If zero (the default), hyphenation is suppressed for `filename` output.

Note

If hyphenation is already turned off globally (that is, if *man.hyphenate* is zero, setting *man.hyphenate.filenames* is not necessary.

If *man.hyphenate.filenames* is non-zero, filenames will not be treated specially and are subject to hyphenation just like other words.

Note

If you are thinking about setting a non-zero value for *man.hyphenate.filenames* in order to make long filenames/pathnames break across lines, you'd probably be better off experimenting with setting the *man.break.after.slash* parameter first. That will cause long pathnames to be broken after slashes.

Name

man.hyphenate.computer.inlines — Hyphenate computer inlines?

Synopsis

```
<xsl:param name="man.hyphenate.computer.inlines">0</xsl:param>
```

Description

If zero (the default), hyphenation is suppressed for “computer inlines” such as environment variables, constants, etc. This parameter current affects output of the following elements: `classname`, `constant`, `envar`, `errorcode`, `option`, `replaceable`, `userinput`, `type`, `varname`

Note

If hyphenation is already turned off globally (that is, if *man.hyphenate* is zero, setting the *man.hyphenate.computer.inlines* is not necessary.

If *man.hyphenate.computer.inlines* is non-zero, computer inlines will not be treated specially and will be hyphenated like other words when needed.

Name

man.justify — Justify text to both right and left margins?

Synopsis

```
<xsl:param name="man.justify">0</xsl:param>
```

Description

If non-zero, text is justified to both the right and left margins (or, in roff terminology, "adjusted and filled" to both the right and left margins). If zero (the default), text is adjusted to the left margin only -- producing what is traditionally called "ragged-right" text.

Note

The default value for this parameter is zero because justified text looks good only when it is also hyphenated. Without hyphenation, excessive amounts of space often end up getting between words, in order to "pad" lines out to align on the right margin.

The problem is that groff is not particularly smart about how it does hyphenation; it can end up hyphenating a lot of things that you don't want hyphenated. So, disabling both justification and hyphenation ensures that hyphens won't get inserted where you don't want to them, and you don't end up with lines containing excessive amounts of space between words.

However, if do you decide to set a non-zero value for the *`man.justify`* parameter (to enable justification), then you should probably also set a non-zero value for *`man.hyphenate`* (to enable hyphenation).

Yes, these default settings run counter to how most existing man pages are formatted. But there are some notable exceptions, such as the `perl` man pages.

Name

man.break.after.slash — Enable line-breaking after slashes?

Synopsis

```
<xsl:param name="man.break.after.slash">0</xsl:param>
```

Description

If non-zero, line-breaking after slashes is enabled. This is mainly useful for causing long URLs or pathnames/filenames to be broken up or "wrapped" across lines (though it also has the side effect of sometimes causing relatively short URLs and pathnames to be broken up across lines too).

If zero (the default), line-breaking after slashes is disabled. In that case, strings containing slashes (for example, URLs or filenames) are not broken across lines, even if they exceed the maximum column widith.

Warning

If you set a non-zero value for this parameter, check your man-page output carefuly afterwards, in order to make sure that the setting has not introduced an excessive amount of breaking-up of URLs or pathnames. If your content contains mostly short URLs or pathnames, setting a non-zero value for *`man.break.after.slash`* will probably result in in a significant number of relatively short URLs and pathnames being broken across lines, which is probably not what you want.

Indentation

Name

man.indent.width — Specifies width used for adjusted indents

Synopsis

```
<xsl:param name="man.indent.width">4</xsl:param>
```

Description

The *man.indent.width* parameter specifies the width used for adjusted indents. The value of *man.indent.width* is used for indenting of lists, verbatims, headings, and elsewhere, depending on whether the values of certain man.indent.* boolean parameters are non-zero.

The value of *man.indent.width* should include a valid roff measurement unit (for example, n or u). The default value of 4n specifies a 4-en width; when viewed on a console, that amounts to the width of four characters. For details about roff measurment units, see the Measurements node in the groff info page.

Name

man.indent.refsect — Adjust indentation of refsect* and refsection?

Synopsis

```
<xsl:param name="man.indent.refsect" select="0"></xsl:param>
```

Description

If the value of *man.indent.refsect* is non-zero, the width of the left margin for refsect1, refsect2 and refsect3 contents and titles (and first-level, second-level, and third-level nested refsectioninstances) is adjusted by the value of the *man.indent.width* parameter. With *man.indent.width* set to its default value of 3n, the main results are that:

- contents of refsect1 are output with a left margin of three characters instead the roff default of seven or eight characters
- contents of refsect2 are displayed in console output with a left margin of six characters instead the of the roff default of seven characters
- the contents of refsect3 and nested refsection instances are adjusted accordingly.

If instead the value of *man.indent.refsect* is zero, no margin adjustment is done for refsect* output.

> **Tip**
>
> If your content is primarly comprised of refsect1 and refsect2 content (or the refsection equivalent) – with few or no refsect3 or lower nested sections , you may be able to “conserve” space in your output by setting *man.indent.refsect* to a non-zero value. Doing so will “squeeze” the left margin in such as way as to provide an additional four characters of “room” per line in refsect1 output. That extra room may be useful if, for example, you have many verbatim sections with long lines in them.

Name

man.indent.blurbs — Adjust indentation of blurbs?

Synopsis

```
<xsl:param name="man.indent.blurbs" select="1"></xsl:param>
```

Description

If the value of *man.indent.blurbs* is non-zero, the width of the left margin for `authorblurb`, `personblurb`, and `contrib` output is set to the value of the *man.indent.width* parameter (`3n` by default). If instead the value of *man.indent.blurbs* is zero, the built-in roff default width (`7.2n`) is used.

Name

man.indent.lists — Adjust indentation of lists?

Synopsis

```
<xsl:param name="man.indent.lists" select="1"></xsl:param>
```

Description

If the value of *man.indent.lists* is non-zero, the width of the left margin for list items in `itemizedlist`, `orderedlist`, `variablelist` output (and output of some other lists) is set to the value of the *man.indent.width* parameter (`4n` by default). If instead the value of *man.indent.lists* is zero, the built-in roff default width (`7.2n`) is used.

Name

man.indent.verbatims — Adjust indentation of verbatims?

Synopsis

```
<xsl:param name="man.indent.verbatims" select="1"></xsl:param>
```

Description

If the value of *man.indent.verbatims* is non-zero, the width of the left margin for output of verbatim environments (`programlisting`, `screen`, and so on) is set to the value of the *man.indent.width* parameter (`3n` by default). If instead the value of *man.indent.verbatims* is zero, the built-in roff default width (`7.2n`) is used.

Fonts

Name

man.font.funcprototype — Specifies font for funcprototype output

Synopsis

```
<xsl:param name="man.font.funcprototype">BI</xsl:param>
```

Description

The *man.font.funcprototype* parameter specifies the font for `funcprototype` output. It should be a valid roff font name, such as `BI` or `B`.

Name

man.font.funcsynopsisinfo — Specifies font for funcsynopsisinfo output

Synopsis

```
<xsl:param name="man.font.funcsynopsisinfo">B</xsl:param>
```

Description

The *man.font.funcsynopsisinfo* parameter specifies the font for `funcsynopsisinfo` output. It should be a valid roff font name, such as `B` or `I`.

Name

man.font.table.headings — Specifies font for table headings

Synopsis

```
<xsl:param name="man.font.table.headings">B</xsl:param>
```

Description

The *man.font.table.headings* parameter specifies the font for `table` headings. It should be a valid roff font, such as `B` or `I`.

Name

man.font.table.title — Specifies font for table headings

Synopsis

```
<xsl:param name="man.font.table.title">B</xsl:param>
```

Description

The *man.font.table.title* parameter specifies the font for `table` titles. It should be a valid roff font, such as `B` or `I`.

AUTHORS and COPYRIGHT sections

Name

man.authors.section.enabled — Display auto-generated AUTHORS section?

Synopsis

```
<xsl:param name="man.authors.section.enabled">1</xsl:param>
```

Description

If the value of *man.authors.section.enabled* is non-zero (the default), then an AUTHORS section is generated near the end of each man page. The output of the AUTHORS section is assembled from any `author`, `editor`, and `othercredit` metadata found in the contents of the child `info` or `refentryinfo` (if any) of the `refentry` itself, or from any `author`, `editor`, and `othercredit` metadata that may appear in `info` contents of any ancestors of the `refentry`.

If the value of *man.authors.section.enabled* is zero, the the auto-generated AUTHORS section is suppressed.

Set the value of *man.authors.section.enabled* to zero if you want to have a manually created AUTHORS section in your source, and you want it to appear in output instead of the auto-generated AUTHORS section.

Name

man.copyright.section.enabled — Display auto-generated COPYRIGHT section?

Synopsis

```
<xsl:param name="man.copyright.section.enabled">1</xsl:param>
```

Description

If the value of *man.copyright.section.enabled* is non-zero (the default), then a COPYRIGHT section is generated near the end of each man page. The output of the COPYRIGHT section is assembled from any `copyright` and `legalnotice` metadata found in the contents of the child `info` or `refentryinfo` (if any) of the `refentry` itself, or from any `copyright` and `legalnotice` metadata that may appear in `info` contents of any ancestors of the `refentry`.

If the value of *man.copyright.section.enabled* is zero, the the auto-generated COPYRIGHT section is suppressed.

Set the value of *man.copyright.section.enabled* to zero if you want to have a manually created COPYRIGHT section in your source, and you want it to appear in output instead of the auto-generated COPYRIGHT section.

Endnotes and link handling

Name

man.endnotes.list.enabled — Display endnotes list at end of man page?

Synopsis

```
<xsl:param name="man.endnotes.list.enabled">1</xsl:param>
```

Description

If the value of *man.endnotes.list.enabled* is non-zero (the default), then an endnotes list is added to the end of the output man page.

If the value of *man.endnotes.list.enabled* is zero, the list is suppressed — unless link numbering is enabled (that is, if *man.endnotes.are.numbered* is non-zero), in which case, that setting overrides the *man.endnotes.list.enabled* setting, and the endnotes list is still displayed. The reason is that inline numbering of notesources associated with endnotes only makes sense if a (numbered) list of endnotes is also generated.

Note

Leaving *man.endnotes.list.enabled* at its default (non-zero) value ensures that no “out of line” information (such as the URLs for hyperlinks and images) gets lost in your man-page output. It just gets “rearranged”.

So if you’re thinking about disabling endnotes listing by setting the value of *man.endnotes.list.enabled* to zero: Before you do so, first take some time to carefully consider the information needs and experiences of your users. The “out of line” information has value even if the presentation of it in text output is not as interactive as it may be in other output formats.

As far as the specific case of URLs: Even though the URLs displayed in text output may not be “real” (clickable) hyperlinks, many X terminals have convenience features for recognizing URLs and can, for example, present users with an options to open a URL in a browser with the user clicks on the URL is a terminal window. And short of those, users with X terminals can always manualy cut and paste the URLs into a web browser.

Also, note that various “man to html” tools, such as the widely used **man2html**[1] (`VH-Man2html`) application, automatically mark up URLs with into a@href markup during conversion — resulting in “real” hyperlinks in HTML output from those tools.

To “turn off” numbering of endnotes in the endnotes list, set *man.endnotes.are.numbered* to zero. The endnotes list will still be displayed; it will just be displayed without the numbers[2]

The default heading for the endnotes list is `NOTES`. To change that, set a non-empty value for the *man.endnotes.list.heading* parameter.

In the case of notesources that are links: Along with the URL for each link, the endnotes list includes the contents of the link. The list thus includes only non-empty[3] links. Empty links are never included, and never numbered. They are simply displayed inline, without any numbering.

[1] http://users.actrix.gen.nz/michael/vhman2html.html

[2] It can still make sense to have the list of endnotes displayed even if you have endnotes numbering turned off. In that case, your endnotes list basically becomes a “list of references” without any association with specific text in your document. This is probably the best option if you find the inline endnotes numbering obtrusive. Your users will still have access to all the “out of line” such as URLs for hyperlinks.

[3] A “non-empty” link is one that looks like this:

In addition, if there are multiple instances of links in a `refentry` that have the same URL, the URL is listed only once. The contents listed for that link in the endnotes list are the contents of the first link which has that URL.

If you disable endnotes listing, you should probably also set *`man.links.are.underlined`* to zero (to disable link underlining).

Name

man.endnotes.list.heading — Specifies an alternate name for endnotes list

Synopsis

```
<xsl:param name="man.endnotes.list.heading"></xsl:param>
```

Description

If the value of the *`man.endnotes.are.numbered`* parameter and/or the *`man.endnotes.list.enabled`* parameter is non-zero (the defaults for both are non-zero), a numbered list of endnotes is generated near the end of each man page. The default heading for the list of endnotes is the equivalent of the English word `NOTES` in the current locale. To cause an alternate heading to be displayed, set a non-empty value for the *`man.endnotes.list.heading`* parameter — for example, `REFERENCES`.

Name

man.endnotes.are.numbered — Number endnotes?

Synopsis

```
<xsl:param name="man.endnotes.are.numbered">1</xsl:param>
```

Description

If the value of *`man.endnotes.are.numbered`* is non-zero (the default), then for each non-empty[1] “notesource”:

- a number (in square brackets) is displayed inline after the rendered inline contents (if any) of the notesource
- the contents of the notesource are included in a numbered list of endnotes that is generated at the end of each man page; the number for each endnote corresponds to the inline number for the notesource with which it is associated

---

```
<ulink url="http://docbook.sf.net/snapshot/xsl/doc/manpages/">manpages</ulink>
```

an “empty link” is on that looks like this:

```
<ulink url="http://docbook.sf.net/snapshot/xsl/doc/manpages/"/>
```

[1] A “non-empty” notesource is one that looks like this:

```
<ulink url="http://docbook.sf.net/snapshot/xsl/doc/manpages/">manpages</ulink>
```

an “empty” notesource is on that looks like this:

```
<ulink url="http://docbook.sf.net/snapshot/xsl/doc/manpages/"/>
```

The default heading for the list of endnotes is `NOTES`. To output a different heading, set a value for the *`man.links.section.heading`* parameter.

Note

The endnotes list is also displayed (but without numbers) if the value of *`man.links.list.enabled`* is non-zero.

If the value of *`man.endnotes.are.numbered`* is zero, numbering of endnotess is suppressed; only inline contents (if any) of the notesource are displayed inline.

Important

If you are thinking about disabling endnote numbering by setting the value of *`man.endnotes.are.numbered`* to zero, before you do so, first take some time to carefully consider the information needs and experiences of your users. The square-bracketed numbers displayed inline after notesources may seem obstrusive and aesthetically unpleasing[2], but in a text-only output format, the numbered-notesources/endnotes-listing mechanism is the only practical way to handle this kind of content.

Also, users of “text based” browsers such as **lynx** will already be accustomed to seeing inline numbers for links. And various "man to html" applications, such as the widely used **man2html**[3] (`VH-Man2html`) application, can automatically turn URLs into "real" HTML hyperlinks in output. So leaving *`man.endnotes.are.numbered`* at its default (non-zero) value ensures that no information is lost in your man-page output. It just gets “rearranged”.

The handling of empty links is not affected by this parameter. Empty links are handled simply by displaying their URLs inline. Empty links are never auto-numbered.

If you disable endnotes numbering, you should probably also set *`man.links.are.underlined`* to zero (to disable link underlining).

Name

man.links.are.underlined — Underline links?

Synopsis

```
<xsl:param name="man.links.are.underlined">1</xsl:param>
```

Description

If the value of *`man.links.are.underlined`* is non-zero (the default), then the contents of links are rendered with an underline.

If the value of *`man.links.are.underlined`* is zero, links are displayed without any underlining.

Note

Currently, this parameter only affects output for `ulink`s.

If you set *`man.links.are.numbered`* and/or *`man.links.list.enabled`* to zero (disabled), then you should probably also set *`man.links.are.underlined`* to zero. But if

[2] As far as notesources that are links, ytou might think it would be better to just display URLs for non-empty links inline, after their content, rather than displaying square-bracketed numbers all over the place. But it's not better. In fact, it's not even practical, because many (most) URLs for links are too long to be displayed inline. They end up overflowing the right margin. You can set a non-zero value for *`man.break.after.slash`* parameter to deal with that, but it could be argued that what you end up with is at least as ugly, and definitely more obstrusive, then having short square-bracketed numbers displayed inline.

[3] http://users.actrix.gen.nz/michael/vhman2html.html

`man.links.are.numbered` is non-zero (enabled), you should probably set a non-zero value for `man.links.are.underlined` also[1].

[1] If the main purpose of underlining of links in most output formats it to indicate that the underlined text is "clickable", given that links rendered in man pages are not "real" hyperlinks that users can click on, it might seem like there is never a good reason to have link contents underlined in man output.

In fact, if you suppress the display of inline link references (by setting `man.links.are.numbered` to zero), there is no good reason to have links underlined. However, if `man.links.are.numbered` is non-zero, having links underlined may (arguably) serve a purpose: It provides "context" information about exactly what part of the text is being "annotated" by the link. Depending on how you mark up your content, that context information may or may not have value.

Lists

Name

man.segtitle.suppress — Suppress display of segtitle contents?

Synopsis

```
<xsl:param name="man.segtitle.suppress" select="0"></xsl:param>
```

Description

If the value of *`man.segtitle.suppress`* is non-zero, then display of `segtitle` contents is suppressed in output.

Character/string substitution

Name

man.charmap.enabled — Apply character map before final output?

Synopsis

```
<xsl:param name="man.charmap.enabled" select="1"></xsl:param>
```

Description

If the value of the *man.charmap.enabled* parameter is non-zero, a "character map" is used to substitute certain Unicode symbols and special characters with appropriate roff/groff equivalents, just before writing each man-page file to the filesystem. If instead the value of *man.charmap.enabled* is zero, Unicode characters are passed through "as is".

Details

For converting certain Unicode symbols and special characters in UTF-8 or UTF-16 encoded XML source to appropriate groff/roff equivalents in man-page output, the DocBook XSL Stylesheets distribution includes a roff character map[1] that is compliant with the XSLT character map[2] format as detailed in the XSLT 2.0 specification. The map contains more than 800 character mappings and can be considered the standard roff character map for the distribution.

You can use the *man.charmap.uri* parameter to specify a URI for the location for an alternate roff character map to use in place of the standard roff character map provided in the distribution.

You can also use a subset of a character map. For details, see the *man.charmap.use.subset* and *man.charmap.subset.profile* parameters.

Name

man.charmap.uri — URI for custom roff character map

Synopsis

```
<xsl:param name="man.charmap.uri"></xsl:param>
```

Description

For converting certain Unicode symbols and special characters in UTF-8 or UTF-16 encoded XML source to appropriate groff/roff equivalents in man-page output, the DocBook XSL Stylesheets distribution includes an XSLT character map[1]. That character map can be considered the standard roff character map for the distribution.

If the value of the *man.charmap.uri* parameter is non-empty, that value is used as the URI for the location for an alternate roff character map to use in place of the standard roff character map provided in the distribution.

Warning

Do not set a value for *man.charmap.uri* unless you have a custom roff character map that differs from the standard one provided in the distribution.

[1] http://docbook.sourceforge.net/snapshot/xsl/manpages/charmap.groff.xsl
[2] http://www.w3.org/TR/xslt20/#character-maps
[1] http://www.w3.org/TR/xslt20/#character-maps

Name

man.charmap.use.subset — Use subset of character map instead of full map?

Synopsis

```
<xsl:param name="man.charmap.use.subset" select="1"></xsl:param>
```

Description

If the value of the *man.charmap.use.subset* parameter is non-zero, a subset of the roff character map is used instead of the full roff character map. The profile of the subset used is specified by the *man.charmap.subset.profile* parameter.

Note

You may want to experiment with setting a non-zero value of *man.charmap.use.subset*, so that the full character map is used. Depending on which XSLT engine you run, setting a non-zero value for *man.charmap.use.subset* may significantly increase the time needed to process your documents. Or it may not. For example, if you set it and run it with xsltproc, it seems to dramatically increase processing time; on the other hand, if you set it and run it with Saxon, it does not seem to increase processing time nearly as much.

If processing time is not a important concern and/or you can tolerate the increase in processing time imposed by using the full character map, set *man.charmap.use.subset* to zero.

Details

For converting certain Unicode symbols and special characters in UTF-8 or UTF-16 encoded XML source to appropriate groff/roff equivalents in man-page output, the DocBook XSL Stylesheets distribution includes a roff character map[1] that is compliant with the XSLT character map[2] format as detailed in the XSLT 2.0 specification. The map contains more than 800 character mappings and can be considered the standard roff character map for the distribution.

Note

You can use the *man.charmap.uri* parameter to specify a URI for the location for an alternate roff character map to use in place of the standard roff character map provided in the distribution.

Because it is not terrifically efficient to use the standard 800-character character map in full -- and for most (or all) users, never necessary to use it in full -- the DocBook XSL Stylesheets support a mechanism for using, within any given character map, a subset of character mappings instead of the full set. You can use the *man.charmap.subset.profile* parameter to tune the profile of that subset to use.

Name

man.charmap.subset.profile — Profile of character map subset

Synopsis

```
<xsl:param name="man.charmap.subset.profile">
@*[local-name() = 'block'] = 'Miscellaneous Technical' or
(@*[local-name() = 'block'] = 'C1 Controls And Latin-1 Supplement (Latin-1 Supplement)' \
and
 @*[local-name() = 'class'] = 'symbols'
) or
(@*[local-name() = 'block'] = 'General Punctuation' and
```

[1] http://docbook.sourceforge.net/snapshot/xsl/manpages/charmap.groff.xsl
[2] http://www.w3.org/TR/xslt20/#character-maps

```
 (@*[local-name() = 'class'] = 'spaces' or
  @*[local-name() = 'class'] = 'dashes' or
  @*[local-name() = 'class'] = 'quotes' or
  @*[local-name() = 'class'] = 'bullets'
 )
) or
@*[local-name() = 'name'] = 'HORIZONTAL ELLIPSIS' or
@*[local-name() = 'name'] = 'WORD JOINER' or
@*[local-name() = 'name'] = 'SERVICE MARK' or
@*[local-name() = 'name'] = 'TRADE MARK SIGN' or
@*[local-name() = 'name'] = 'ZERO WIDTH NO-BREAK SPACE'
</xsl:param>
```

Description

If the value of the *man.charmap.use.subset* parameter is non-zero, The character-map subset specified by the *man.charmap.subset.profile* parameter is used instead of the full roff character map.

The value of *man.charmap.subset.profile* is a string representating an XPath expression that matches attribute names and values for `output-character` elements in the character map.

The attributes supported in the standard roff character map included in the distribution[1] are:

character
: a raw Unicode character or numeric Unicode character-entity value (either in decimal or hex); all characters have this attribute

name
: a standard full/long ISO/Unicode character name (e.g., "OHM SIGN"); all characters have this attribute

block
: a standard Unicode "block" name (e.g., "General Punctuation"); all characters have this attribute. For the full list of Unicode block names supported in the standard roff character map, see the section called “Supported Unicode block names and "class" values”.

class
: a class of characters (e.g., "spaces"). Not all characters have this attribute; currently, it is used only with certain characters within the "C1 Controls And Latin-1 Supplement" and "General Punctuation" blocks. For details, see the section called “Supported Unicode block names and "class" values”.

entity
: an ISO entity name (e.g., "ohm"); not all characters have this attribute, because not all characters have ISO entity names; for example, of the 800 or so characters in the standard roff character map included in the distribution, only around 300 have ISO entity names.

string
: a string representating an roff/groff escape-code (with "@esc@" used in place of the backslash), or a simple ASCII string; all characters in the roff character map have this attribute

The value of *man.charmap.subset.profile* is evaluated as an XPath expression at run-time to select a portion of the roff character map to use. You can tune the subset used by adding or removing parts. For example, if you need to use a wide range of mathematical operators in a document, and you want to have them converted into roff markup properly, you might add the following:

```
  @*[local-name() = 'block'] ='MathematicalOperators'
```

[1] http://docbook.sourceforge.net/snapshot/xsl/manpages/charmap.groff.xsl

That will cause a additional set of around 67 additional "math" characters to be converted into roff markup.

Note

Depending on which XSLT engine you use, either the EXSLT `dyn:evaluate` extension function (for xsltproc or Xalan) or `saxon:evaluate` extensio function (for Saxon) are used to dynamically evaluate the value of *`man.charmap.subset.profile`* at run-time. If you don't use xsltproc, Saxon, Xalan -- or some other XSLT engine that supports `dyn:evaluate` -- you must either set the value of the *`man.charmap.use.subset`* parameter to zero and process your documents using the full character map instead, or set the value of the *`man.charmap.enabled`* parameter to zero instead (so that character-map processing is disabled completely.

An alternative to using *`man.charmap.subset.profile`* is to create your own custom character map, and set the value of *`man.charmap.uri`* to the URI/filename for that. If you use a custom character map, you will probably want to include in it just the characters you want to use, and so you will most likely also want to set the value of *`man.charmap.use.subset`* to zero.

You can create a custom character map by making a copy of the standard roff character map[2] provided in the distribution, and then adding to, changing, and/or deleting from that.

Caution

If you author your DocBook XML source in UTF-8 or UTF-16 encoding and aren't sure what OSes or environments your man-page output might end up being viewed on, and not sure what version of nroff/groff those environments might have, you should be careful about what Unicode symbols and special characters you use in your source and what parts you add to the value of *`man.charmap.subset.profile`*.

Many of the escape codes used are specific to groff and using them may not provide the expected output on an OS or environment that uses nroff instead of groff.

On the other hand, if you intend for your man-page output to be viewed only on modern systems (for example, GNU/Linux systems, FreeBSD systems, or Cygwin environments) that have a good, up-to-date groff, then you can safely include a wide range of Unicode symbols and special characters in your UTF-8 or UTF-16 encoded DocBook XML source and add any of the supported Unicode block names to the value of *`man.charmap.subset.profile`*.

For other details, see the documentation for the *`man.charmap.use.subset`* parameter.

Supported Unicode block names and "class" values

Below is the full list of Unicode block names and "class" values supported in the standard roff stylesheet provided in the distribution, along with a description of which codepoints from the Unicode range corresponding to that block name or block/class combination are supported.

- C1 Controls And Latin-1 Supplement (Latin-1 Supplement)[3] (x00a0 to x00ff)

class values

- symbols
- letters

[2] http://docbook.sourceforge.net/snapshot/xsl/manpages/charmap.groff.xsl

[3] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=C1%20Controls%20and%20Latin-1%20Supplement%20(Latin-1%20Supplement)

- Latin Extended-A[4] (x0100 to x017f, partial)
- Spacing Modifier Letters[5] (x02b0 to x02ee, partial)
- Greek and Coptic[6] (x0370 to x03ff, partial)
- General Punctuation[7] (x2000 to x206f, partial)

class values

- spaces[8]
- dashes[9]
- quotes
- daggers
- bullets
- leaders
- primes

- Superscripts and Subscripts[10] (x2070 to x209f)
- Currency Symbols[11] (x20a0 to x20b1)
- Letterlike Symbols[12] (x2100 to x214b)
- Number Forms[13] (x2150 to x218f)
- Arrows[14] (x2190 to x21ff, partial)
- Mathematical Operators[15] (x2200 to x22ff, partial)
- Control Pictures[16] (x2400 to x243f)
- Enclosed Alphanumerics[17] (x2460 to x24ff)
- Geometric Shapes[18] (x25a0 to x25f7, partial)
- Miscellaneous Symbols[19] (x2600 to x26ff, partial)
- Dingbats[20] (x2700 to x27be, partial)

[4] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Latin%20Extended-A
[5] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Spacing%20Modifier%20Letters
[6] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Greek%20and%20Coptic
[7] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=General%20Punctuation
[8] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&start=8192&end=8203
[9] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&start=8208&end=8213
[10] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Superscripts%20and%20Subscripts
[11] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Currency%20Symbols
[12] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Letterlike%20Symbols
[13] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Number%20Forms
[14] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Arrows
[15] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Mathematical%20Operators
[16] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Control%20Pictures
[17] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Enclosed%20Alphanumerics
[18] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Geometric%20Shapes
[19] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Miscellaneous%20Symbols
[20] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Dingbats

- Alphabetic Presentation Forms[21] (xfb00 to xfb04 only)

Name

man.string.subst.map.local.pre — Specifies “local” string substitutions

Synopsis

```
<xsl:param name="man.string.subst.map.local.pre"></xsl:param>
```

Description

Use the *man.string.subst.map.local.pre* parameter to specify any “local” string substitutions to perform over the entire roff source for each man page *before* performing the string substitutions specified by the *man.string.subst.map* parameter.

For details about the format of this parameter, see the documentation for the *man.string.subst.map* parameter.

Name

man.string.subst.map — Specifies a set of string substitutions

Synopsis

```
<xsl:param name="man.string.subst.map">

  <!-- * remove no-break marker at beginning of line (stylesheet artifact) -->
  <ss:substitution oldstring="  " newstring=" "></ss:substitution>
  <!-- * replace U+2580 no-break marker (stylesheet-added) w/ no-break space -->
  <ss:substitution oldstring=" " newstring="\ "></ss:substitution>

  <!-- ====================================================================== -->

  <!-- * squeeze multiple newlines before a roff request  -->
  <ss:substitution oldstring="

." newstring="
."></ss:substitution>
  <!-- * remove any .sp instances that directly precede a .PP  -->
  <ss:substitution oldstring=".sp
.PP" newstring=".PP"></ss:substitution>
  <!-- * remove any .sp instances that directly follow a .PP  -->
  <ss:substitution oldstring=".PP
.sp" newstring=".PP"></ss:substitution>
  <!-- * squeeze multiple newlines after start of no-fill (verbatim) env. -->
  <ss:substitution oldstring=".nf

" newstring=".nf
"></ss:substitution>
  <!-- * squeeze multiple newlines after REstoring margin -->
  <ss:substitution oldstring=".RE

" newstring=".RE
"></ss:substitution>
  <!-- * U+2591 is a marker we add before and after every Parameter in -->
  <!-- * Funcprototype output -->
  <ss:substitution oldstring=" " newstring=" "></ss:substitution>
  <!-- * U+2592 is a marker we add for the newline before output of <sbr>; -->
  <ss:substitution oldstring=" " newstring="
"></ss:substitution>
  <!-- * -->
  <!-- * Now deal with some other characters that are added by the -->
  <!-- * stylesheets during processing. -->
```

[21] http://zvon.org/other/charSearch/PHP/search.php?searchType=103&id=Alphabetic%20Presentation%20Forms

```
  <!-- * -->
  <!-- * bullet -->
  <ss:substitution oldstring="•" newstring="\(bu"></ss:substitution>
  <!-- * left double quote -->
  <ss:substitution oldstring="“" newstring="\(lq"></ss:substitution>
  <!-- * right double quote -->
  <ss:substitution oldstring="”" newstring="\(rq"></ss:substitution>
  <!-- * left single quote -->
  <ss:substitution oldstring="‘" newstring="\(oq"></ss:substitution>
  <!-- * right single quote -->
  <ss:substitution oldstring="’" newstring="\(cq"></ss:substitution>
  <!-- * copyright sign -->
  <ss:substitution oldstring="©" newstring="\(co"></ss:substitution>
  <!-- * registered sign -->
  <ss:substitution oldstring="®" newstring="\(rg"></ss:substitution>
  <!-- * ...servicemark... -->
  <!-- * There is no groff equivalent for it. -->
  <ss:substitution oldstring=" " newstring="(SM)"></ss:substitution>
  <!-- * ...trademark... -->
  <!-- * We don't do "\(tm" because for console output, -->
  <!-- * groff just renders that as "tm"; that is: -->
  <!-- * -->
  <!-- *   Product&#x2122; -> Producttm -->
  <!-- * -->
  <!-- * So we just make it to "(TM)" instead; thus: -->
  <!-- * -->
  <!-- *   Product&#x2122; -> Product(TM) -->
  <ss:substitution oldstring="™" newstring="(TM)"></ss:substitution>

</xsl:param>
```

Description

The *man.string.subst.map* parameter contains a map that specifies a set of string substitutions to perform over the entire roff source for each man page, either just before generating final man-page output (that is, before writing man-page files to disk) or, if the value of the *man.charmap.enabled* parameter is non-zero, before applying the roff character map.

You can use *man.string.subst.map* as a “lightweight” character map to perform “essential” substitutions -- that is, substitutions that are *always* performed, even if the value of the *man.charmap.enabled* parameter is zero. For example, you can use it to replace quotation marks or other special characters that are generated by the DocBook XSL stylesheets for a particular locale setting (as opposed to those characters that are actually in source XML documents), or to replace any special characters that may be automatically generated by a particular customization of the DocBook XSL stylesheets.

Warning

Do you not change value of the *man.string.subst.map* parameter unless you are sure what you are doing. First consider adding your string-substitution mappings to either or both of the following parameters:

man.string.subst.map.local.pre
: applied before *man.string.subst.map*

man.string.subst.map.local.post
: applied after *man.string.subst.map*

By default, both of those parameters contain no string substitutions. They are intended as a means for you to specify your own local string-substitution mappings.

If you remove any of default mappings from the value of the *man.string.subst.map* parameter, you are likely to end up with broken output. And be very careful about adding anything to it; it’s used for doing string substitution over the entire roff source of each man

page – it causes target strings to be replaced in roff requests and escapes, not just in the visible contents of the page.

Contents of the substitution map

The string-substitution map contains one or more `ss:substitution` elements, each of which has two attributes:

oldstring
: string to replace

newstring
: string with which to replace `oldstring`

It may also include XML comments (that is, delimited with "`<!--`" and "`-->`").

Name

man.string.subst.map.local.post — Specifies “local” string substitutions

Synopsis

```
<xsl:param name="man.string.subst.map.local.post"></xsl:param>
```

Description

Use the *`man.string.subst.map.local.post`* parameter to specify any “local” string substitutions to perform over the entire roff source for each man page *after* performing the string substitutions specified by the *`man.string.subst.map`* parameter.

For details about the format of this parameter, see the documentation for the *`man.string.subst.map`* parameter.

Refentry metadata gathering

Name

refentry.meta.get.quietly — Suppress notes and warnings when gathering refentry metadata?

Synopsis

```
<xsl:param name="refentry.meta.get.quietly" select="0"></xsl:param>
```

Description

If zero (the default), notes and warnings about “missing” markup are generated during gathering of refentry metadata. If non-zero, the metadata is gathered “quietly” -- that is, the notes and warnings are suppressed.

Tip

If you are processing a large amount of `refentry` content, you may be able to speed up processing significantly by setting a non-zero value for *`refentry.meta.get.quietly`*.

Name

refentry.date.profile — Specifies profile for refentry "date" data

Synopsis

```
<xsl:param name="refentry.date.profile">
  (($info[//date])[last()]/date)[1]|
  (($info[//pubdate])[last()]/pubdate)[1]
</xsl:param>
```

Description

The value of *`refentry.date.profile`* is a string representing an XPath expression. It is evaluated at run-time and used only if *`refentry.date.profile.enabled`* is non-zero. Otherwise, the `refentry` metadata-gathering logic "hard coded" into the stylesheets is used.

The `man(7)` man page describes this content as "the date of the last revision". In man pages, it is the content that is usually displayed in the center footer.

Name

refentry.date.profile.enabled — Enable refentry "date" profiling?

Synopsis

```
<xsl:param name="refentry.date.profile.enabled">0</xsl:param>
```

Description

If the value of *`refentry.date.profile.enabled`* is non-zero, then during `refentry` metadata gathering, the info profile specified by the customizable *`refentry.date.profile`* parameter is used.

If instead the value of *`refentry.date.profile.enabled`* is zero (the default), then "hard coded" logic within the DocBook XSL stylesheets is used for gathering `refentry` "date" data.

If you find that the default `refentry` metadata-gathering behavior is causing incorrect "date" data to show up in your output, then consider setting a non-zero value for *`refentry.date.profile.enabled`* and adjusting the value of *`refentry.date.profile`* to cause correct data to be gathered.

Note that the terms "source" and "date" have special meanings in this context. For details, see the documentation for the *`refentry.date.profile`* parameter.

Name

refentry.manual.profile — Specifies profile for refentry "manual" data

Synopsis

```
<xsl:param name="refentry.manual.profile">
  (($info[//title])[last()]/title)[1]|
  ../title/node()
</xsl:param>
```

Description

The value of *`refentry.manual.profile`* is a string representing an XPath expression. It is evaluated at run-time and used only if *`refentry.manual.profile.enabled`* is non-zero. Otherwise, the `refentry` metadata-gathering logic "hard coded" into the stylesheets is used.

In man pages, this content is usually displayed in the middle of the header of the page. The `man(7)` man page describes this as "the title of the manual (e.g., *Linux Programmer's Manual*)". Here are some examples from existing man pages:

- *dpkg utilities* (**dpkg-name**)
- *User Contributed Perl Documentation* (**GET**)
- *GNU Development Tools* (**ld**)
- *Emperor Norton Utilities* (**ddate**)
- *Debian GNU/Linux manual* (**faked**)
- *GIMP Manual Pages* (**gimp**)
- *KDOC Documentation System* (**qt2kdoc**)

Name

refentry.manual.profile.enabled — Enable refentry "manual" profiling?

Synopsis

```
<xsl:param name="refentry.manual.profile.enabled">0</xsl:param>
```

Description

If the value of *`refentry.manual.profile.enabled`* is non-zero, then during `refentry` metadata gathering, the info profile specified by the customizable *`refentry.manual.profile`* parameter is used.

If instead the value of *`refentry.manual.profile.enabled`* is zero (the default), then "hard coded" logic within the DocBook XSL stylesheets is used for gathering `refentry` "manual" data.

If you find that the default `refentry` metadata-gathering behavior is causing incorrect "manual" data to show up in your output, then consider setting a non-zero value for *`refentry.manual.profile.enabled`* and adjusting the value of *`refentry.manual.profile`* to cause correct data to be gathered.

Note that the term "manual" has a special meanings in this context. For details, see the documentation for the *`refentry.manual.profile`* parameter.

Name

refentry.source.name.suppress — Suppress "name" part of refentry "source" contents?

Synopsis

```
<xsl:param name="refentry.source.name.suppress">0</xsl:param>
```

Description

If the value of *`refentry.source.name.suppress`* is non-zero, then during `refentry` metadata gathering, no "source name" data is added to the `refentry` "source" contents. Instead (unless *`refentry.version.suppress`* is also non-zero), only "version" data is added to the "source" contents.

If you find that the `refentry` metadata gathering mechanism is causing unwanted "source name" data to show up in your output -- for example, in the footer (or possibly header) of a man page -- then you might consider setting a non-zero value for *`refentry.source.name.suppress`*.

Note that the terms "source", "source name", and "version" have special meanings in this context. For details, see the documentation for the *`refentry.source.name.profile`* parameter.

Name

refentry.source.name.profile — Specifies profile for refentry "source name" data

Synopsis

```
<xsl:param name="refentry.source.name.profile">
  (($info[//productname])[last()]/productname)[1]|
  (($info[//corpname])[last()]/corpname)[1]|
  (($info[//corpcredit])[last()]/corpcredit)[1]|
  (($info[//corpauthor])[last()]/corpauthor)[1]|
  (($info[//orgname])[last()]/orgname)[1]|
  (($info[//publishername])[last()]/publishername)[1]
</xsl:param>
```

Description

The value of *`refentry.source.name.profile`* is a string representing an XPath expression. It is evaluated at run-time and used only if *`refentry.source.name.profile.enabled`* is non-zero. Otherwise, the `refentry` metadata-gathering logic "hard coded" into the stylesheets is used.

A "source name" is one part of a (potentially) two-part *`Name Version`* "source" field. In man pages, it is usually displayed in the left footer of the page. It typically indicates the software system or product that the item documented in the man page belongs to. The `man(7)` man page describes it as "the source of the command", and provides the following examples:

- For binaries, use something like: GNU, NET-2, SLS Distribution, MCC Distribution.
- For system calls, use the version of the kernel that you are currently looking at: Linux 0.99.11.

- For library calls, use the source of the function: GNU, BSD 4.3, Linux DLL 4.4.1.

In practice, there are many pages that simply have a Version number in the "source" field. So, it looks like what we have is a two-part field, *`Name Version`*, where:

Name
: product name (e.g., BSD) or org. name (e.g., GNU)

Version
: version number

Each part is optional. If the *`Name`* is a product name, then the *`Version`* is probably the version of the product. Or there may be no *`Name`*, in which case, if there is a *`Version`*, it is probably the version of the item itself, not the product it is part of. Or, if the *`Name`* is an organization name, then there probably will be no *`Version`*.

Name

refentry.source.name.profile.enabled — Enable refentry "source name" profiling?

Synopsis

```
<xsl:param name="refentry.source.name.profile.enabled">0</xsl:param>
```

Description

If the value of *`refentry.source.name.profile.enabled`* is non-zero, then during `refentry` metadata gathering, the info profile specified by the customizable *`refentry.source.name.profile`* parameter is used.

If instead the value of *`refentry.source.name.profile.enabled`* is zero (the default), then "hard coded" logic within the DocBook XSL stylesheets is used for gathering `refentry` "source name" data.

If you find that the default `refentry` metadata-gathering behavior is causing incorrect "source name" data to show up in your output, then consider setting a non-zero value for *`refentry.source.name.profile.enabled`* and adjusting the value of *`refentry.source.name.profile`* to cause correct data to be gathered.

Note that the terms "source" and "source name" have special meanings in this context. For details, see the documentation for the *`refentry.source.name.profile`* parameter.

Name

refentry.version.suppress — Suppress "version" part of refentry "source" contents?

Synopsis

```
<xsl:param name="refentry.version.suppress">0</xsl:param>
```

Description

If the value of *`refentry.version.suppress`* is non-zero, then during `refentry` metadata gathering, no "version" data is added to the `refentry` "source" contents. Instead (unless *`refentry.source.name.suppress`* is also non-zero), only "source name" data is added to the "source" contents.

If you find that the `refentry` metadata gathering mechanism is causing unwanted "version" data to show up in your output -- for example, in the footer (or possibly header) of a man page -- then you might consider setting a non-zero value for *`refentry.version.suppress`*.

Note that the terms "source", "source name", and "version" have special meanings in this context. For details, see the documentation for the *`refentry.source.name.profile`* parameter.

Name

refentry.version.profile — Specifies profile for refentry "version" data

Synopsis

```
<xsl:param name="refentry.version.profile">
  (($info[//productnumber])[last()]/productnumber)[1]|
  (($info[//edition])[last()]/edition)[1]|
  (($info[//releaseinfo])[last()]/releaseinfo)[1]
</xsl:param>
```

Description

The value of *`refentry.version.profile`* is a string representing an XPath expression. It is evaluated at run-time and used only if *`refentry.version.profile.enabled`* is non-zero. Otherwise, the `refentry` metadata-gathering logic "hard coded" into the stylesheets is used.

A "source.name" is one part of a (potentially) two-part *`Name Version`* "source" field. For more details, see the documentation for the *`refentry.source.name.profile`* parameter.

Name

refentry.version.profile.enabled — Enable refentry "version" profiling?

Synopsis

```
<xsl:param name="refentry.version.profile.enabled">0</xsl:param>
```

Description

If the value of *`refentry.version.profile.enabled`* is non-zero, then during `refentry` metadata gathering, the info profile specified by the customizable *`refentry.version.profile`* parameter is used.

If instead the value of *`refentry.version.profile.enabled`* is zero (the default), then "hard coded" logic within the DocBook XSL stylesheets is used for gathering `refentry` "version" data.

If you find that the default `refentry` metadata-gathering behavior is causing incorrect "version" data to show up in your output, then consider setting a non-zero value for *`refentry.version.profile.enabled`* and adjusting the value of *`refentry.version.profile`* to cause correct data to be gathered.

Note that the terms "source" and "version" have special meanings in this context. For details, see the documentation for the *`refentry.version.profile`* parameter.

Name

refentry.manual.fallback.profile — Specifies profile of "fallback" for refentry "manual" data

Synopsis

```
<xsl:param name="refentry.manual.fallback.profile">
refmeta/refmiscinfo[1]/node()</xsl:param>
```

Description

The value of *refentry.manual.fallback.profile* is a string representing an XPath expression. It is evaluated at run-time and used only if no "manual" data can be found by other means (that is, either using the refentry metadata-gathering logic "hard coded" in the stylesheets, or the value of *refentry.manual.profile*, if it is enabled).

Important

Depending on which XSLT engine you run, either the EXSLT dyn:evaluate extension function (for xsltproc or Xalan) or saxon:evaluate extension function (for Saxon) are used to dynamically evaluate the value of *refentry.manual.fallback.profile* at run-time. If you don't use xsltproc, Saxon, Xalan -- or some other XSLT engine that supports dyn:evaluate -- you must manually disable fallback processing by setting an empty value for the *refentry.manual.fallback.profile* parameter.

Name

refentry.source.fallback.profile — Specifies profile of "fallback" for refentry "source" data

Synopsis

```
<xsl:param name="refentry.source.fallback.profile">
refmeta/refmiscinfo[1]/node()</xsl:param>
```

Description

The value of *refentry.source.fallback.profile* is a string representing an XPath expression. It is evaluated at run-time and used only if no "source" data can be found by other means (that is, either using the refentry metadata-gathering logic "hard coded" in the stylesheets, or the value of the *refentry.source.name.profile* and *refentry.version.profile* parameters, if those are enabled).

Important

Depending on which XSLT engine you run, either the EXSLT dyn:evaluate extension function (for xsltproc or Xalan) or saxon:evaluate extension function (for Saxon) are used to dynamically evaluate the value of *refentry.source.fallback.profile* at run-time. If you don't use xsltproc, Saxon, Xalan -- or some other XSLT engine that supports dyn:evaluate -- you must manually disable fallback processing by setting an empty value for the *refentry.source.fallback.profile* parameter.

Page header/footer

Name

man.th.extra1.suppress — Suppress extra1 part of header/footer?

Synopsis

```
<xsl:param name="man.th.extra1.suppress">0</xsl:param>
```

Description

If the value of *`man.th.extra1.suppress`* is non-zero, then the `extra1` part of the `.TH` title line header/footer is suppressed.

The content of the `extra1` field is almost always displayed in the center footer of the page and is, universally, a date.

Name

man.th.extra2.suppress — Suppress extra2 part of header/footer?

Synopsis

```
<xsl:param name="man.th.extra2.suppress">0</xsl:param>
```

Description

If the value of *`man.th.extra2.suppress`* is non-zero, then the `extra2` part of the `.TH` title line header/footer is suppressed.

The content of the `extra2` field is usually displayed in the left footer of the page and is typically "source" data, often in the form *`Name Version`*; for example, "GTK+ 1.2" (from the `gtk-options(7)` man page).

Note

You can use the *`refentry.source.name.suppress`* and *`refentry.version.suppress`* parameters to independently suppress the *`Name`* and *`Version`* parts of the `extra2` field.

Name

man.th.extra3.suppress — Suppress extra3 part of header/footer?

Synopsis

```
<xsl:param name="man.th.extra3.suppress">0</xsl:param>
```

Description

If the value of *`man.th.extra3.suppress`* is non-zero, then the `extra3` part of the `.TH` title line header/footer is suppressed.

The content of the `extra3` field is usually displayed in the middle header of the page and is typically a "manual name"; for example, "GTK+ User's Manual" (from the `gtk-options(7)` man page).

Name

man.th.title.max.length — Maximum length of title in header/footer

Synopsis

```
<xsl:param name="man.th.title.max.length">20</xsl:param>
```

Description

Specifies the maximum permitted length of the title part of the man-page `.TH` title line header/footer. If the title exceeds the maxiumum specified, it is truncated down to the maximum permitted length.

Details

Every man page generated using the DocBook stylesheets has a title line, specified using the `TH` roff macro. Within that title line, there is always, at a minimum, a title, followed by a section value (representing a man "section" -- usually just a number).

The title and section are displayed, together, in the visible header of each page. Where in the header they are displayed depends on OS the man page is viewed on, and on what version of nroff/groff/man is used for viewing the page. But, at a minimum and across all systems, the title and section are displayed on the right-hand column of the header. On many systems -- those with a modern groff, including Linux systems -- they are displayed twice: both in the left and right columns of the header.

So if the length of the title exceeds a certain percentage of the column width in which the page is viewed, the left and right titles can end up overlapping, making them unreadable, or breaking to another line, which doesn't look particularly good.

So the stylesheets provide the *`man.th.title.max.length`* parameter as a means for truncating titles that exceed the maximum length that can be viewing properly in a page header.

The default value is reasonable but somewhat arbitrary. If you have pages with long titles, you may want to experiment with changing the value in order to achieve the correct aesthetic results.

Name

man.th.extra2.max.length — Maximum length of extra2 in header/footer

Synopsis

```
<xsl:param name="man.th.extra2.max.length">30</xsl:param>
```

Description

Specifies the maximum permitted length of the `extra2` part of the man-page part of the `.TH` title line header/footer. If the `extra2` content exceeds the maxiumum specified, it is truncated down to the maximum permitted length.

The content of the `extra2` field is usually displayed in the left footer of the page and is typically "source" data indicating the software system or product that the item documented in the man page belongs to, often in the form *`Name Version`*; for example, "GTK+ 1.2" (from the `gtk-options(7)` man page).

The default value for this parameter is reasonable but somewhat arbitrary. If you are processing pages with long "source" information, you may want to experiment with changing the value in order to achieve the correct aesthetic results.

Name

man.th.extra3.max.length — Maximum length of extra3 in header/footer

Synopsis

```
<xsl:param name="man.th.extra3.max.length">30</xsl:param>
```

Description

Specifies the maximum permitted length of the `extra3` part of the man-page `.TH` title line header/footer. If the `extra3` content exceeds the maxiumum specified, it is truncated down to the maximum permitted length.

The content of the `extra3` field is usually displayed in the middle header of the page and is typically a "manual name"; for example, "GTK+ User's Manual" (from the `gtk-options(7)` man page).

The default value for this parameter is reasonable but somewhat arbitrary. If you are processing pages with long "manual names" -- or especially if you are processing pages that have both long "title" parts (command/function, etc. names) *and* long manual names -- you may want to experiment with changing the value in order to achieve the correct aesthetic results.

Output

Name

man.output.manifest.enabled — Generate a manifest file?

Synopsis

```
<xsl:param name="man.output.manifest.enabled" select="0"></xsl:param>
```

Description

If non-zero, a list of filenames for man pages generated by the stylesheet transformation is written to the file named by the *man.output.manifest.filename* parameter.

Name

man.output.manifest.filename — Name of manifest file

Synopsis

```
<xsl:param name="man.output.manifest.filename">MAN.MANIFEST</xsl:param>
```

Description

The *man.output.manifest.filename* parameter specifies the name of the file to which the manpages manifest file is written (if the value of the *man.output.manifest.enabled* parameter is non-zero).

Name

man.output.in.separate.dir — Output man-page files in separate output directory?

Synopsis

```
<xsl:param name="man.output.in.separate.dir" select="0"></xsl:param>
```

Description

If the value of `man.output.in.separate.dir` parameter is non-zero, man-page files are output in a separate directory, specified by the *man.output.base.dir* parameter; otherwise, if the value of `man.output.in.separate.dir` is zero, man-page files are not output in a separate directory.

Name

man.output.lang.in.name.enabled — Include $LANG value in man-page filename/pathname?

Synopsis

```
<xsl:param name="man.output.lang.in.name.enabled" select="0"></xsl:param>
```

Description

The *man.output.lang.in.name.enabled* parameter specifies whether a `$lang` value is included in man-page filenames and pathnames.

If the value of *man.output.lang.in.name.enabled* is non-zero, man-page files are output with the `$lang` value included in their filenames or pathnames as follows;

- if *man.output.subdirs.enabled* is non-zero, each file is output to, e.g., a man/*$lang*/man8/foo.8 pathname
- if *man.output.subdirs.enabled* is zero, each file is output with a foo.*$lang*.8 filename

Name

man.output.base.dir — Specifies separate output directory

Synopsis

```
<xsl:param name="man.output.base.dir">man/</xsl:param>
```

Description

The *man.output.base.dir* parameter specifies the base directory into which man-page files are output. The *man.output.subdirs.enabled* parameter controls whether the files are output in subdirectories within the base directory.

Note

The values of the *man.output.base.dir* and *man.output.subdirs.enabled* parameters are used only if the value of *man.output.in.separate.dir* parameter is non-zero. If the value of the *man.output.in.separate.dir* is zero, man-page files are not output in a separate directory.

Name

man.output.subdirs.enabled — Output man-page files in subdirectories within base output directory?

Synopsis

```
<xsl:param name="man.output.subdirs.enabled" select="1"></xsl:param>
```

Description

The *man.output.subdirs.enabled* parameter controls whether man-pages files are output in subdirectories within the base directory specified by the directory specified by the *man.output.base.dir* parameter.

Note

The values of the *man.output.base.dir* and *man.output.subdirs.enabled* parameters are used only if the value of *man.output.in.separate.dir* parameter is non-zero. If the value of the *man.output.in.separate.dir* is zero, man-page files are not output in a separate directory.

Name

man.output.quietly — Suppress filename messages emitted when generating output?

Synopsis

```
<xsl:param name="man.output.quietly" select="0"></xsl:param>
```

Description

If zero (the default), for each man-page file created, a message with the name of the file is emitted. If non-zero, the files are output "quietly" -- that is, the filename messages are suppressed.

Tip

If you are processing a large amount of `refentry` content, you may be able to speed up processing significantly by setting a non-zero value for *`man.output.quietly`*.

Name

man.output.encoding — Encoding used for man-page output

Synopsis

```
<xsl:param name="man.output.encoding">UTF-8</xsl:param>
```

Description

This parameter specifies the encoding to use for files generated by the manpages stylesheet. Not all processors support specification of this parameter.

Important

If the value of the *`man.charmap.enabled`* parameter is non-zero (the default), keeping the *`man.output.encoding`* parameter at its default value (`UTF-8`) or setting it to `UTF-16` **does not cause your man pages to be output in raw UTF-8 or UTF-16** -- because any Unicode characters for which matches are found in the enabled character map will be replaced with roff escape sequences before the final man-page files are generated.

So if you want to generate "real" UTF-8 man pages, without any character substitution being performed on your content, you need to set *`man.charmap.enabled`* to zero (which will completely disable character-map processing).

You may also need to set *`man.charmap.enabled`* to zero if you want to output man pages in an encoding other than `UTF-8` or `UTF-16`. Character-map processing is based on Unicode character values and may not work with other output encodings.

Other

Name

man.table.footnotes.divider — Specifies divider string that appears before table footnotes

Synopsis

```
<xsl:param name="man.table.footnotes.divider">----</xsl:param>
```

Description

In each table that contains footenotes, the string specified by the *man.table.footnotes.divider* parameter is output before the list of footnotes for the table.

Name

man.subheading.divider.enabled — Add divider comment to roff source before/after subheadings?

Synopsis

```
<xsl:param name="man.subheading.divider.enabled">0</xsl:param>
```

Description

If the value of the *man.subheading.divider.enabled* parameter is non-zero, the contents of the *man.subheading.divider* parameter are used to add a "divider" before and after subheadings in the roff output. **The divider is not visisble in the rendered man page**; it is added as a comment, in the source, simply for the purpose of increasing reability of the source.

If *man.subheading.divider.enabled* is zero (the default), the subheading divider is suppressed.

Name

man.subheading.divider — Specifies string to use as divider comment before/after subheadings

Synopsis

```
<xsl:param \
name="man.subheading.divider">==========</xsl:param>
```

Description

If the value of the *man.subheading.divider.enabled* parameter is non-zero, the contents of the *man.subheading.divider* parameter are used to add a "divider" before and after subheadings in the roff output. **The divider is not visisble in the rendered man page**; it is added as a comment, in the source, simply for the purpose of increasing reability of the source.

If *man.subheading.divider.enabled* is zero (the default), the subheading divider is suppressed.

Part IV. Roundtrip Parameter Reference

This is reference documentation for all user-configurable parameters in the DocBook “Roundtrip” Stylesheets (for transforming DocBook to WordML, OpenDocument, and Apple Pages, and for converting from those formats back to DocBook).

Name

wordml.template — Specify the template WordML document

Synopsis

```
<xsl:param name="wordml.template"></xsl:param>
```

Description

The *wordml.template* parameter specifies a WordML document to use as a template for the generated document. The template document is used to define the (extensive) headers for the generated document, in particular the paragraph and character styles that are used to format the various elements. Any content in the template document is ignored.

A template document is used in order to allow maintenance of the paragraph and character styles to be done using Word itself, rather than these XSL stylesheets.

Name

pages.template — Specify the template Pages document

Synopsis

```
<xsl:param name="pages.template"></xsl:param>
```

Description

The *pages.template* parameter specifies a Pages (the Apple word processing application) document to use as a template for the generated document. The template document is used to define the (extensive) headers for the generated document, in particular the paragraph and character styles that are used to format the various elements. Any content in the template document is ignored.

A template document is used in order to allow maintenance of the paragraph and character styles to be done using Pages itself, rather than these XSL stylesheets.

Part V. Slides Parameter Reference

This is reference documentation for all user-configurable parameters in the DocBook XSL Slides stylesheets (for generating HTML and PDF slide presentations).

Note

The Slides stylesheet for HTML output is a customization layer of the DocBook XSL HTML stylesheet; the Slides stylesheet for FO output is a customization layer of the DocBook XSL FO stylesheet. Therefore, in addition to the slides-specific parameters listed in this section, you can also use a number of HTML stylesheet parameters[1] and FO stylesheet parameters[2] to control Slides output.

[1] ../html
[2] ../fo

HTML: General Parameters

Name

keyboard.nav — Enable keyboard navigation?

Synopsis

```
<xsl:param name="keyboard.nav" select="1"></xsl:param>
```

Description

If non-zero, JavaScript is added to the slides to enable keyboard navigation. Pressing 'n', space, or return moves forward; pressing 'p' moves backward.

Name

css.stylesheet — CSS stylesheet for slides

Synopsis

```
<xsl:param name="css.stylesheet">slides.css</xsl:param>
```

Description

Identifies the CSS stylesheet used by all the slides. This parameter can be set in the source document with the <?dbhtml?> pseudo-attribute `css-stylesheet`.

Name

css.stylesheet.dir — Default directory for CSS stylesheets

Synopsis

```
<xsl:param name="css.stylesheet.dir"></xsl:param>
```

Description

Identifies the default directory for the CSS stylesheet generated on all the slides. This parameter can be set in the source document with the <?dbhtml?> pseudo-attribute `css-stylesheet-dir`.

If non-empty, this value is prepended to each of the stylesheets.

Name

titlefoil.html — Name of title foil HTML file

Synopsis

```
<xsl:param name="titlefoil.html" select="concat('index', $html.ext)"></xsl:param>
```

Description

Sets the filename used for the slides titlepage.

Name

toc.html — Name of ToC HTML file

Synopsis

```
<xsl:param name="toc.html" select="concat('toc', $html.ext)"></xsl:param>
```

Description

Sets the filename used for the table of contents page.

Name

foilgroup.toc — Put ToC on foilgroup pages?

Synopsis

```
<xsl:param name="foilgroup.toc" select="1"></xsl:param>
```

Description

If non-zero, a ToC will be placed on foilgroup pages (after any other content).

Name

output.indent — Indent output?

Synopsis

```
<xsl:param name="output.indent">no</xsl:param>
```

Description

Specifies the setting of the *`indent`* parameter on the HTML slides. For more information, see the discussion of the `xsl:output` element in the XSLT specification.

Select from `yes` or `no`.

Name

overlay — Overlay footer navigation?

Synopsis

```
<xsl:param name="overlay" select="0"></xsl:param>
```

Description

If non-zero, JavaScript is added to the slides to make the bottom navigation appear at the bottom of each page. This option and multiframe are mutually exclusive.

If this parameter is zero, the bottom navigation simply appears below the content of each slide.

Name

show.foil.number — Show foil number on each foil?

Synopsis

```
<xsl:param name="show.foil.number" select="0"></xsl:param>
```

Description

If non-zero, on each slide there will be its number. Currently not supported in all output formats.

HTML: Frames Parameters

Name

nav.separator — Output separator between navigation and body?

Synopsis

```
<xsl:param name="nav.separator" select="1"></xsl:param>
```

Description

If non-zero, a separator (<HR>) is added between the navigation links and the content of each slide.

Name

toc.row.height — Height of ToC rows in dynamic ToCs

Synopsis

```
<xsl:param name="toc.row.height">22</xsl:param>
```

Description

This parameter specifies the height of each row in the table of contents. This is only applicable if a dynamic ToC is used. You may want to adjust this parameter for optimal appearance with the font and image sizes selected by your CSS stylesheet.

Name

toc.bg.color — Background color for ToC frame

Synopsis

```
<xsl:param name="toc.bg.color">#FFFFFF</xsl:param>
```

Description

Specifies the background color used in the ToC frame.

Name

body.bg.color — Background color for body frame

Synopsis

```
<xsl:param name="body.bg.color">#FFFFFF</xsl:param>
```

Description

Specifies the background color used in the body column of tabular slides.

Name

toc.width — Width of ToC frame

Synopsis

```
<xsl:param name="toc.width">250</xsl:param>
<!-- Presumably in pixels? -->
```

Description

Specifies the width of the ToC frame in pixels.

Name

toc.hide.show — Enable hide/show button for ToC frame

Synopsis

```
<xsl:param name="toc.hide.show" select="0"></xsl:param>
```

Description

If non-zero, JavaScript (and an additional icon, see hidetoc.image and showtoc.image) is added to each slide to allow the ToC panel to be “toggled” on each panel.

Note

There is a bug in Mozilla 1.0 (at least as of CR3) that causes the browser to reload the titlepage when this feature is used.

Name

dynamic.toc — Dynamic ToCs?

Synopsis

```
<xsl:param name="dynamic.toc" select="0"></xsl:param>
```

Description

If non-zero, JavaScript is used to make the ToC panel “dynamic”. In a dynamic ToC, each section in the ToC can be expanded and collapsed by clicking on the appropriate image.

Name

active.toc — Active ToCs?

Synopsis

```
<xsl:param name="active.toc" select="0"></xsl:param>
```

Description

If non-zero, JavaScript is used to keep the ToC and the current slide “in sync”. That is, each time the slide changes, the corresponding ToC entry will be underlined.

Name

overlay.logo — Logo to overlay on ToC frame

Synopsis

```
<xsl:param \
name="overlay.logo">http://docbook.sourceforge.net/release/buttons/slides-1.png</xsl:param>
```

Description

If this URI is non-empty, JavaScript is used to overlay the specified image on the ToC frame.

Name

multiframe — Use multiple frames for slide bodies?

Synopsis

```
<xsl:param name="multiframe" select="0"></xsl:param>
```

Description

If non-zero, multiple frames are used for the body of each slide. This is one way of forcing the slide navigation elements to appear in constant locations. The other way is with overlays. The *`overlay`* and *`multiframe`* parameters are mutually exclusive.

Name

multiframe.top.bgcolor — Background color for top navigation frame

Synopsis

```
<xsl:param name="multiframe.top.bgcolor">white</xsl:param>
```

Description

Specifies the background color of the top navigation frame when multiframe is enabled.

Name

multiframe.bottom.bgcolor — Background color for bottom navigation frame

Synopsis

```
<xsl:param name="multiframe.bottom.bgcolor">white</xsl:param>
```

Description

Specifies the background color of the bottom navigation frame when multiframe is enabled.

Name

multiframe.navigation.height — Height of navigation frames

Synopsis

```
<xsl:param name="multiframe.navigation.height">40</xsl:param>
```

Description

Specifies the height of the navigation frames in pixels when multiframe is enabled.

HTML: Graphics Parameters

Name

graphics.dir — Graphics directory

Synopsis

```
<xsl:param name="graphics.dir"></xsl:param>
```

Description

Identifies the graphics directory for the navigation components generated on all the slides. This parameter can be set in the source document with the <?dbhtml?> pseudo-attribute `graphics-dir`.

If non-empty, this value is prepended to each of the graphic image paths.

Name

bullet.image — Bullet image

Synopsis

```
<xsl:param name="bullet.image">toc/bullet.png</xsl:param>
```

Description

Specifies the filename of the bullet image used for foils in the framed ToC.

Name

next.image — Right-arrow image

Synopsis

```
<xsl:param name="next.image">active/nav-next.png</xsl:param>
```

Description

Specifies the filename of the right-pointing navigation arrow.

Name

prev.image — Left-arrow image

Synopsis

```
<xsl:param name="prev.image">active/nav-prev.png</xsl:param>
```

Description

Specifies the filename of the left-pointing navigation arrow.

Name

up.image — Up-arrow image

Synopsis

```
<xsl:param name="up.image">active/nav-up.png</xsl:param>
```

Description

Specifies the filename of the upward-pointing navigation arrow.

Name

home.image — Home image

Synopsis

```
<xsl:param name="home.image">active/nav-home.png</xsl:param>
```

Description

Specifies the filename of the home navigation icon.

Name

toc.image — ToC image

Synopsis

```
<xsl:param name="toc.image">active/nav-toc.png</xsl:param>
```

Description

Specifies the filename of the ToC navigation icon.

Name

no.next.image — Inactive right-arrow image

Synopsis

```
<xsl:param name="no.next.image">inactive/nav-next.png</xsl:param>
```

Description

Specifies the filename of the inactive right-pointing navigation arrow.

Name

no.prev.image — Inactive left-arrow image

Synopsis

```
<xsl:param name="no.prev.image">inactive/nav-prev.png</xsl:param>
```

Description

Specifies the filename of the inactive left-pointing navigation arrow.

Name

no.up.image — Inactive up-arrow image

Synopsis

```
<xsl:param name="no.up.image">inactive/nav-up.png</xsl:param>
```

Description

Specifies the filename of the inactive upward-pointing navigation arrow.

Name

no.home.image — Inactive home image

Synopsis

```
<xsl:param name="no.home.image">inactive/nav-home.png</xsl:param>
```

Description

Specifies the filename of the inactive home navigation icon.

Name

no.toc.image — Inactive ToC image

Synopsis

```
<xsl:param name="no.toc.image">inactive/nav-toc.png</xsl:param>
```

Description

Specifies the filename of the inactive ToC navigation icon.

Name

plus.image — Plus image

Synopsis

```
<xsl:param name="plus.image">toc/closed.png</xsl:param>
```

Description

Specifies the filename of the “plus” image; the image used in a dynamic ToC to indicate that a section can be expanded.

Name

minus.image — Minus image

Synopsis

```
<xsl:param name="minus.image">toc/open.png</xsl:param>
```

Description

Specifies the filename of the “minus” image; the image used in a dynamic ToC to indicate that a section can be collapsed.

Name

hidetoc.image — Hide ToC image

Synopsis

```
<xsl:param name="hidetoc.image">hidetoc.gif</xsl:param>
```

Description

Specifies the filename of the “hide ToC” image. This is used when the ToC hide/show parameter is enabled.

Name

showtoc.image — Show ToC image

Synopsis

```
<xsl:param name="showtoc.image">showtoc.gif</xsl:param>
```

Description

Specifies the filename of the “show ToC” image. This is used when the ToC hide/show parameter is enabled.

HTML: JavaScript Parameters

Name

script.dir — Script directory

Synopsis

```
<xsl:param name="script.dir"></xsl:param>
```

Description

Identifies the JavaScript source directory for the slides. This parameter can be set in the source document with the <?dbhtml?> pseudo-attribute `script-dir`.

If non-empty, this value is prepended to each of the JavaScript files.

Name

ua.js — UA JavaScript file

Synopsis

```
<xsl:param name="ua.js">ua.js</xsl:param>
```

Description

Specifies the filename of the UA JavaScript file. It's unlikely that you will ever need to change this parameter.

Name

xbDOM.js — xbDOM JavaScript file

Synopsis

```
<xsl:param name="xbDOM.js">xbDOM.js</xsl:param>
```

Description

Specifies the filename of the xbDOM JavaScript file. It's unlikely that you will ever need to change this parameter.

Name

xbStyle.js — xbStyle JavaScript file

Synopsis

```
<xsl:param name="xbStyle.js">xbStyle.js</xsl:param>
```

Description

Specifies the filename of the xbStyle JavaScript file. It's unlikely that you will ever need to change this parameter.

Name

xbLibrary.js — xbLibrary JavaScript file

Synopsis

```
<xsl:param name="xbLibrary.js">xbLibrary.js</xsl:param>
```

Description

Specifies the filename of the xbLibrary JavaScript file. It's unlikely that you will ever need to change this parameter.

Name

xbCollapsibleLists.js — xbCollapsibleLists JavaScript file

Synopsis

```
<xsl:param name="xbCollapsibleLists.js">xbCollapsibleLists.js</xsl:param>
```

Description

Specifies the filename of the xbCollapsibleLists JavaScript file. It's unlikely that you will ever need to change this parameter.

Name

overlay.js — Overlay JavaScript file

Synopsis

```
<xsl:param name="overlay.js">overlay.js</xsl:param>
```

Description

Specifies the filename of the overlay JavaScript file. It's unlikely that you will ever need to change this parameter.

Name

slides.js — Slides overlay file

Synopsis

```
<xsl:param name="slides.js">slides.js</xsl:param>
```

Description

Specifies the filename of the slides JavaScript file. It's unlikely that you will ever need to change this parameter.

HTML: Localization Parameters

Name

text.home — Home

Synopsis

```
<xsl:param name="text.home">Home</xsl:param>
```

Description

FIXME:

Name

text.toc — FIXME:

Synopsis

```
<xsl:param name="text.toc">ToC</xsl:param>
```

Description

FIXME:

Name

text.prev — FIXME:

Synopsis

```
<xsl:param name="text.prev">Prev</xsl:param>
```

Description

FIXME:

Name

text.up — FIXME:

Synopsis

```
<xsl:param name="text.up">Up</xsl:param>
```

Description

FIXME:

Name

text.next — FIXME:

Synopsis

```
<xsl:param name="text.next">Next</xsl:param>
```

Description

FIXME:

FO: General Params

Name

slide.title.font.family — Specifies font family to use for slide titles

Synopsis

```
<xsl:param name="slide.title.font.family">Helvetica</xsl:param>
```

Description

Specifies the font family to use for slides titles.

Name

slide.font.family — Specifies font family to use for slide bodies

Synopsis

```
<xsl:param name="slide.font.family">Helvetica</xsl:param>
```

Description

Specifies the font family to use for slides bodies.

Name

foil.title.master — Specifies unitless font size to use for foil titles

Synopsis

```
<xsl:param name="foil.title.master">36</xsl:param>
<!-- Inconsistant use of point size? -->
```

Description

Specifies a unitless font size to use for foil titles; used in combination with the *`foil.title.size`* parameter.

Name

foil.title.size — Specifies font size to use for foil titles, including units

Synopsis

```
   <xsl:param name="foil.title.size">
    <xsl:value-of select="$foil.title.master"></xsl:value-of><xsl:text>pt</xsl:text>
   </xsl:param>
  \
```

Description

This parameter combines the value of the *`foil.title.master`* parameter with a unit specification. The default unit is `pt` (points).

FO: Property Sets

Name

slides.properties — Specifies properties for all slides

Synopsis

```
   <xsl:attribute-set name="slides.properties">
     <xsl:attribute name="font-family">
       <xsl:value-of select="$slide.font.family"></xsl:value-of>
     </xsl:attribute>
   </xsl:attribute-set>
 \
```

Description

This parameter specifies properties that are applied to all slides.

Name

foilgroup.properties — Specifies properties for all foilgroups

Synopsis

```
   <xsl:attribute-set name="foilgroup.properties">
     <xsl:attribute name="font-family">
       <xsl:value-of select="$slide.font.family"></xsl:value-of>
     </xsl:attribute>
   </xsl:attribute-set>
 \
```

Description

This parameter specifies properties that are applied to all foilgroups.

Name

foil.subtitle.properties — Specifies properties for all foil subtitles

Synopsis

```
   <xsl:attribute-set name="foil.subtitle.properties">
     <xsl:attribute name="font-family">
       <xsl:value-of select="$slide.title.font.family"></xsl:value-of>
     </xsl:attribute>
     <xsl:attribute name="text-align">center</xsl:attribute>
     <xsl:attribute name="font-size">
       <xsl:value-of select="$foil.title.master * \
0.8"></xsl:value-of><xsl:text>pt</xsl:text>
     </xsl:attribute>
     <xsl:attribute name="space-after">12pt</xsl:attribute>
   </xsl:attribute-set>
 \
```

Description

This parameter specifies properties that are applied to all foil subtitles.

Name

foil.properties — Specifies properties for all foils

Synopsis

```
  <xsl:attribute-set name="foil.properties">
    <xsl:attribute name="font-family">
      <xsl:value-of select="$slide.font.family"></xsl:value-of>
    </xsl:attribute>
    <xsl:attribute name="margin-left">1in</xsl:attribute>
    <xsl:attribute name="margin-right">1in</xsl:attribute>
    <xsl:attribute name="font-size">
      <xsl:value-of select="$body.font.size"></xsl:value-of>
    </xsl:attribute>
    <xsl:attribute name="font-weight">bold</xsl:attribute>
  </xsl:attribute-set>
\
```

Description

This parameter specifies properties that are applied to all foils.

Name

speakernote.properties — Specifies properties for all speakernotes

Synopsis

```
  <xsl:attribute-set name="speakernote.properties">
    <xsl:attribute name="font-family">Times Roman</xsl:attribute>
    <xsl:attribute name="font-style">italic</xsl:attribute>
    <xsl:attribute name="font-size">12pt</xsl:attribute>
    <xsl:attribute name="font-weight">normal</xsl:attribute>
  </xsl:attribute-set>
\
```

Description

This parameter specifies properties that are applied to all speakernotes.

Name

running.foot.properties — Specifies properties for running foot on each slide

Synopsis

```
  <xsl:attribute-set name="running.foot.properties">
    <xsl:attribute name="font-family">
      <xsl:value-of select="$slide.font.family"></xsl:value-of>
    </xsl:attribute>
    <xsl:attribute name="font-size">14pt</xsl:attribute>
    <xsl:attribute name="color">#9F9F9F</xsl:attribute>
  </xsl:attribute-set>
\
```

Description

This parameter specifies properties that are applied to the running foot area of each slide.

Part VI. Website Parameter Reference

This is reference documentation for all user-configurable parameters in the DocBook XSL Website stylesheet (for generating websites from DocBook XML sources). Note that the Website stylesheet is a customization layer of the DocBook XSL HTML stylesheet. Therefore, in addition to the Website-specific parameters listed in this section, you can also use a number of HTML stylesheet parameters[1] to control Website output.

[1] ../html/

General Parameters

Name

autolayout-file — Identifies the autolayout.xml file

Synopsis

```
<xsl:param name="autolayout-file">autolayout.xml</xsl:param>
```

Description

When the source pages are spread over several directories, this parameter can be set (for example, from the command line of a batch-mode XSLT processor) to indicate the location of the autolayout.xml file.

FIXME: for browser-based use, there needs to be a PI for this...

Name

body.attributes — DEPRECATED

Synopsis

```
<xsl:attribute-set name="body.attributes">
  <xsl:attribute name="bgcolor">white</xsl:attribute>
  <xsl:attribute name="text">black</xsl:attribute>
  <xsl:attribute name="link">#0000FF</xsl:attribute>
  <xsl:attribute name="vlink">#840084</xsl:attribute>
  <xsl:attribute name="alink">#0000FF</xsl:attribute>
</xsl:attribute-set>
```

Description

DEPRECATED

Name

currentpage.marker — The text symbol used to mark the current page

Synopsis

```
<xsl:param name="currentpage.marker">@</xsl:param>
```

Description

Character to use as identifying the current page in

Name

dry-run — Indicates that no files should be produced

Synopsis

```
    <xsl:param name="dry-run" select="0"></xsl:param>
  \
```

Description

When using the XSLT processor to manage dependencies and construct the website, this parameter can be used to suppress the generation of new and updated files. Effectively, this allows you to see what the stylesheet would do, without actually making any changes.

Only applies when XSLT-based chunking is being used.

Name

feedback.href — HREF (URI) for feedback link

Synopsis

```
<xsl:param name="feedback.href"></xsl:param>
```

Description

The `feedback.href` value is used as the value for the `href` attribute on the feedback link. If `feedback.href` is empty, no feedback link is generated.

Name

feedback.link.text — The text of the feedback link

Synopsis

```
<xsl:param name="feedback.link.text">Feedback</xsl:param>
```

Description

The contents of this variable is used as the text of the feedback link if `feedback.href` is not empty. If `feedback.href` is empty, no feedback link is generated.

Name

feedback.with.ids — Toggle use of IDs in feedback

Synopsis

```
<xsl:param name="feedback.with.ids" select="0"></xsl:param>
```

Description

If `feedback.with.ids` is non-zero, the ID of the current page will be added to the feedback link. This can be used, for example, if the `feedback.href` is a CGI script.

Name

filename-prefix — Prefix added to all filenames

Synopsis

```
<xsl:param name="filename-prefix"></xsl:param>
```

Description

To produce the “text-only” (that is, non-tabular) layout of a website simultaneously with the tabular layout, the filenames have to be distinguished. That's accomplished by adding the `filename-prefix` to the front of each filename.

Name

footer.hr — Toggle <HR> before footer

Synopsis

```
<xsl:param name="footer.hr" select="1"></xsl:param>
```

Description

If non-zero, an <HR> is generated at the bottom of each web page, before the footer.

Name

header.hr — Toggle <HR> after header

Synopsis

```
<xsl:param name="header.hr" select="1"></xsl:param>
```

Description

If non-zero, an <HR> is generated at the bottom of each web page, before the footer.

Name

output-root — Specifies the root directory of the website

Synopsis

```
<xsl:param name="output-root">.</xsl:param>
```

Description

When using the XSLT processor to manage dependencies and construct the website, this parameter can be used to indicate the root directory where the resulting pages are placed.

Only applies when XSLT-based chunking is being used.

Name

rebuild-all — Indicates that all files should be produced

Synopsis

```
<xsl:param name="rebuild-all" select="0"></xsl:param>
```

Description

When using the XSLT processor to manage dependencies and construct the website, this parameter can be used to regenerate the whole website, updating even pages that don't appear to need to be updated.

The dependency extension only looks at the source documents. So if you change something in the stylesheet, for example, that has a global effect, you can use this parameter to force the stylesheet to rebuild the whole website.

Only applies when XSLT-based chunking is being used.

Name

sequential.links — Make sequentional links?

Synopsis

```
<xsl:param name="sequential.links" select="0"></xsl:param>
```

Description

FIXME

Name

suppress.homepage.title — Suppress title on homepage?

Synopsis

```
<xsl:param name="suppress.homepage.title" select="1"></xsl:param>
```

Description

FIXME:If non-zero, the title on the homepage is suppressed?

Name

table.spacer.image — Invisible pixel for tabular accessibility

Synopsis

```
<xsl:param name="table.spacer.image">graphics/spacer.gif</xsl:param>
```

Description

This is the 1x1 pixel, transparent pixel used for the table trick[1] to increase the accessibility of the tabular website presentation.

[1] http://diveintoaccessibility.org/day_10_presenting_your_main_content_first.html

Navigation Parameters

Name

banner.before.navigation — Put banner before navigation?

Synopsis

```
<xsl:param name="banner.before.navigation" select="1"></xsl:param>
```

Description

FIXME

Name

navbgcolor — The background color of the navigation TOC

Synopsis

```
<xsl:param name="navbgcolor">#4080FF</xsl:param>
```

Description

The background color of the navigation TOC.

Only applies with the tabular presentation is being used.

Name

navbodywidth — Specifies the width of the navigation table body

Synopsis

```
<xsl:param name="navbodywidth"></xsl:param>
```

Description

The width of the body column.

Only applies with the tabular presentation is being used.

Name

nav.table.summary — HTML Table summary attribute value for navigation tables

Synopsis

```
<xsl:param name="nav.table.summary">Navigation</xsl:param>
```

Description

The value of this parameter is used as the value of the table summary attribute for the navigation table.

Only applies with the tabular presentation is being used.

Name

navtocwidth — Specifies the width of the navigation table TOC

Synopsis

```
<xsl:param name="navtocwidth">220</xsl:param>
```

Description

The width, in pixels, of the navigation column.

Only applies with the tabular presentation is being used.

Name

textbgcolor — The background color of the table body

Synopsis

```
<xsl:param name="textbgcolor">white</xsl:param>
```

Description

The background color of the table body.

Only applies with the tabular presentation is being used.

ToC Parameters

Name

toc.blank.graphic — Use graphic for "blanks" in TOC?

Synopsis

```
<xsl:param name="toc.blank.graphic" select="1"></xsl:param>
```

Description

If non-zero, "blanks" in the the TOC will be accomplished with the graphic identified by `toc.spacer.image`.

Only applies with the tabular presentation is being used.

Name

toc.blank.image — The image for "blanks" in the TOC

Synopsis

```
<xsl:param name="toc.blank.image">graphics/blank.gif</xsl:param>
```

Description

If `toc.blank.graphic` is non-zero, this image will be used to for "blanks" in the TOC.

Only applies with the tabular presentation is being used.

Name

toc.blank.text — The text for "blanks" in the TOC

Synopsis

```
<xsl:param name="toc.blank.text">   </xsl:param>
```

Description

If `toc.blank.graphic` is zero, this text string will be used for "blanks" in the TOC.

Only applies with the tabular presentation is being used.

Name

toc.pointer.graphic — Use graphic for TOC pointer?

Synopsis

```
<xsl:param name="toc.pointer.graphic" select="1"></xsl:param>
```

Description

If non-zero, the "pointer" in the TOC will be displayed with the graphic identified by `toc.pointer.image`.

Only applies with the tabular presentation is being used.

Name

toc.pointer.image — The image for the "pointer" in the TOC

Synopsis

```
<xsl:param name="toc.pointer.image">graphics/arrow.gif</xsl:param>
```

Description

If `toc.pointer.graphic` is non-zero, this image will be used for the "pointer" in the TOC.

Only applies with the tabular presentation is being used.

Name

toc.pointer.text — The text for the "pointer" in the TOC

Synopsis

```
<xsl:param name="toc.pointer.text"> > </xsl:param>
```

Description

If `toc.pointer.graphic` is zero, this text string will be used to display the "pointer" in the TOC.

Only applies with the tabular presentation is being used.

Name

toc.spacer.graphic — Use graphic for TOC spacer?

Synopsis

```
<xsl:param name="toc.spacer.graphic" select="1"></xsl:param>
```

Description

If non-zero, the indentation in the TOC will be accomplished with the graphic identified by `toc.spacer.image`.

Only applies with the tabular presentation is being used.

Name

toc.spacer.image — The image for spacing the TOC

Synopsis

```
<xsl:param name="toc.spacer.image">graphics/blank.gif</xsl:param>
```

Description

If `toc.spacer.graphic` is non-zero, this image will be used to indent the TOC.

Only applies with the tabular presentation is being used.

Name

toc.spacer.text — The text for spacing the TOC

Synopsis

```
<xsl:param name="toc.spacer.text">   </xsl:param>
```

Description

If `toc.spacer.graphic` is zero, this text string will be used to indent the TOC.

Only applies with the tabular presentation is being used.

DocBook XSL Stylesheets
User Reference: PIs

DocBook XSL Stylesheets User Reference: PIs

Abstract

This is generated reference documentation for all user-specifiable processing instructions in the DocBook XSL stylesheets.

Note

You add these PIs at particular points in a document to cause specific “exceptions” to formatting/output behavior. To make global changes in formatting/output behavior across an entire document, it’s better to do it by setting an appropriate stylesheet parameter (if there is one).

Table of Contents

Part I. HTML Processing Instruction Reference

Introduction

This is generated reference documentation for all user-specifiable processing instructions (PIs) in the DocBook XSL stylesheets for HTML output.

Note

You add these PIs at particular points in a document to cause specific “exceptions” to formatting/output behavior. To make global changes in formatting/output behavior across an entire document, it’s better to do it by setting an appropriate stylesheet parameter (if there is one).

Name

dbhtml_background-color — Sets background color for an image

Synopsis

```
<?dbhtml background-color="color"?>
```

Use the `<?dbhtml background-color?>` PI before or after an image (`graphic`, `inlinegraphic`, `imagedata`, or `videodata` element) as a sibling to the element, to set a background color for the image.

Parameters

background-color="*color*"
: An HTML color value

Related Information in DocBook XSL: The Complete Guide[1]

Background color[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/BGcolor.html

Name

dbhtml_bgcolor — Sets background color on a table row or table cell

Synopsis

```
<?dbhtml bgcolor="color"?>
```

Use the `<?dbhtml bgcolor?>` PI as child of a table row or cell to set a background color for that table row or cell.

Parameters

bgcolor="*color*"
: An HTML color value

Related Information in DocBook XSL: The Complete Guide[1]

Cell background color[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/BGtableColor.html#CellBGColor

Name

dbhtml_cellpadding — Specifies cellpadding in table or qandaset output

Synopsis

```
<?dbhtml cellpadding="number"?>
```

Use the `<?dbhtml cellpadding?>` PI as a child of a `table` or `qandaset` to specify the value for the HTML `cellpadding` attribute in the output HTML table.

Parameters

cellpadding="*number*"
: Specifies the cellpadding

Related Global Parameters

`html.cellpadding`

Related Information in DocBook XSL: The Complete Guide[1]

Cell spacing and cell padding[2], Q and A formatting[3]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/CellSpacing.html
[3] http://www.sagehill.net/docbookxsl/QandAformat.html

Name

dbhtml_cellspacing — Specifies cellspacing in table or qandaset output

Synopsis

```
<?dbhtml cellspacing="number"?>
```

Use the `<?dbhtml cellspacing?>` PI as a child of a `table` or `qandaset` to specify the value for the HTML `cellspacing` attribute in the output HTML table.

Parameters

cellspacing="*number*"
: Specifies the cellspacing

Related Global Parameters

`html.cellspacing`

Related Information in DocBook XSL: The Complete Guide[1]

Cell spacing and cell padding[2], Q and A formatting[3]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/CellSpacing.html
[3] http://www.sagehill.net/docbookxsl/QandAformat.html

Name

dbhtml_class — Set value of the class attribute for a table row

Synopsis

```
<?dbhtml class="name"?>
```

Use the `<?dbhtml class?>` PI as a child of a `row` to specify a `class` attribute and value in the HTML output for that row.

Parameters

class="*name*"
: Specifies the class name

Related Information in DocBook XSL: The Complete Guide[1]

Table styles in HTML output[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/CSSTableCells.html

Name

dbhtml_dir — Specifies a directory name in which to write files

Synopsis

```
<?dbhtml dir="path"?>
```

When chunking output, use the `<?dbhtml dir?>` PI as a child of a chunk source to cause the output of that chunk to be written to the specified directory; also, use it as a child of a `mediaobject` to specify a directory into which any long-description files for that `mediaobject` will be written.

Parameters

dir="*path*"
: Specifies the pathname for the directory

Related Global Parameters

base.dir

Related Information in DocBook XSL: The Complete Guide[1]

dbhtml dir processing instruction[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Chunking.html#dbhtmlDirPI

Name

dbhtml_filename — Specifies a filename for a chunk

Synopsis

```
<?dbhtml filename="filename"?>
```

When chunking output, use the `<?dbhtml filename?>` PI as a child of a chunk source to specify a filename for the output file for that chunk.

Parameters

filename="*path*"
: Specifies the filename for the file

Related Global Parameters

use.id.as.filename

Related Information in DocBook XSL: The Complete Guide[1]

dbhtml filenames[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Chunking.html#DbhtmlFilenames

Name

dbhtml_funcsynopsis-style — Specifies presentation style for a funcsynopsis

Synopsis

```
<?dbhtml funcsynopsis-style="kr"|"ansi"?>
```

Use the `<?dbhtml funcsynopsis-style?>` PI as a child of a `funcprototype` or anywhere within a funcprototype control the presentation style for the `funcsynopsis` in output.

Parameters

funcsynopsis-style="kr"
: Displays the `funcprototype` in K&R style

funcsynopsis-style="ansi"
: Displays the `funcprototype` in ANSI style

Related Global Parameters

`funcsynopsis.style`

Name

dbhtml_img.src.path — Specifies a path to the location of an image file

Synopsis

```
<?dbhtml img.src.path="path"?>
```

Use the `<?dbhtml img.src.path?>` PI before or after an image (`graphic`, `inlinegraphic`, `imagedata`, or `videodata` element) as a sibling to the element, to specify a path to the location of the image; in HTML output, the value specified for the `img.src.path` attribute is prepended to the filename.

Parameters

img.src.path="*path*"
: Specifies the pathname to prepend to the name of the image file

Related Global Parameters

img.src.path

Related Information in DocBook XSL: The Complete Guide[1]

Using fileref[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/GraphicsLocations.html#UsingFileref

Name

dbhtml_label-width — Specifies the label width for a qandaset

Synopsis

```
<?dbhtml label-width="width"?>
```

Use the `<?dbhtml label-width?>` PI as a child of a `qandaset` to specify the width of labels.

Parameters

label-width="*width*"
: Specifies the label width (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Q and A formatting[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/QandAformat.html

Name

dbhtml_linenumbering.everyNth — Specifies interval for lines numbers in verbatims

Synopsis

```
<?dbhtml linenumbering.everyNth="N"?>
```

Use the `<?dbhtml linenumbering.everyNth?>` PI as a child of a “verbatim” element – `programlisting`, `screen`, `synopsis` — to specify the interval at which lines are numbered.

Parameters

linenumbering.everyNth="*N*"
: Specifies numbering interval; a number is output before every *N*th line

Related Global Parameters

`linenumbering.everyNth`

Related Information in DocBook XSL: The Complete Guide[1]

Line numbering[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/AnnotateListing.html#LineNumbering

Name

dbhtml_linenumbering.separator — Specifies separator text for line numbers in verbatims

Synopsis

```
<?dbhtml linenumbering.separator="text"?>
```

Use the `<?dbhtml linenumbering.separator?>` PI as a child of a “verbatim” element – `programlisting`, `screen`, `synopsis` — to specify the separator text output between the line numbers and content.

Parameters

linenumbering.separator="*`text`*"
: Specifies the text (zero or more characters)

Related Global Parameters

`linenumbering.separator`

Related Information in DocBook XSL: The Complete Guide[1]

Line numbering[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/AnnotateListing.html#LineNumbering

Name

dbhtml_linenumbering.width — Specifies width for line numbers in verbatims

Synopsis

```
<?dbhtml linenumbering.width="width"?>
```

Use the `<?dbhtml linenumbering.width?>` PI as a child of a “verbatim” element – `programlisting`, `screen`, `synopsis` — to specify the width set aside for line numbers.

Parameters

linenumbering.width="*width*"
: Specifies the width (inluding units)

Related Global Parameters

linenumbering.width

Related Information in DocBook XSL: The Complete Guide[1]

Line numbering[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/AnnotateListing.html#LineNumbering

Name

dbhtml_list-presentation — Specifies presentation style for a variablelist or segmentedlist

Synopsis

```
<?dbhtml list-presentation="list"|"table"?>
```

Use the `<?dbhtml list-presentation?>` PI as a child of a `variablelist` or `segmentedlist` to control the presentation style for the list (to cause it, for example, to be displayed as a table).

Parameters

list-presentation="list"
: Displays the list as a list

list-presentation="table"
: Displays the list as a table

Related Global Parameters

- *`variablelist.as.table`*
- *`segmentedlist.as.table`*

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in HTML[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#VarListFormatting

Name

dbhtml_list-width — Specifies the width of a variablelist or simplelist

Synopsis

```
<?dbhtml list-width="width"?>
```

Use the `<?dbhtml list-width?>` PI as a child of a `variablelist` or a `simplelist` presented as a table, to specify the output width.

Parameters

list-width="*width*"
: Specifies the output width (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in HTML[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#VarListFormatting

Name

dbhtml_row-height — Specifies the height for a table row

Synopsis

```
<?dbhtml row-height="height"?>
```

Use the `<?dbhtml row-height?>` PI as a child of a `row` to specify the height of the row.

Parameters

row-height="*height*"
: Specifies the label height (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Row height[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/RowHeight.html

Name

dbhtml_start — (obsolete) Sets the starting number on an ordered list

Synopsis

```
<?dbhtml start="character"?>
```

This PI is obsolete. The intent of this PI was to provide a means for setting a specific starting number for an ordered list. Instead of this PI, set a value for the `override` attribute on the first `listitem` in the list; that will have the same effect as what this PI was intended for.

Parameters

start="*`character`*"
: Specifies the character to use as the starting number; use 0-9, a-z, A-Z, or lowercase or uppercase Roman numerals

Related Information in DocBook XSL: The Complete Guide[1]

List starting number[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Orderedlists.html#ListStartNum

Name

dbhtml_table-summary — Specifies summary for table, variablelist, segmentedlist, or qandaset output

Synopsis

```
<?dbhtml table-summary="text"?>
```

Use the `<?dbhtml table-summary?>` PI as a child of a `table`, `variablelist`, `segmentedlist`, or `qandaset` to specify the text for the HTML `summary` attribute in the output HTML table.

Parameters

table-summary="*text*"
: Specifies the summary text (zero or more characters)

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in HTML[2], Table summary text[3]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#VarListFormatting
[3] http://www.sagehill.net/docbookxsl/TableSummary.html

Name

dbhtml_table-width — Specifies the width for a table

Synopsis

```
<?dbhtml table-width="width"?>
```

Use the `<?dbhtml table-width?>` PI as a child of a `table` to specify the width of the table in output.

Parameters

table-width="*width*"
: Specifies the table width (including units or as a percentage)

Related Global Parameters

default.table.width

Related Information in DocBook XSL: The Complete Guide[1]

Table width[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Tables.html#TableWidth

Name

dbhtml_term-presentation — Sets character formatting for terms in a variablelist

Synopsis

```
<?dbhtml term-presentation="bold"|"italic"|"bold-italic"?>
```

Use the `<?dbhtml term-presentation?>` PI as a child of a `variablelist` to set character formatting for the `term` output of the list.

Parameters

term-presentation="*`bold`*"
: Specifies that terms are displayed in bold

term-presentation="*`italic`*"
: Specifies that terms are displayed in italic

term-presentation="*`bold-italic`*"
: Specifies that terms are displayed in bold-italic

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in HTML[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#VarListFormatting

Name

dbhtml_term-separator — Specifies separator text among terms in a varlistentry

Synopsis

```
<?dbhtml term-separator="text"?>
```

Use the `<?dbhtml term-separator?>` PI as a child of a `variablelist` to specify the separator text among `term` instances.

Parameters

term-separator="*`text`*"
: Specifies the text (zero or more characters)

Related Global Parameters

`variablelist.term.separator`

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in HTML[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#VarListFormatting

Name

dbhtml_term-width — Specifies the term width for a variablelist

Synopsis

```
<?dbhtml term-width="width"?>
```

Use the `<?dbhtml term-width?>` PI as a child of a `variablelist` to specify the width for `term` output.

Parameters

term-width="*width*"
: Specifies the term width (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in HTML[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#VarListFormatting

Name

dbhtml_toc — Specifies whether a TOC should be generated for a qandaset

Synopsis

```
<?dbhtml toc="0"|"1"?>
```

Use the `<?dbhtml toc?>` PI as a child of a `qandaset` to specify whether a table of contents (TOC) is generated for the `qandaset`.

Parameters

toc="0"
: If zero, no TOC is generated

toc="1"
: If `1` (or any non-zero value), a TOC is generated

Related Information in DocBook XSL: The Complete Guide[1]

Q and A list of questions[2], Q and A formatting[3]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/QandAtoc.html
[3] http://www.sagehill.net/docbookxsl/QandAformat.html

Name

dbcmdlist — Generates a hyperlinked list of commands

Synopsis

```
<?dbcmdlist?>
```

Use the `<?dbcmdlist?>` PI as the child of any element (for example, `refsynopsisdiv`) containing multiple `cmdsynopsis` instances; a hyperlinked navigational “command list” will be generated at the top of output for that element, enabling users to quickly jump to each command synopsis.

Parameters

[No parameters]

Name

dbfunclist — Generates a hyperlinked list of functions

Synopsis

```
<?dbfunclist?>
```

Use the `<?dbfunclist?>` PI as the child of any element (for example, `refsynopsisdiv`) containing multiple `funcsynopsis` instances; a hyperlinked navigational “function list” will be generated at the top of output for that element, enabling users to quickly jump to to each function synopsis.

Parameters

[No parameters]

Name

dbhtml-include_href — Copies an external well-formed HTML/XML file into current doc

Synopsis

```
<?dbhtml-include href="URI"?>
```

Use the `<?dbhtml-include href?>` PI anywhere in a document to cause the contents of the file referenced by the `href` pseudo-attribute to be copied/inserted "as is" into your HTML output at the point in document order where the PI occurs in the source.

Note

The referenced file may contain plain text (as long as it is "wrapped" in an `html` element — see the note below) or markup in any arbitrary vocabulary, including HTML — but it must conform to XML well-formedness constraints (because the feature in XSLT 1.0 for opening external files, the `document()` function, can only handle files that meet XML well-formedness constraints).

Among other things, XML well-formedness constraints require a document to have *a single root element*. So if the content you want to include is plain text or is markup that does *not* have a single root element, **wrap the content in an `html` element**. The stylesheets will strip out that surrounding `html` "wrapper" when they find it, leaving just the content you want to insert.

Parameters

href="*URI*"
: Specifies the URI for the file to include; the URI can be, for example, a remote `http:` URI, or a local filesystem `file:` URI

Related Global Parameters

`textinsert.extension`

Related Information in DocBook XSL: The Complete Guide[1]

Inserting external HTML code[2], External code files[3]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/InsertExtHtml.html
[3] ExternalCode.html

Name

dbhh — Sets topic name and topic id for context-sensitive HTML Help

Synopsis

```
<?dbhh topicname="name" topicid="id"?>
```

Use the <?dbhh?> PI as a child of components that should be used as targets for context-sensitive help requests.

Parameters

topicname="*name*"
: Specifies a unique string constant that identifies a help topic

topicid="*id*"
: Specifies a unique integer value for the `topicname` string

Related Information in DocBook XSL: The Complete Guide[1]

Context-sensitive help[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/HtmlHelp.html#HHContextHelp

Part II. FO Processing Instruction Reference

Introduction

This is generated reference documentation for all user-specifiable processing instructions (PIs) in the DocBook XSL stylesheets for FO output.

Note

You add these PIs at particular points in a document to cause specific “exceptions” to formatting/output behavior. To make global changes in formatting/output behavior across an entire document, it’s better to do it by setting an appropriate stylesheet parameter (if there is one).

Name

dbfo_background-color — Sets background color for an image

Synopsis

```
<?dbfo background-color="color"?>
```

Use the `<?dbfo background-color?>` PI before or after an image (`graphic`, `inlinegraphic`, `imagedata`, or `videodata` element) as a sibling to the element, to set a background color for the image.

Parameters

background-color="*color*"
: An HTML color value

Related Information in DocBook XSL: The Complete Guide[1]

Background color[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/BGcolor.html

Name

dbfo_bgcolor — Sets background color on a table row or table cell

Synopsis

```
<?dbfo bgcolor="color"?>
```

Use the `<?dbfo bgcolor?>` PI as child of a table row or cell to set a background color for that table row or cell.

Parameters

bgcolor="*color*"
: An HTML color value

Related Information in DocBook XSL: The Complete Guide[1]

Cell background color[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/BGtableColor.html#CellBGColor

Name

dbfo_float-type — Specifies float behavior for a sidebar

Synopsis

```
<?dbfo float-type="margin.note"?>
```

Use the `<?dbfo float-type?>` PI to specify the float behavior for a `sidebar` (to cause the sidebar to be displayed as a marginal note).

Parameters

float-type="margin.note"
: Specifies that the `sidebar` should be displayed as a marginal note.

Related Global Parameters

`sidebar.float.type parameter`, *`sidebar.float.width parameter`*, *`sidebar.properties attribute-set`*, *`sidebar.title.properties`*

Related Information in DocBook XSL: The Complete Guide[1]

A sidebar as side float[2]

[1] http://www.sagehill.net/docbookxsl/
[2] SideFloats.html#SidebarFloats

Name

dbfo_glossary-presentation — Specifies presentation style for a glossary

Synopsis

```
<?dbfo glossary-presentation="list"|"blocks"?>
```

Use the `<?dbfo glossary-presentation?>` PI as a child of a `glossary` to control its presentation style.

Parameters

glossary-presentation="list"
: Displays the glossary as a list

glossary-presentation="blocks"
: Displays the glossary as blocks

Related Global Parameters

`glossary.as.blocks`

Related Information in DocBook XSL: The Complete Guide[1]

Glossary formatting in print[2]

[1] http://www.sagehill.net/docbookxsl/
[2] Glossaries.html#GlossaryFormatPrint

Name

dbfo_glosslist-presentation — Specifies presentation style for a glosslist

Synopsis

```
<?dbfo glosslist-presentation="list"|"blocks"?>
```

Use the `<?dbfo glosslist-presentation?>` PI as a child of a `glosslist` to control its presentation style.

Parameters

glosslist-presentation="list"
: Displays the glosslist as a list

glosslist-presentation="blocks"
: Displays the glosslist as blocks

Related Global Parameters

`glosslist.as.blocks`

Related Information in DocBook XSL: The Complete Guide[1]

Glossary formatting in print[2]

[1] http://www.sagehill.net/docbookxsl/
[2] Glossaries.html#GlossaryFormatPrint

Name

dbfo_glossterm-width — Specifies the glossterm width for a glossary or glosslist

Synopsis

```
<?dbfo glossterm-width="width"?>
```

Use the `<?dbfo glossterm-width?>` PI as a child of a `glossary` or `glosslist` to specify the width for output of `glossterm` instances in the output.

Parameters

glossterm-width="*`width`*"
: Specifies the glossterm width (including units)

Related Global Parameters

`glossterm.width`, *`glossterm.separation`*

Related Information in DocBook XSL: The Complete Guide[1]

Glossary formatting in print[2]

[1] http://www.sagehill.net/docbookxsl/
[2] Glossaries.html#GlossaryFormatPrint

Name

dbfo_keep-together — Specifies “keep” behavior for a table, example, figure, or equation

Synopsis

```
<?dbfo keep-together="auto"|"always"?>
```

Use the `<?dbfo keep-together?>` PI as a child of a formal object (`table`, `example`, `figure`, or `equation`) or their informal equivalents) to specify “keep” behavior for the object (to allow the object to “break” across a page).

Parameters

keep-together="auto"
: Enables the object to break across a page

keep-together="always"
: Prevents the object from breaking across a page (the default stylesheet behavior)

Related Global Parameters

formal.object.properties

Related Information in DocBook XSL: The Complete Guide[1]

Keep-together processing instruction[2]

[1] http://www.sagehill.net/docbookxsl/
[2] PageBreaking.html#KeepTogetherPI

Name

dbfo_label-width — Specifies the label width for a qandaset

Synopsis

```
<?dbfo label-width="width"?>
```

Use the `<?dbfo label-width?>` PI as a child of a `qandaset` to specify the width of labels.

Parameters

label-width="*width*"
: Specifies the label width (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Q and A formatting[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/QandAformat.html

Name

dbfo_linenumbering.everyNth — Specifies interval for lines numbers in verbatims

Synopsis

```
<?dbfo linenumbering.everyNth="N"?>
```

Use the `<?dbfo linenumbering.everyNth?>` PI as a child of a “verbatim” element – `programlisting`, `screen`, `synopsis` — to specify the interval at which lines are numbered.

Parameters

linenumbering.everyNth="*N*"
: Specifies numbering interval; a number is output before every *N*th line

Related Global Parameters

`linenumbering.everyNth`

Related Information in DocBook XSL: The Complete Guide[1]

Line numbering[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/AnnotateListing.html#LineNumbering

Name

dbfo_linenumbering.separator — Specifies separator text for line numbers in verbatims

Synopsis

```
<?dbfo linenumbering.separator="text"?>
```

Use the `<?dbfo linenumbering.separator?>` PI as a child of a “verbatim” element – `programlisting`, `screen`, `synopsis` — to specify the separator text output between the line numbers and content.

Parameters

linenumbering.separator="*`text`*"
: Specifies the text (zero or more characters)

Related Global Parameters

`linenumbering.separator`

Related Information in DocBook XSL: The Complete Guide[1]

Line numbering[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/AnnotateListing.html#LineNumbering

Name

dbfo_linenumbering.width — Specifies width for line numbers in verbatims

Synopsis

```
<?dbfo linenumbering.width="width"?>
```

Use the `<?dbfo linenumbering.width?>` PI as a child of a “verbatim” element – `programlisting`, `screen`, `synopsis` — to specify the width set aside for line numbers.

Parameters

linenumbering.width="*width*"
: Specifies the width (inluding units)

Related Global Parameters

linenumbering.width

Related Information in DocBook XSL: The Complete Guide[1]

Line numbering[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/AnnotateListing.html#LineNumbering

Name

dbfo_list-presentation — Specifies presentation style for a variablelist or segmentedlist

Synopsis

```
<?dbfo list-presentation="list"|"blocks"|"table"?>
```

Use the `<?dbfo list-presentation?>` PI as a child of a `variablelist` or `segmentedlist` to control the presentation style for the list (to cause it, for example, to be displayed as a table).

Parameters

list-presentation="list"
: Displays the list as a list

list-presentation="blocks"
: (`variablelist` only) Displays the list as blocks

list-presentation="table"
: (`segmentedlist` only) Displays the list as a table

Related Global Parameters

- *`variablelist.as.blocks`*
- *`variablelist.as.table`*

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in print[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#ListIndents

Name

dbfo_list-width — Specifies the width of a horizontal simplelist

Synopsis

```
<?dbfo list-width="width"?>
```

Use the `<?dbfo list-width?>` PI as a child of a `simplelist` whose `class` value is `horizontal`, to specify the width of the `simplelist`.

Parameters

list-width="*width*"
: Specifies the `simplelist` width (including units)

Name

dbfo_orientation — Specifies the orientation for table row or cell

Synopsis

```
<?dbfo orientation="0"|"90"|"180"|"270"|"-90"|"-180"|"-270"?>
```

Use the `<?dbfo orientation?>` PI as a child of an `table` row or cell to specify the orientation (rotation) for the row or cell.

Parameters

orientation="0"|"90"|"180"|"270"|"-90"|"-180"|"-270"
: Specifies the number of degrees by which the cell or row is rotated

Name

dbfo_pgwide — Specifies if an `equation` or `example` goes across full page width

Synopsis

```
<?dbfo pgwide="0"|"1"?>
```

Use the `<?dbfo pgwide?>` PI as a child of an `equation` or `example` to specify that the content should rendered across the full width of the page.

Parameters

pgwide="0"
: If zero, the content is rendered across the current text flow

pgwide="1"
: If `1` (or any non-zero value), the content is rendered across the full width of the page

Related Global Parameters

`pgwide.properties`

Name

dbfo_rotated-width — Specifies the width for a table `entry` or `row`

Synopsis

```
<?dbfo rotated-width="width"?>
```

Use the `<?dbfo rotated-width?>` PI as a child of an `entry` or `row` instance to specify the width of that the `entry` or `row`; or use it higher up in table to cause the width to be inherited recursively down.

Parameters

rotated-width="*`width`*"
: Specifies the width of a row or cell (including units)

Name

dbfo_sidebar-width — Specifies the width of a sidebar

Synopsis

```
<?dbfo sidebar-width="width"?>
```

Use the `<?dbfo sidebar-width?>` PI as a child of a `sidebar` to specify the width of the sidebar.

Parameters

sidebar-width="*`width`*"
: Specifies the `sidebar` width (including units)

Related Global Parameters

`sidebar.float.type parameter`, *`sidebar.float.width parameter`*, *`sidebar.properties attribute-set`*, *`sidebar.title.properties`*

Related Information in DocBook XSL: The Complete Guide[1]

A sidebar as side float[2]

[1] http://www.sagehill.net/docbookxsl/
[2] SideFloats.html#SidebarFloats

Name

dbfo_start — (obsolete) Sets the starting number on an ordered list

Synopsis

```
<?dbfo start="character"?>
```

This PI is obsolete. The intent of it was to provide a means for setting a specific starting number for an ordered list. Instead of this PI, set a value for the `override` attribute on the first `listitem` in the list; that will have the same effect as what this PI was intended for.

Parameters

start="*`character`*"
: Specifies the character to use as the starting number; use 0-9, a-z, A-Z, or lowercase or uppercase Roman numerals

Related Information in DocBook XSL: The Complete Guide[1]

List starting number[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Orderedlists.html#ListStartNum

Name

dbfo_table-width — Specifies the width for a table or for revhistory output

Synopsis

```
<?dbfo table-width="width"?>
```

Use the `<?dbfo table-width?>` PI as a child or sibling of a `table`, or as a child of an `informaltable`, `entrybl`, or `revhistory` instance (which is rendered as a table in output) to specify the width of the table in output.

Parameters

table-width="*width*"
: Specifies the table width (including units or as a percentage)

Related Information in DocBook XSL: The Complete Guide[1]

Table width[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Tables.html#TableWidth

Name

dbfo_term-width — Specifies the term width for a variablelist

Synopsis

```
<?dbfo term-width="width"?>
```

Use the `<?dbfo term-width?>` PI as a child of a `variablelist` to specify the width for `term` output.

Parameters

term-width="*width*"
: Specifies the term width (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Variable list formatting in print[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/Variablelists.html#ListIndents

Name

dbfo_toc — Specifies whether a TOC should be generated for a qandaset

Synopsis

```
<?dbfo toc="0"|"1"?>
```

Use the `<?dbfo toc?>` PI as a child of a `qandaset` to specify whether a table of contents (TOC) is generated for the `qandaset`.

Parameters

toc="0"
: If zero, no TOC is generated

toc="1"
: If `1` (or any non-zero value), a TOC is generated

Related Information in DocBook XSL: The Complete Guide[1]

Q and A list of questions[2], Q and A formatting[3]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/QandAtoc.html
[3] http://www.sagehill.net/docbookxsl/QandAformat.html

Name

dbfo-need — Specify a need for space (a kind of soft page break)

Synopsis

```
<?dbfo-need height="n" [space-before="n"]?>
```

A “need” is a request for space on a page. If the requested space is not available, the page breaks and the content that follows the need request appears on the next page. If the requested space is available, then no page break is inserted.

Parameters

height="*n*"
: The amount of height needed (including units)

space-before="*n*"
: The amount of extra vertical space to add (including units)

Related Information in DocBook XSL: The Complete Guide[1]

Soft page breaks[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/PageBreaking.html#SoftPageBreaks

Part III. Common Processing Instruction Reference

Introduction

This is generated reference documentation for all user-specifiable processing instructions (PIs) in the “common” part of the DocBook XSL stylesheets.

Note

You add these PIs at particular points in a document to cause specific “exceptions” to formatting/output behavior. To make global changes in formatting/output behavior across an entire document, it’s better to do it by setting an appropriate stylesheet parameter (if there is one).

Name

dbchoice_choice — Generates a localized choice separator

Synopsis

```
<?dbchoice choice="and"|"or"|string"?>
```

Use the `<?dbchoice choice?>` PI to generate an appropriate localized “choice” separator (for example, `and` or `or`) before the final item in an inline `simplelist`

Warning

This PI is a less-than-ideal hack; support for it may disappear in the future (particularly if and when a more appropriate means for marking up "choice" lists becomes available in DocBook).

Parameters

choice="and"
: generates a localized `and` separator

choice="or"
: generates a localized `or` separator

choice="*`string`*"
: generates a literal *`string`* separator

Name

dbtimestamp — Inserts a date timestamp

Synopsis

```
<?dbtimestamp format="formatstring" [padding="0"|"1"]?>
```

Use the `<?dbtimestamp?>` PI at any point in a source document to cause a date timestamp (a formatted string representing the current date and time) to be inserted in output of the document.

Parameters

format="*formatstring*"
: Specifies format in which the date and time are output

> **Note**
>
> For details of the content of the format string, see DocBook XSL: TCG, Date and time[1].

padding="0"|"1"
: Specifies padding behavior; if non-zero, padding is is added

[1] http://www.sagehill.net/docbookxsl/Datetime.html

Name

dbtex_delims — Generates delimiters around embedded TeX equations in output

Synopsis

```
<?dbtex delims="no"|"yes"?>
```

Use the `<?dbtex delims?>` PI as a child of a `textobject` containing embedded TeX markup, to cause that markup to be surrounded by `$` delimiter characters in output.

Parameters

dbtex delims="no"|"yes"
: Specifies whether delimiters are output

Related Global Parameters

`tex.math.delims`

Related Information in DocBook XSL: The Complete Guide[1]

DBTeXMath[2]

[1] http://www.sagehill.net/docbookxsl/
[2] http://www.sagehill.net/docbookxsl/TexMath.html

DocBook XSL Stylesheets Developer Reference

DocBook XSL Stylesheets Developer Reference

Abstract

This is technical reference documentation for developers using the DocBook XSL Stylesheets. It is not intended to be user documentation, but is instead provided for developers writing customization layers for the stylesheets.

Table of Contents

Part I. XSL Library Template Reference

Introduction

This is technical reference documentation for the vocabulary-independent “library” templates in the DocBook XSL Stylesheets.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

General Library Templates

Name

dot.count — Returns the number of “.” characters in a string

Description

Given a string, the `dot.count` template returns the number of dot/period characters in the string. This template is useful, for example, when testing the nesting level of nested inline markup (for nested emphasis, quotations, etc.).

```
<xsl:template name="dot.count">
  <!-- Returns the number of "." characters in a string -->
  <xsl:param name="string"></xsl:param>
  <xsl:param name="count" select="0"></xsl:param>
  <xsl:choose>
    <xsl:when test="contains($string, '.')">
      <xsl:call-template name="dot.count">
        <xsl:with-param name="string" select="substring-after($string, \
'.')"></xsl:with-param>
        <xsl:with-param name="count" select="$count+1"></xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$count"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

copy-string — Returns “count” copies of a string

Description

Given a string, the `copy-string` template creates *`n`* copies of the string, when the value of *`n`* is given by the *`count`* parameter.

```
<xsl:template name="copy-string">
  <!-- returns 'count' copies of 'string' -->
  <xsl:param name="string"></xsl:param>
  <xsl:param name="count" select="0"></xsl:param>
  <xsl:param name="result"></xsl:param>

  <xsl:choose>
    <xsl:when test="$count>0">
      <xsl:call-template name="copy-string">
        <xsl:with-param name="string" select="$string"></xsl:with-param>
        <xsl:with-param name="count" select="$count - 1"></xsl:with-param>
        <xsl:with-param name="result">
          <xsl:value-of select="$result"></xsl:value-of>
          <xsl:value-of select="$string"></xsl:value-of>
        </xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$result"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

string.subst — Substitute one text string for another in a string

Description

The `string.subst` template replaces all occurances of *`target`* in *`string`* with *`replacement`* and returns the result.

```
<xsl:template name="string.subst">
  <xsl:param name="string"></xsl:param>
  <xsl:param name="target"></xsl:param>
  <xsl:param name="replacement"></xsl:param>

  <xsl:choose>
    <xsl:when test="contains($string, $target)">
      <xsl:variable name="rest">
        <xsl:call-template name="string.subst">
          <xsl:with-param name="string" select="substring-after($string, \
$target)"></xsl:with-param>
          <xsl:with-param name="target" select="$target"></xsl:with-param>
          <xsl:with-param name="replacement" select="$replacement"></xsl:with-param>
        </xsl:call-template>
      </xsl:variable>
      <xsl:value-of select="concat(substring-before($string, $target),             \
                    $replacement,                                          \
$rest)"></xsl:value-of>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$string"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

xpointer.idref — Extract IDREF from an XPointer

Description

The `xpointer.idref` template returns the ID portion of an XPointer which is a pointer to an ID within the current document, or the empty string if it is not.

In other words, `xpointer.idref` returns “foo” when passed either `#foo` or `#xpointer(id('foo'))`, otherwise it returns the empty string.

```
<xsl:template name="xpointer.idref">
  <xsl:param name="xpointer">http://...</xsl:param>
  <xsl:choose>
    <xsl:when test="starts-with($xpointer, '#xpointer(id(')">
      <xsl:variable name="rest" select="substring-after($xpointer, \
'#xpointer(id(')"></xsl:variable>
      <xsl:variable name="quote" select="substring($rest, 1, 1)"></xsl:variable>
      <xsl:value-of select="substring-before(substring-after($xpointer, $quote), \
$quote)"></xsl:value-of>
    </xsl:when>
    <xsl:when test="starts-with($xpointer, '#')">
      <xsl:value-of select="substring-after($xpointer, '#')"></xsl:value-of>
    </xsl:when>
    <!-- otherwise it's a pointer to some other document -->
  </xsl:choose>
</xsl:template>
```

Name

length-magnitude — Return the unqualified dimension from a length specification

Description

The `length-magnitude` template returns the unqualified length ("20" for "20pt") from a dimension.

```
<xsl:template name="length-magnitude">
  <xsl:param name="length" select="'0pt'"></xsl:param>

  <xsl:choose>
    <xsl:when test="string-length($length) = 0"></xsl:when>
    <xsl:when test="substring($length,1,1) = '0'                    or \
substring($length,1,1) = '1'                    or substring($length,1,1) = '2'    \
                or substring($length,1,1) = '3'                    or \
substring($length,1,1) = '4'                    or substring($length,1,1) = '5'    \
                or substring($length,1,1) = '6'                    or \
substring($length,1,1) = '7'                    or substring($length,1,1) = '8'    \
                or substring($length,1,1) = '9'                    or \
substring($length,1,1) = '.'">
      <xsl:value-of select="substring($length,1,1)"></xsl:value-of>
      <xsl:call-template name="length-magnitude">
       <xsl:with-param name="length" select="substring($length,2)"></xsl:with-param>
      </xsl:call-template>
    </xsl:when>
  </xsl:choose>
</xsl:template>
```

Name

length-units — Return the units from a length specification

Description

The `length-units` template returns the units ("pt" for "20pt") from a length. If no units are supplied on the length, the *`defauilt.units`* are returned.

```
<xsl:template name="length-units">
  <xsl:param name="length" select="'0pt'"></xsl:param>
  <xsl:param name="default.units" select="'px'"></xsl:param>
  <xsl:variable name="magnitude">
    <xsl:call-template name="length-magnitude">
      <xsl:with-param name="length" select="$length"></xsl:with-param>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="units">
    <xsl:value-of select="substring($length, \
string-length($magnitude)+1)"></xsl:value-of>
  </xsl:variable>

  <xsl:choose>
    <xsl:when test="$units = ''">
      <xsl:value-of select="$default.units"></xsl:value-of>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$units"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

length-spec — Return a fully qualified length specification

Description

The `length-spec` template returns the qualified length from a dimension. If an unqualified length is given, the *`default.units`* will be added to it.

```
<xsl:template name="length-spec">
  <xsl:param name="length" select="'0pt'"></xsl:param>
  <xsl:param name="default.units" select="'px'"></xsl:param>

  <xsl:variable name="magnitude">
    <xsl:call-template name="length-magnitude">
      <xsl:with-param name="length" select="$length"></xsl:with-param>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="units">
    <xsl:value-of select="substring($length, \
string-length($magnitude)+1)"></xsl:value-of>
  </xsl:variable>

  <xsl:value-of select="$magnitude"></xsl:value-of>
  <xsl:choose>
    <xsl:when test="$units='cm'                    or $units='mm'                  \
   or $units='in'                  or $units='pt'                    or $units='pc' \
                    or $units='px'                      or $units='em'">
      <xsl:value-of select="$units"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = ''">
      <xsl:value-of select="$default.units"></xsl:value-of>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>
        <xsl:text>Unrecognized unit of measure: </xsl:text>
        <xsl:value-of select="$units"></xsl:value-of>
        <xsl:text>.</xsl:text>
      </xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

length-in-points — Returns the size, in points, of a specified length

Description

The `length-in-points` template converts a length specification to points and returns that value as an unqualified number.

Caution

There is no way for the template to infer the size of an `em`. It relies on the default *`em.size`* which is initially `10` (for 10pt).

Similarly, converting pixels to points relies on the *`pixels.per.inch`* parameter which is initially `90`.

```
<xsl:template name="length-in-points">
  <xsl:param name="length" select="'0pt'"></xsl:param>
  <xsl:param name="em.size" select="10"></xsl:param>
  <xsl:param name="pixels.per.inch" select="90"></xsl:param>

  <xsl:variable name="magnitude">
    <xsl:call-template name="length-magnitude">
      <xsl:with-param name="length" select="$length"></xsl:with-param>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="units">
    <xsl:value-of select="substring($length, \
string-length($magnitude)+1)"></xsl:value-of>
```

```
  </xsl:variable>

  <xsl:choose>
    <xsl:when test="$units = 'pt'">
      <xsl:value-of select="$magnitude"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = 'cm'">
      <xsl:value-of select="$magnitude div 2.54 * 72.0"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = 'mm'">
      <xsl:value-of select="$magnitude div 25.4 * 72.0"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = 'in'">
      <xsl:value-of select="$magnitude * 72.0"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = 'pc'">
      <xsl:value-of select="$magnitude * 12.0"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = 'px'">
      <xsl:value-of select="$magnitude div $pixels.per.inch * 72.0"></xsl:value-of>
    </xsl:when>
    <xsl:when test="$units = 'em'">
      <xsl:value-of select="$magnitude * $em.size"></xsl:value-of>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>
        <xsl:text>Unrecognized unit of measure: </xsl:text>
        <xsl:value-of select="$units"></xsl:value-of>
        <xsl:text>.</xsl:text>
      </xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

pi-attribute — Extract a pseudo-attribute from a PI

Description

The `pi-attribute` template extracts a pseudo-attribute from a processing instruction. For example, given the PI “`<?foo bar="1" baz='red'?>`”,

```
<xsl:call-template name="pi-attribute">
  <xsl:with-param name="pis" select="processing-instruction('foo')"/>
  <xsl:with-param name="attribute" select="'baz'"/>
</xsl:call-template>
```

will return “red”. This template returns the first matching attribute that it finds. Presented with processing instructions that contain badly formed pseudo-attributes (missing or unbalanced quotes, for example), the template may silently return erroneous results.

```
<xsl:template name="pi-attribute">
  <xsl:param name="pis" select="processing-instruction('BOGUS_PI')"></xsl:param>
  <xsl:param name="attribute">filename</xsl:param>
  <xsl:param name="count">1</xsl:param>

  <xsl:choose>
    <xsl:when test="$count>count($pis)">
      <!-- not found -->
    </xsl:when>
    <xsl:otherwise>
      <xsl:variable name="pi">
        <xsl:value-of select="$pis[$count]"></xsl:value-of>
      </xsl:variable>
      <xsl:variable name="pivalue">
        <xsl:value-of select="concat(' ', normalize-space($pi))"></xsl:value-of>
```

```
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="contains($pivalue,concat(' ', $attribute, '='))">
          <xsl:variable name="rest" select="substring-after($pivalue,concat(' ', \
$attribute,'='))"></xsl:variable>
          <xsl:variable name="quote" select="substring($rest,1,1)"></xsl:variable>
          <xsl:value-of \
select="substring-before(substring($rest,2),$quote)"></xsl:value-of>
        </xsl:when>
        <xsl:otherwise>
          <xsl:call-template name="pi-attribute">
            <xsl:with-param name="pis" select="$pis"></xsl:with-param>
            <xsl:with-param name="attribute" select="$attribute"></xsl:with-param>
            <xsl:with-param name="count" select="$count + 1"></xsl:with-param>
          </xsl:call-template>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

lookup.key — Retrieve the value associated with a particular key in a table

Description

Given a table of space-delimited key/value pairs, the `lookup.key` template extracts the value associated with a particular key.

```
<xsl:template name="lookup.key">
  <xsl:param name="key" select="''"></xsl:param>
  <xsl:param name="table" select="''"></xsl:param>

  <xsl:if test="contains($table, ' ')">
    <xsl:choose>
      <xsl:when test="substring-before($table, ' ') = $key">
       <xsl:variable name="rest" select="substring-after($table, ' ')"></xsl:variable>
        <xsl:choose>
          <xsl:when test="contains($rest, ' ')">
            <xsl:value-of select="substring-before($rest, ' ')"></xsl:value-of>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="$rest"></xsl:value-of>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:otherwise>
        <xsl:call-template name="lookup.key">
          <xsl:with-param name="key" select="$key"></xsl:with-param>
         <xsl:with-param name="table" select="substring-after(substring-after($table,' \
'), ' ')"></xsl:with-param>
        </xsl:call-template>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:if>
</xsl:template>
```

Name

xpath.location — Calculate the XPath child-sequence to the current node

Description

The `xpath.location` template calculates the absolute path from the root of the tree to the current element node.

```
<xsl:template name="xpath.location">
  <xsl:param name="node" select="."></xsl:param>
  <xsl:param name="path" select="''"></xsl:param>

  <xsl:variable name="next.path">
    <xsl:value-of select="local-name($node)"></xsl:value-of>
    <xsl:if test="$path != ''">/</xsl:if>
    <xsl:value-of select="$path"></xsl:value-of>
  </xsl:variable>

  <xsl:choose>
    <xsl:when test="$node/parent::*">
      <xsl:call-template name="xpath.location">
        <xsl:with-param name="node" select="$node/parent::*"></xsl:with-param>
        <xsl:with-param name="path" select="$next.path"></xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>/</xsl:text>
      <xsl:value-of select="$next.path"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

comment-escape-string — Prepare a string for inclusion in an XML comment

Description

The `comment-escape-string` template returns a string that has been transformed so that it can safely be output as an XML comment. Internal occurrences of "--" will be replaced with "- -" and a leading and/or trailing space will be added to the string, if necessary.

```
<xsl:template name="comment-escape-string">
  <xsl:param name="string" select="''"></xsl:param>

  <xsl:if test="starts-with($string, '-')">
    <xsl:text> </xsl:text>
  </xsl:if>

  <xsl:call-template name="comment-escape-string.recursive">
    <xsl:with-param name="string" select="$string"></xsl:with-param>
  </xsl:call-template>

  <xsl:if test="substring($string, string-length($string), 1) = '-'">
    <xsl:text> </xsl:text>
  </xsl:if>
</xsl:template>
```

Name

comment-escape-string.recursive — Internal function used by comment-escape-string

Description

The `comment-escape-string.recursive` template is used by `comment-escape-string`.

```
<xsl:template name="comment-escape-string.recursive">
  <xsl:param name="string" select="''"></xsl:param>
  <xsl:choose>
    <xsl:when test="contains($string, '--')">
      <xsl:value-of select="substring-before($string, '--')"></xsl:value-of>
      <xsl:value-of select="'- -'"></xsl:value-of>
      <xsl:call-template name="comment-escape-string.recursive">
```

```
        <xsl:with-param name="string" select="substring-after($string, \
'--')"></xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$string"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

trim.text — Trim leading and trailing whitespace from a text node

Description

Given a text node, this function trims leading and trailing whitespace from it and returns the trimmed contents.

```
  <xsl:template name="trim.text">
    <xsl:param name="contents" select="."></xsl:param>
    <xsl:variable name="contents-left-trimmed">
      <xsl:call-template name="trim-left">
        <xsl:with-param name="contents" select="$contents"></xsl:with-param>
      </xsl:call-template>
    </xsl:variable>
    <xsl:variable name="contents-trimmed">
      <xsl:call-template name="trim-right">
        <xsl:with-param name="contents" \
select="$contents-left-trimmed"></xsl:with-param>
      </xsl:call-template>
    </xsl:variable>
    <xsl:value-of select="$contents-trimmed"></xsl:value-of>
  </xsl:template>

  <xsl:template name="trim-left">
    <xsl:param name="contents"></xsl:param>
    <xsl:choose>
      <xsl:when test="starts-with($contents,'
') or                       starts-with($contents,'
') or                       starts-with($contents,' ') or                       \
starts-with($contents,' ')">
        <xsl:call-template name="trim-left">
          <xsl:with-param name="contents" select="substring($contents, \
2)"></xsl:with-param>
        </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$contents"></xsl:value-of>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <xsl:template name="trim-right">
    <xsl:param name="contents"></xsl:param>
    <xsl:variable name="last-char">
      <xsl:value-of select="substring($contents, string-length($contents), \
1)"></xsl:value-of>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="($last-char = '
') or                       ($last-char = '
') or                       ($last-char = ' ') or                       ($last-char \
= ' ')">
        <xsl:call-template name="trim-right">
          <xsl:with-param name="contents" select="substring($contents, 1, \
string-length($contents) - 1)"></xsl:with-param>
        </xsl:call-template>
```

```
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$contents"></xsl:value-of>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
```

Name

str.tokenize.keep.delimiters — Tokenize a string while preserving any delimiters

Description

Based on the occurrence of one or more delimiter characters, this function breaks a string into a list of tokens and delimiters, marking up each of the tokens with a `token` element and preserving the delimiters as text nodes between the tokens.

Note

This function is a very slightly modified version of a function from the EXSLT site[1]. The original is available at:

http://www.exslt.org/str/functions/tokenize/str.tokenize.template.xsl

The `str.tokenize.keep.delimiters` function differs only in that it preserves the delimiters instead of discarding them.

```
  <xsl:template name="str.tokenize.keep.delimiters">
    <xsl:param name="string" select="''"></xsl:param>
    <xsl:param name="delimiters" select="' '"></xsl:param>
    <xsl:choose>
      <xsl:when test="not($string)"></xsl:when>
      <xsl:when test="not($delimiters)">
        <xsl:call-template name="str.tokenize.keep.delimiters-characters">
          <xsl:with-param name="string" select="$string"></xsl:with-param>
        </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
        <xsl:call-template name="str.tokenize.keep.delimiters-delimiters">
          <xsl:with-param name="string" select="$string"></xsl:with-param>
          <xsl:with-param name="delimiters" select="$delimiters"></xsl:with-param>
        </xsl:call-template>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template name="str.tokenize.keep.delimiters-characters">
    <xsl:param name="string"></xsl:param>
    <xsl:if test="$string">
      <ssb:token><xsl:value-of select="substring($string, 1, \
1)"></xsl:value-of></ssb:token>
      <xsl:call-template name="str.tokenize.keep.delimiters-characters">
       <xsl:with-param name="string" select="substring($string, 2)"></xsl:with-param>
      </xsl:call-template>
    </xsl:if>
  </xsl:template>
  <xsl:template name="str.tokenize.keep.delimiters-delimiters">
    <xsl:param name="string"></xsl:param>
    <xsl:param name="delimiters"></xsl:param>
   <xsl:variable name="delimiter" select="substring($delimiters, 1, 1)"></xsl:variable>
    <xsl:choose>
      <xsl:when test="not($delimiter)">
        <ssb:token><xsl:value-of select="$string"></xsl:value-of></ssb:token>
      </xsl:when>
      <xsl:when test="contains($string, $delimiter)">
```

[1] http://www.exslt.org/

```
        <xsl:if test="not(starts-with($string, $delimiter))">
          <xsl:call-template name="str.tokenize.keep.delimiters-delimiters">
            <xsl:with-param name="string" select="substring-before($string, \
$delimiter)"></xsl:with-param>
            <xsl:with-param name="delimiters" select="substring($delimiters, \
2)"></xsl:with-param>
          </xsl:call-template>
        </xsl:if>
        <!-- output each delimiter -->
        <xsl:value-of select="$delimiter"></xsl:value-of>
        <xsl:call-template name="str.tokenize.keep.delimiters-delimiters">
          <xsl:with-param name="string" select="substring-after($string, \
$delimiter)"></xsl:with-param>
          <xsl:with-param name="delimiters" select="$delimiters"></xsl:with-param>
        </xsl:call-template>
      </xsl:when>
      <xsl:otherwise>
        <xsl:call-template name="str.tokenize.keep.delimiters-delimiters">
          <xsl:with-param name="string" select="$string"></xsl:with-param>
          <xsl:with-param name="delimiters" select="substring($delimiters, \
2)"></xsl:with-param>
        </xsl:call-template>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
```

Name

apply-string-subst-map — Apply a string-substitution map

Description

This function applies a “string substitution” map. Use it when you want to do multiple string substitutions on the same target content. It reads in two things: *`content`*, the content on which to perform the substitution, and *`map.contents`*, a node set of elements (the names of the elements don't matter), with each element having the following attributes:

- `oldstring`, a string to be replaced
- `newstring`, a string with which to replace `oldstring`

The function uses *`map.contents`* to do substitution on *`content`*, and then returns the modified contents.

Note

This function is a very slightly modified version of Jeni Tennison’s `replace_strings` function in the multiple string replacements[1] section of Dave Pawson’s XSLT FAQ[2].

The `apply-string-subst-map` function is essentially the same function as the `apply-character-map` function; the only difference is that in the map that `apply-string-subst-map` expects, `oldstring` and `newstring` attributes are used instead of `character` and `string` attributes.

```
  <xsl:template name="apply-string-subst-map">
    <xsl:param name="content"></xsl:param>
    <xsl:param name="map.contents"></xsl:param>
    <xsl:variable name="replaced_text">
      <xsl:call-template name="string.subst">
        <xsl:with-param name="string" select="$content"></xsl:with-param>
        <xsl:with-param name="target" \
```

[1] http://www.dpawson.co.uk/xsl/sect2/StringReplace.html#d9351e13
[2] http://www.dpawson.co.uk/xsl/index.html

```
select="$map.contents[1]/@oldstring"></xsl:with-param>
          <xsl:with-param name="replacement" \
select="$map.contents[1]/@newstring"></xsl:with-param>
        </xsl:call-template>
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="$map.contents[2]">
          <xsl:call-template name="apply-string-subst-map">
            <xsl:with-param name="content" select="$replaced_text"></xsl:with-param>
            <xsl:with-param name="map.contents" select="$map.contents[position() > \
1]"></xsl:with-param>
          </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$replaced_text"></xsl:value-of>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
  \
```

Relative URI Functions

Introduction

These functions manipulate relative URI references.

The following assumptions must hold true:

1. All URIs are relative.

2. No URI contains the "`../`" sequence which would effectively move "up" the hierarchy.

If these assumptions do not hold, the results are unpredictable.

Name

count.uri.path.depth — Count the number of path components in a relative URI

Description

This function counts the number of path components in a relative URI.

```
<xsl:template name="count.uri.path.depth">
  <xsl:param name="filename" select="''"></xsl:param>
  <xsl:param name="count" select="0"></xsl:param>

  <xsl:choose>
    <xsl:when test="contains($filename, '/')">
      <xsl:call-template name="count.uri.path.depth">
        <xsl:with-param name="filename" select="substring-after($filename, \
'/')"></xsl:with-param>
        <xsl:with-param name="count" select="$count + 1"></xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$count"></xsl:value-of>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Name

trim.common.uri.paths — Trim common leading path components from a relative URI

Description

This function trims common leading path components from a relative URI.

```
<xsl:template name="trim.common.uri.paths">
  <xsl:param name="uriA" select="''"></xsl:param>
  <xsl:param name="uriB" select="''"></xsl:param>
  <xsl:param name="return" select="'A'"></xsl:param>

  <xsl:choose>
    <xsl:when test="contains($uriA, '/') and contains($uriB, '/')                  \
  and substring-before($uriA, '/') = substring-before($uriB, '/')">
      <xsl:call-template name="trim.common.uri.paths">
        <xsl:with-param name="uriA" select="substring-after($uriA, \
'/')"></xsl:with-param>
        <xsl:with-param name="uriB" select="substring-after($uriB, \
```

```
'/')"></xsl:with-param>
        <xsl:with-param name="return" select="$return"></xsl:with-param>
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
        <xsl:when test="$return = 'A'">
          <xsl:value-of select="$uriA"></xsl:value-of>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="$uriB"></xsl:value-of>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Part II. Common Template Reference

Table of Contents

Common » Base Template Reference

$Id: common.xsl 7056 2007-07-17 13:56:09Z xmldoc $

Introduction

This is technical reference documentation for the “base” set of common templates in the DocBook XSL Stylesheets.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

Name

is.component — Tests if a given node is a component-level element

Synopsis

```
<xsl:template name="is.component">
<xsl:param name="node" select="."/>
  ...
</xsl:template>
```

This template returns '1' if the specified node is a component (Chapter, Appendix, etc.), and '0' otherwise.

Parameters

node
: The node which is to be tested.

Returns

This template returns '1' if the specified node is a component (Chapter, Appendix, etc.), and '0' otherwise.

Name

is.section — Tests if a given node is a section-level element

Synopsis

```
<xsl:template name="is.section">
<xsl:param name="node" select="."/>
  ...
</xsl:template>
```

This template returns '1' if the specified node is a section (Section, Sect1, Sect2, etc.), and '0' otherwise.

Parameters

node
: The node which is to be tested.

Returns

This template returns '1' if the specified node is a section (Section, Sect1, Sect2, etc.), and '0' otherwise.

Name

section.level — Returns the hierarchical level of a section

Synopsis

```
<xsl:template name="section.level">
<xsl:param name="node" select="."/>
  ...
</xsl:template>
```

This template calculates the hierarchical level of a section. The element `sect1` is at level 1, `sect2` is at level 2, etc.

Recursive sections are calculated down to the fifth level.

Parameters

node
: The section node for which the level should be calculated. Defaults to the context node.

Returns

The section level, “1”, “2”, etc.

Name

qanda.section.level — Returns the hierarchical level of a QandASet

Synopsis

```
<xsl:template name="qanda.section.level"/>
```

This template calculates the hierarchical level of a QandASet.

Returns

The level, “1”, “2”, etc.

Name

select.mediaobject — Selects and processes an appropriate media object from a list

Synopsis

```
<xsl:template name="select.mediaobject">
<xsl:param name="olist" select="imageobject|imageobjectco                    \
|videoobject|audioobject|textobject"/>
  ...
</xsl:template>
```

This template takes a list of media objects (usually the children of a mediaobject or inlinemediaobject) and processes the "right" object.

This template relies on a template named "select.mediaobject.index" to determine which object in the list is appropriate.

If no acceptable object is located, nothing happens.

Parameters

olist
: The node list of potential objects to examine.

Returns

Calls <xsl:apply-templates> on the selected object.

Name

select.mediaobject.index — Selects the position of the appropriate media object from a list

Synopsis

```
<xsl:template name="select.mediaobject.index">
<xsl:param name="olist" select="imageobject|imageobjectco                    \
|videoobject|audioobject|textobject"/>
<xsl:param name="count">1</xsl:param>
  ...
</xsl:template>
```

This template takes a list of media objects (usually the children of a mediaobject or inlinemediaobject) and determines the "right" object. It returns the position of that object to be used by the calling template.

If the parameter *`use.role.for.mediaobject`* is nonzero, then it first checks for an object with a role attribute of the appropriate value. It takes the first of those. Otherwise, it takes the first acceptable object through a recursive pass through the list.

This template relies on a template named "is.acceptable.mediaobject" to determine if a given object is an acceptable graphic. The semantics of media objects is that the first acceptable graphic should be used.

If no acceptable object is located, no index is returned.

Parameters

olist
: The node list of potential objects to examine.

count
: The position in the list currently being considered by the recursive process.

Returns

Returns the position in the original list of the selected object.

Name

is.acceptable.mediaobject — Returns '1' if the specified media object is recognized

Synopsis

```
<xsl:template name="is.acceptable.mediaobject">
<xsl:param name="object"/>
  ...
</xsl:template>
```

This template examines a media object and returns '1' if the object is recognized as a graphic.

Parameters

object
: The media object to consider.

Returns

0 or 1

Name

check.id.unique — Warn users about references to non-unique IDs

Synopsis

```
<xsl:template name="check.id.unique">
<xsl:param name="linkend"/>
  ...
</xsl:template>
```

If passed an ID in `linkend`, `check.id.unique` prints a warning message to the user if either the ID does not exist or the ID is not unique.

Name

check.idref.targets — Warn users about incorrectly typed references

Synopsis

```
<xsl:template name="check.idref.targets">
<xsl:param name="linkend"/>
<xsl:param name="element-list"/>
  ...
</xsl:template>
```

If passed an ID in `linkend`, `check.idref.targets` makes sure that the element pointed to by the link is one of the elements listed in `element-list` and warns the user otherwise.

Name

copyright.years — Print a set of years with collapsed ranges

Synopsis

```
<xsl:template name="copyright.years">
<xsl:param name="years"/>
<xsl:param name="print.ranges" select="1"/>
<xsl:param name="single.year.ranges" select="0"/>
<xsl:param name="firstyear" select="0"/>
<xsl:param name="nextyear" select="0"/>
  ...
</xsl:template>
```

This template prints a list of year elements with consecutive years printed as a range. In other words:

```
<year>1992</year>
<year>1993</year>
<year>1994</year>
```

is printed “1992-1994”, whereas:

```
<year>1992</year>
<year>1994</year>
```

is printed “1992, 1994”.

This template assumes that all the year elements contain only decimal year numbers, that the elements are sorted in increasing numerical order, that there are no duplicates, and that all the years are expressed in full “century+year” (“1999” not “99”) notation.

Parameters

years
: The initial set of year elements.

print.ranges
: If non-zero, multi-year ranges are collapsed. If zero, all years are printed discretely.

single.year.ranges
: If non-zero, two consecutive years will be printed as a range, otherwise, they will be printed discretely. In other words, a single year range is “1991-1992” but discretely it's “1991, 1992”.

Returns

This template returns the formatted list of years.

Name

find.path.params — Search in a table for the "best" match for the node

Synopsis

```
<xsl:template name="find.path.params">
<xsl:param name="node" select="."/>
<xsl:param name="table" select="''"/>
<xsl:param name="location">
    <xsl:call-template name="xpath.location">
      <xsl:with-param name="node" select="$node"/>
    </xsl:call-template>
  </xsl:param>
  ...
</xsl:template>
```

This template searches in a table for the value that most-closely (in the typical best-match sense of XSLT) matches the current (element) node location.

Name

string.upper — Converts a string to all uppercase letters

Synopsis

```
<xsl:template name="string.upper">
<xsl:param name="string" select="''"/>
  ...
</xsl:template>
```

Given a string, this template does a language-aware conversion of that string to all uppercase letters, based on the values of the `lowercase.alpha` and `uppercase.alpha` gentext keys for the current locale. It affects only those characters found in the values of `lowercase.alpha` and `uppercase.alpha`. All other characters are left unchanged.

Parameters

string
: The string to convert to uppercase.

Name

string.lower — Converts a string to all lowercase letters

Synopsis

```
<xsl:template name="string.lower">
<xsl:param name="string" select="''"/>
  ...
</xsl:template>
```

Given a string, this template does a language-aware conversion of that string to all lowercase letters, based on the values of the `uppercase.alpha` and `lowercase.alpha` gentext keys for the current locale. It affects only those characters found in the values of `uppercase.alpha` and `lowercase.alpha`. All other characters are left unchanged.

Parameters

string
: The string to convert to lowercase.

Name

select.choice.separator — Returns localized choice separator

Synopsis

```
<xsl:template name="select.choice.separator"/>
```

This template enables auto-generation of an appropriate localized "choice" separator (for example, "and" or "or") before the final item in an inline list (though it could also be useful for generating choice separators for non-inline lists).

It currently works by evaluating a processing instruction (PI) of the form <?dbchoice choice="foo"?> :

- if the value of the `choice` pseudo-attribute is "and" or "or", returns a localized "and" or "or"
- otherwise returns the literal value of the `choice` pseudo-attribute

The latter is provided only as a temporary workaround because the locale files do not currently have translations for the word *or*. So if you want to generate a a logical "or" separator in French (for example), you currently need to do this:

```
<?dbchoice choice="ou"?>
```

Warning

The `dbchoice` processing instruction is an unfortunate hack; support for it may disappear in the future (particularly if and when a more appropriate means for marking up "choice" lists becomes available in DocBook).

Name

evaluate.info.profile — Evaluates an info profile

Synopsis

```
<xsl:template name="evaluate.info.profile">
<xsl:param name="profile"/>
<xsl:param name="info"/>
```

```
  ...
</xsl:template>
```

This template evaluates an "info profile" matching the XPath expression given by the *`profile`* parameter. It relies on the XSLT `evaluate()` extension function.

The value of the *`profile`* parameter can include the literal string `$info`. If found in the value of the *`profile`* parameter, the literal string `$info` string is replaced with the value of the *`info`* parameter, which should be a set of *`*info`* nodes; the expression is then evaluated using the XSLT `evaluate()` extension function.

Parameters

profile
: A string representing an XPath expression

info
: A set of *info nodes

Returns

Returns a node (the result of evaluating the *`profile`* parameter)

Common » Refentry Metadata Template Reference

$Id: refentry.xsl 7056 2007-07-17 13:56:09Z xmldoc $

Introduction

This is technical reference documentation for the “refentry metadata” templates in the DocBook XSL Stylesheets.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

Note

Currently, only the manpages stylesheets make use of these templates. They are, however, potentially useful elsewhere.

Name

get.refentry.metadata — Gathers metadata from a refentry and its ancestors

Synopsis

```
<xsl:template name="get.refentry.metadata">
<xsl:param name="refname"/>
<xsl:param name="info"/>
<xsl:param name="prefs"/>
  ...
</xsl:template>
```

Reference documentation for particular commands, functions, etc., is sometimes viewed in isolation from its greater "context". For example, users view Unix man pages as, well, individual pages, not as part of a "book" of some kind. Therefore, it is sometimes necessary to embed "context" information in output for each `refentry`.

However, one problem is that different users mark up that context information in different ways. Often (usually), the context information is not actually part of the content of the `refentry` itself, but instead part of the content of a parent or ancestor element to the the `refentry`. And even then, DocBook provides a variety of elements that users might potentially use to mark up the same kind of information. One user might use the `productnumber` element to mark up version information about a particular product, while another might use the `releaseinfo` element.

Taking all that in mind, the `get.refentry.metadata` template tries to gather metadata from a `refentry` element and its ancestor elements in an intelligent and user-configurable way. The basic mechanism used in the XPath expressions throughout this stylesheet is to select the relevant metadata from the *info element that is closest to the actual `refentry` – either on the `refentry` itself, or on its nearest ancestor.

Note

The `get.refentry.metadata` template is actually just sort of a "driver" template; it calls other templates that do the actual data collection, then returns the data as a set.

Parameters

refname
: The first `refname` in the refentry

info
: A set of info nodes (from a `refentry` element and its ancestors)

prefs
: A node containing user preferences (from global stylesheet parameters)

Returns

Returns a node set with the following elements. The descriptions are verbatim from the `man(7)` man page.

title
: the title of the man page (e.g., `MAN`)

section
: the section number the man page should be placed in (e.g., `7`)

date
: the date of the last revision

source
: the source of the command

manual
: the title of the manual (e.g., *Linux Programmer's Manual*)

Name

get.refentry.title — Gets title metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.title">
<xsl:param name="refname"/>
  ...
</xsl:template>
```

The `man(7)` man page describes this as "the title of the man page (e.g., `MAN`). This differs from `refname` in that, if the `refentry` has a `refentrytitle`, we use that as the `title`; otherwise, we just use first `refname` in the first `refnamediv` in the source.

Parameters

refname
: The first `refname` in the refentry

Returns

Returns a `title` node.

Name

get.refentry.section — Gets section metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.section">
<xsl:param name="refname"/>
<xsl:param name="quiet" select="0"/>
  ...
</xsl:template>
```

The `man(7)` man page describes this as "the section number the man page should be placed in (e.g., `7`)". If we do not find a `manvolnum` specified in the source, and we find that the `refentry` is for a function, we use the section number `3` ["Library calls (functions within program libraries)"]; otherwise, we default to using `1` ["Executable programs or shell commands"].

Parameters

refname
: The first `refname` in the refentry

quiet
: If non-zero, no "missing" message is emitted

Returns

Returns a string representing a section number.

Name

get.refentry.date — Gets date metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.date">
<xsl:param name="refname"/>
<xsl:param name="info"/>
<xsl:param name="prefs"/>
  ...
</xsl:template>
```

The `man(7)` man page describes this as "the date of the last revision". If we cannot find a date in the source, we generate one.

Parameters

refname
: The first `refname` in the refentry

info
: A set of info nodes (from a `refentry` element and its ancestors)

prefs
: A node containing users preferences (from global stylesheet parameters)

Returns

Returns a `date` node.

Name

get.refentry.source — Gets source metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.source">
<xsl:param name="refname"/>
<xsl:param name="info"/>
<xsl:param name="prefs"/>
  ...
</xsl:template>
```

The `man(7)` man page describes this as "the source of the command", and provides the following examples:

- For binaries, use something like: GNU, NET-2, SLS Distribution, MCC Distribution.
- For system calls, use the version of the kernel that you are currently looking at: Linux 0.99.11.
- For library calls, use the source of the function: GNU, BSD 4.3, Linux DLL 4.4.1.

The `solbook(5)` man page describes something very much like what `man(7)` calls "source", except that `solbook(5)` names it "software" and describes it like this:

> This is the name of the software product that the topic discussed on the reference page belongs to. For example UNIX commands are part of the `SunOS x.x` release.

In practice, there are many pages that simply have a version number in the "source" field. So, it looks like what we have is a two-part field, *`Name Version`*, where:

Name
: product name (e.g., BSD) or org. name (e.g., GNU)

Version
: version name

Each part is optional. If the *`Name`* is a product name, then the *`Version`* is probably the version of the product. Or there may be no *`Name`*, in which case, if there is a *`Version`*, it is probably the version of the item itself, not the product it is part of. Or, if the *`Name`* is an organization name, then there probably will be no *`Version`*.

Parameters

refname
: The first `refname` in the refentry

info
: A set of info nodes (from a `refentry` element and its ancestors)

prefs
: A node containing users preferences (from global stylesheet parameters)

Returns

Returns a `source` node.

Name

get.refentry.source.name — Gets source-name metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.source.name">
<xsl:param name="refname"/>
```

```
<xsl:param name="info"/>
<xsl:param name="prefs"/>
  ...
</xsl:template>
```

A "source name" is one part of a (potentially) two-part *`Name`* *`Version`* source field. For more details, see the documentation for the `get.refentry.source` template.

Parameters

refname
: The first `refname` in the refentry

info
: A set of info nodes (from a `refentry` element and its ancestors)

prefs
: A node containing users preferences (from global stylesheet parameters)

Returns

Depending on what output method is used for the current stylesheet, either returns a text node or possibly an element node, containing "source name" data.

Name

get.refentry.version — Gets version metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.version">
<xsl:param name="refname"/>
<xsl:param name="info"/>
<xsl:param name="prefs"/>
  ...
</xsl:template>
```

A "version" is one part of a (potentially) two-part *`Name`* *`Version`* source field. For more details, see the documentation for the `get.refentry.source` template.

Parameters

refname
: The first `refname` in the refentry

info
: A set of info nodes (from a `refentry` element and its ancestors)

prefs
: A node containing users preferences (from global stylesheet parameters)

Returns

Depending on what output method is used for the current stylesheet, either returns a text node or possibly an element node, containing "version" data.

Name

get.refentry.manual — Gets source metadata for a refentry

Synopsis

```
<xsl:template name="get.refentry.manual">
<xsl:param name="refname"/>
<xsl:param name="info"/>
<xsl:param name="prefs"/>
  ...
</xsl:template>
```

The `man(7)` man page describes this as "the title of the manual (e.g., *Linux Programmer's Manual*)". Here are some examples from existing man pages:

- *dpkg utilities* (**dpkg-name**)
- *User Contributed Perl Documentation* (**GET**)
- *GNU Development Tools* (**ld**)
- *Emperor Norton Utilities* (**ddate**)
- *Debian GNU/Linux manual* (**faked**)
- *GIMP Manual Pages* (**gimp**)
- *KDOC Documentation System* (**qt2kdoc**)

The `solbook(5)` man page describes something very much like what `man(7)` calls "manual", except that `solbook(5)` names it "sectdesc" and describes it like this:

> This is the section title of the reference page; for example `User Commands`.

Parameters

refname
: The first `refname` in the refentry

info
: A set of info nodes (from a `refentry` element and its ancestors)

prefs
: A node containing users preferences (from global stylesheet parameters)

Returns

Returns a `manual` node.

Name

get.refentry.metadata.prefs — Gets user preferences for refentry metadata gathering

Synopsis

```
<xsl:template name="get.refentry.metadata.prefs"/>
```

The DocBook XSL stylesheets include several user-configurable global stylesheet parameters for controlling `refentry` metadata gathering. Those parameters are not read directly by the other `refentry` metadata-gathering templates. Instead, they are read only by the `get.refentry.metadata.prefs` template, which assembles them into a structure that is then passed to the other `refentry` metadata-gathering templates.

So the, `get.refentry.metadata.prefs` template is the only interface to collecting stylesheet parameters for controlling `refentry` metadata gathering.

Parameters

There are no local parameters for this template; however, it does rely on a number of global parameters.

Returns

Returns a `manual` node.

Name

set.refentry.metadata — Sets content of a refentry metadata item

Synopsis

```
<xsl:template name="set.refentry.metadata">
<xsl:param name="refname"/>
<xsl:param name="info"/>
<xsl:param name="contents"/>
<xsl:param name="context"/>
<xsl:param name="preferred"/>
  ...
</xsl:template>
```

The `set.refentry.metadata` template is called each time a suitable source element is found for a certain metadata field.

Parameters

refname
: The first `refname` in the refentry

info
: A single *info node that contains the selected source element.

contents
: A node containing the selected source element.

context
: A string describing the metadata context in which the `set.refentry.metadata` template was called: either "date", "source", "version", or "manual".

Returns

Returns formatted contents of a selected source element.

Common » Utility Template Reference

$Id: utility.xsl 7101 2007-07-20 15:32:12Z xmldoc $

Introduction

This is technical reference documentation for the miscellaneous utility templates in the DocBook XSL Stylesheets.

Note

These templates are defined in a separate file from the set of “common” templates because some of the common templates reference DocBook XSL stylesheet parameters, requiring the entire set of parameters to be imported/included in any stylesheet that imports/includes the common templates.

The utility templates don’t import or include any DocBook XSL stylesheet parameters, so the utility templates can be used without importing the whole set of parameters.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

Name

log.message — Logs/emits formatted notes and warnings

Synopsis

```
<xsl:template name="log.message">
<xsl:param name="level"/>
<xsl:param name="source"/>
<xsl:param name="context-desc"/>
<xsl:param name="context-desc-field-length">12</xsl:param>
<xsl:param name="context-desc-padded">
    <xsl:if test="not($context-desc = '')">
      <xsl:call-template name="pad-string">
        <xsl:with-param name="leftRight">right</xsl:with-param>
        <xsl:with-param name="padVar" select="substring($context-desc, 1, \
$context-desc-field-length)"/>
        <xsl:with-param name="length" select="$context-desc-field-length"/>
      </xsl:call-template>
    </xsl:if>
  </xsl:param>
<xsl:param name="message"/>
<xsl:param name="message-field-length" select="45"/>
<xsl:param name="message-padded">
    <xsl:variable name="spaces-for-blank-level">
      <!-- * if the level field is blank, we'll need to pad out -->
      <!-- * the message field with spaces to compensate -->
      <xsl:choose>
        <xsl:when test="$level = ''">
          <xsl:value-of select="4 + 2"/>
          <!-- * 4 = hard-coded length of comment text ("Note" or "Warn") -->
          <!-- * + 2 = length of colon-plus-space separator ": " -->
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="0"/>
        </xsl:otherwise>
      </xsl:choose>
```

```
    </xsl:variable>
    <xsl:variable name="spaces-for-blank-context-desc">
      <!-- * if the context-description field is blank, we'll need -->
      <!-- * to pad out the message field with spaces to compensate -->
      <xsl:choose>
        <xsl:when test="$context-desc = ''">
          <xsl:value-of select="$context-desc-field-length + 2"/>
          <!-- * + 2 = length of colon-plus-space separator ": " -->
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="0"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="extra-spaces" select="$spaces-for-blank-level + \
$spaces-for-blank-context-desc"/>
    <xsl:call-template name="pad-string">
      <xsl:with-param name="leftRight">right</xsl:with-param>
      <xsl:with-param name="padVar" select="substring($message, 1, \
($message-field-length + $extra-spaces))"/>
      <xsl:with-param name="length" select="$message-field-length + $extra-spaces"/>
    </xsl:call-template>
  </xsl:param>
  ...
</xsl:template>
```

The `log.message` template is a utility template for logging/emitting formatted messages – that is, notes and warnings, along with a given log “level” and an identifier for the “source” that the message relates to.

Parameters

level
: Text to log/emit in the message-level field to indicate the message level (`Note` or `Warning`)

source
: Text to log/emit in the source field to identify the “source” to which the notification/warning relates. This can be any arbitrary string, but because the message lacks line and column numbers to identify the exact part of the source document to which it relates, the intention is that the value you pass into the `source` parameter should give the user some way to identify the portion of their source document on which to take potentially take action in response to the log message (for example, to edit, change, or add content).

  So the `source` value should be, for example, an ID, book/chapter/article title, title of some formal object, or even a string giving an XPath expression.

context-desc
: Text to log/emit in the context-description field to describe the context for the message.

context-desc-field-length
: Specifies length of the context-description field (in characters); default is 12

  If the text specified by the `context-desc` parameter is longer than the number of characters specified in `context-desc-field-length`, it is truncated to `context-desc-field-length` (12 characters by default).

  If the specified text is shorter than `context-desc-field-length`, it is right-padded out to `context-desc-field-length` (12 by default).

  If no value has been specified for the `context-desc` parameter, the field is left empty and the text of the log message begins with the value of the `message` parameter.

message
: Text to log/emit in the actual message field

message-field-length
: Specifies length of the message field (in characters); default is 45

Returns

Outputs a message (generally, to standard error).

Name

get.doc.title — Gets a title from the current document

Synopsis

```
<xsl:template name="get.doc.title"/>
```

The `get.doc.title` template is a utility template for returning the first title found in the current document.

Returns

Returns a string containing some identifying title for the current document .

Name

pad-string — Right-pads or left-pads a string out to a certain length

Synopsis

```
<xsl:template name="pad-string">
<xsl:param name="padChar" select="' '"/>
<xsl:param name="leftRight">left</xsl:param>
<xsl:param name="padVar"/>
<xsl:param name="length"/>
  ...
</xsl:template>
```

This function takes string *`padVar`* and pads it out in the direction *`rightLeft`* to the string-length *`length`*, using string *`padChar`* (a space character by default) as the padding string (note that *`padChar`* can be a string; it is not limited to just being a single character).

Note

This function began as a copy of Nate Austin's `prepend-pad` function in the Padding Content[1] section of Dave Pawson's XSLT FAQ[2].

Returns

Returns a (padded) string.

[1] http://www.dpawson.co.uk/xsl/sect2/padding.html
[2] http://www.dpawson.co.uk/xsl/index.html

Common » Character-Map Template Reference

$Id: charmap.xsl 7266 2007-08-22 11:58:42Z xmldoc $

Introduction

This is technical reference documentation for the character-map templates in the DocBook XSL Stylesheets.

Note

These templates are defined in a separate file from the set of “common” templates because some of the common templates reference DocBook XSL stylesheet parameters, requiring the entire set of parameters to be imported/included in any stylesheet that imports/includes the common templates.

The character-map templates don’t import or include any DocBook XSL stylesheet parameters, so the character-map templates can be used without importing the whole set of parameters.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

Name

apply-character-map — Applies an XSLT character map

Synopsis

```
<xsl:template name="apply-character-map">
<xsl:param name="content"/>
<xsl:param name="map.contents"/>
  ...
</xsl:template>
```

This template applies an XSLT character map[1]; that is, it causes certain individual characters to be substituted with strings of one or more characters. It is useful mainly for replacing multiple “special” characters or symbols in the same target content. It uses the value of *`map.contents`* to do substitution on *`content`*, and then returns the modified contents.

Note

This template is a very slightly modified version of Jeni Tennison’s `replace_strings` template in the multiple string replacements[2] section of Dave Pawson’s XSLT FAQ[3].

The `apply-string-subst-map` template is essentially the same template as the `apply-character-map` template; the only difference is that in the map that `apply-string-subst-map` expects, `oldstring` and `newstring` attributes are used instead of `character` and `string` attributes.

[1] http://www.w3.org/TR/xslt20/#character-maps
[2] http://www.dpawson.co.uk/xsl/sect2/StringReplace.html#d9351e13
[3] http://www.dpawson.co.uk/xsl/index.html

Parameters

content
: The content on which to perform the character-map substitution.

map.contents
: A node set of elements, with each element having the following attributes:

- `character`, a character to be replaced
- `string`, a string with which to replace `character`

Name

read-character-map — Reads in all or part of an XSLT character map

Synopsis

```
<xsl:template name="read-character-map">
<xsl:param name="use.subset"/>
<xsl:param name="subset.profile"/>
<xsl:param name="uri"/>
  ...
</xsl:template>
```

The XSLT 2.0 specification describes character maps[1] and explains how they may be used to allow a specific character appearing in a text or attribute node in a final result tree to be substituted by a specified string of characters during serialization. The `read-character-map` template provides a means for reading and using character maps with XSLT 1.0-based tools.

This template reads the character-map contents from *`uri`* (in full or in part, depending on the value of the *`use.subset`* parameter), then passes those contents to the `apply-character-map` template, along with *`content`*, the data on which to perform the character substitution.

Using the character map "in part" means that it uses only those `output-character` elements that match the XPath expression given in the value of the *`subset.profile`* parameter. The current implementation of that capability here relies on the `evaluate` extension XSLT function.

Parameters

use.subset
: Specifies whether to use a subset of the character map instead of the whole map; boolean `0` or `1`

subset.profile
: XPath expression that specifies what subset of the character map to use

uri
: URI for a character map

[1] http://www.w3.org/TR/xslt20/#character-maps

Part III. Formatting Object Table Reference

Introduction

This is technical reference documentation for the FO table-processing templates in the DocBook XSL Stylesheets.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

Name

calc.column.width — Calculate an XSL FO table column width specification from a CALS table column width specification.

Synopsis

```
<xsl:template name="calc.column.width">
<xsl:param name="colwidth">1*</xsl:param>
  ...
</xsl:template>
```

CALS expresses table column widths in the following basic forms:

- *99.99units*, a fixed length specifier.
- *99.99*, a fixed length specifier without any units.
- *99.99**, a relative length specifier.
- *99.99*+99.99units*, a combination of both.

The CALS units are points (pt), picas (pi), centimeters (cm), millimeters (mm), and inches (in). These are the same units as XSL, except that XSL abbreviates picas "pc" instead of "pi". If a length specifier has no units, the CALS default unit (pt) is assumed.

Relative length specifiers are represented in XSL with the proportional-column-width() function.

Here are some examples:

- "36pt" becomes "36pt"
- "3pi" becomes "3pc"
- "36" becomes "36pt"
- "3*" becomes "proportional-column-width(3)"
- "3*+2pi" becomes "proportional-column-width(3)+2pc"
- "1*+2" becomes "proportional-column-width(1)+2pt"

Parameters

colwidth
: The CALS column width specification.

Returns

The XSL column width specification.

Part IV. Titlepage Template Stylesheet Reference

Introduction

This is technical reference documentation for the “titlepage” templates in the DocBook XSL Stylesheets.

This is not intended to be user documentation. It is provided for developers writing customization layers for the stylesheets.

Name

t:templates — Construct a stylesheet for the templates provided

Synopsis

```
<xsl:template match="t:templates"/>
```

The `t:templates` element is the root of a set of templates. This template creates an appropriate `xsl:stylesheet` for the templates.

If the `t:templates` element has a `base-stylesheet` attribute, an `xsl:import` statement is constructed for it.

Name

xsl:* — Copy xsl: elements straight through

Synopsis

```
<xsl:template match="xsl:*"/>
```

This template simply copies the xsl: elements straight through into the result tree.

Name

t:titlepage — Create the templates necessary to construct a title page

Synopsis

```
<xsl:template match="t:titlepage"/>
```

The `t:titlepage` element creates a set of templates for processing the titlepage for an element. The "root" of this template set is the template named "`wrapper.titlepage`". That is the template that should be called to generate the title page.

The `t:titlepage` element has three attributes:

element
: The name of the source document element for which these templates apply. In other words, to make a title page for the `article` element, set the `element` attribute to "`article`". This attribute is required.

wrapper
: The entire title page can be wrapped with an element. This attribute identifies that element.

class
: If the `class` attribute is set, a `class` attribute with this value will be added to the wrapper element that surrounds the entire title page.

Any other attributes are copied through literally to the wrapper element.

The content of a `t:titlepage` is one or more `t:titlepage-content`, `t:titlepage-separator`, and `t:titlepage-before` elements.

Each of these elements may be provided for the "recto" and "verso" sides of the title page.

Name

@* (in copy.literal.atts mode) — Copy t:titlepage attributes

Synopsis

```
<xsl:template match="@*" mode="copy.literal.atts"/>
```

This template copies all of the “other” attributes from a `t:titlepage` element onto the specified wrapper.

Name

t:titlepage-content — Create templates for the content of one side of a title page

Synopsis

```
<xsl:template match="t:titlepage-content"/>
```

The title page content, that is, the elements from the source document that are rendered on the title page, can be controlled independently for the recto and verso sides of the title page.

The `t:titlepage-content` element has two attributes:

side
: Identifies the side of the page to which this title page content applies. The `side` attribute is required and must be set to either "`recto`" or "`verso`". In addition, you must specify exactly one `t:titlepage-content` for each side within each `t:titlepage`.

order
: Indicates how the order of the elements presented on the title page is determined. If the `order` is "`document`", the elements are presented in document order. Otherwise (if the `order` is "`stylesheet`"), the elements are presented in the order that they appear in the template (and consequently in the stylesheet).

The content of a `t:titlepage-content` element is a list of element names. These names should be unqualified. They identify the elements in the source document that should appear on the title page.

Each element may have a single attribute: `predicate`. The value of this attribute is used as a predicate for the expression that matches the element on which it occurs.

In other words, to put only the first three authors on the recto-side of a title page, you could specify:

```
<t:titlepage-contents side="recto">
  <!-- other titlepage elements -->
  <author predicate="[count(previous-sibling::author)<2]"/>
  <!-- other titlepage elements -->
</t:titlepage-contents>
```

Usually, the elements so named are empty. But it is possible to make one level of selection within them. Suppose that you want to process `authorgroup` elements on the title page, but you want to select only proper authors, editors, or corporate authors, not collaborators or other credited authors.

In that case, you can put a `t:or` group inside the `authorgroup` element:

```
<t:titlepage-contents side="recto">
  <!-- other titlepage elements -->
  <authorgroup>
    <t:or>
      <author/>
      <editor/>
      <corpauthor/>
    </t:or>
  </authorgroup>
  <!-- other titlepage elements -->
</t:titlepage-contents>
```

This will have the effect of automatically generating a template for processing `authorgroup`s in the title page mode, selecting only the specified children. If you need more complex processing, you'll have to construct the templates by hand.

Name

t:titlepage-separator — Create templates for the separator

Synopsis

```
<xsl:template match="t:titlepage-separator"/>
```

The title page is separated from the content which follows it by the markup specified in the `t:titlepage-separator` element.

Name

t:titlepage-before — Create templates for what precedes a title page

Synopsis

```
<xsl:template match="t:titlepage-before"/>
```

Each side of the title page is preceded by the markup specified in the `t:titlepage-before` element for that side.

Name

* (in copy mode) — Copy elements

Synopsis

```
<xsl:template match="*" mode="copy"/>
```

This template simply copies the elements that it applies to straight through into the result tree.

Name

@* (in copy mode) — Copy attributes

Synopsis

```
<xsl:template match="@*" mode="copy"/>
```

This template simply copies the attributes that it applies to straight through into the result tree.

Name

* (in document.order mode) — Create rules to process titlepage elements in document order

Synopsis

```
<xsl:template match="*" mode="document.order"/>
```

This template is called to process all of the children of the `t:titlepage-content` element. It creates the hairy select expression necessary to process each of those elements in the title page.

Note that this template automatically handles the case where some DocBook elements, like title and subtitle, can occur both inside the *info elements where metadata is usually stored and outside.

It also automatically calculates the name for the *info container and handles elements that have historically had containers with different names.

Name

* (in document.order mode) — Create rules to process titlepage elements in stylesheet order

Synopsis

```
<xsl:template match="*" mode="document.order"/>
```

This template is called to process all of the children of the `t:titlepage-content` element. It creates the set of `xsl:apply-templates` elements necessary process each of those elements in the title page.

Note that this template automatically handles the case where some DocBook elements, like title and subtitle, can occur both inside the *info elements where metadata is usually stored and outside.

It also automatically calculates the name for the *info container and handles elements that have historically had containers with different names.

Name

* (in titlepage.specialrules mode) — Create templates for special rules

Synopsis

```
<xsl:template match="*" mode="titlepage.specialrules"/>
```

This template is called to process all of the descendants of the `t:titlepage-content` element that require special processing. At present, that's just `t:or` elements.

Name

* (in titlepage.subrules mode) — Create template for individual special rules

Synopsis

```
<xsl:template match="*" mode="titlepage.subrules"/>
```

This template is called to process the children of special template elements.

Name

t:or — Process the t:or special rule

Synopsis

```
<xsl:template match="t:or"/><xsl:template match="t:or" mode="titlepage.subrules"/>
```

This template processes t:or.

Name

t:or (in titlepage.subrules mode) — Process the t:or special rule in titlepage.subrules mode

Synopsis

```
<xsl:template match="t:or" mode="titlepage.subrules"/>
```

The titlepage.subrules mode doesn't apply to t:or, so just reprocess this node in the normal mode.

Name

element-or-list — Construct the "or-list" used in the select attribute for special rules.

Synopsis

```
<xsl:template name="element-or-list">
<xsl:param name="elements" select="*"/>
<xsl:param name="element.count" select="count($elements)"/>
<xsl:param name="count" select="1"/>
<xsl:param name="orlist"/>
  ...
</xsl:template>
```

Walk through each of the children of t:or, producing the text of the select attribute.