DocBook to LaTeX Publishing

User Manual

Benoît Guillon

August 9, 2005

DocBook to LaTeX Publishing
by Benoît Guillon

Edition 7

Contents

Chapter 1

Documentation

1.1 Reference

[TDG] Norman Walsh and Leonard Muellner, *DocBook: The Definitive Guide*, Copyright © 1999, 2000, 2001 O'Reilly & Associates, Inc., 156592-580-7, O'Reilly.

Chapter 2

Introduction

2.1 Why a DB2LaTeX clone?

dblatex is actually a DB2LaTeX clone, since it takes the major part of its XSL stylesheets. So, why this project? The purpose is a bit different on these points:

- The project is end-user oriented, that is it tries to hide as much as possible the latex compiling stuff by providing a single clean script to produce directly DVI, PostScript and PDF output.
- The actual output rendering is done not only by the XSL stylesheets transformation, but also by a dedicated LaTeX package. The purpose is to allow a deep LaTeX customisation without changing the XSL stylesheets.
- Post-processing is done by Perl, to make publication faster, and to parse things not so easily done by the XSL stylesheets like tables.

2.2 Features

With dblatex you can:

- transform a DocBook XML/SGML book or article to pure LaTeX,
- compile the temporary LaTeX file with latex or pdflatex, to produce DVI, PostScript and PDF files,
- convert on the fly the figures included in the document,
- write complex tables,
- write several bibliographies,
- create an index,
- write mathematical equations in LaTeX,
- write mathematical equation in MathML,
- have revision bars,
- customise the output rendering with an XSL configuration file,
- use your own LaTeX style package.

2.3 Version

This manual is for dblatex version 0.1.4.1p1 .

2.4 Changes

2.4.1 Changes of version 0.1.4.1p1

The changes between release 0.1.4.1p1 and release 0.1.4.1 are:

- **dblatex** supports the new option `-T` *`target_style`*. It specifies which latex style to use for formatting the output. See Section 4.2 for more details.
- The configure script can select the default latex style to use with the option `--target`. Example:

```
./configure --prefix=/where/to/install --target=db2latex
```

- The use of **make** instead of **gmake** is now detected by configure.
- Any document language should be well supported, since babel is now included for the related language.
- New table support, completely re-written by David Hedley. It is very good and no Perl parsing is needed. One can use this new XSL table code by setting the parameter newtbl.use=1.
- The following XSL parameters are added:

  latex.babel.use Set to 1 the babel package corresponding to the document language is included. Set to 0 no babel package is included whatever the document language is. Default is 1.

  latex.babel.language Empty by default, this parameter forces the use of the specified babel language whatever the document language is.

  newtbl.use Set to 1, use the David Hedley table support. By default it is set to 0.

  figure.note Figure to use to render a `note` block. This parameter is added to allow new latex styles to use their own figures in admonitions.

  figure.tip Figure to use to render a `tip` block. This parameter is added to allow new latex styles to use their own figures in admonitions.

  figure.important Figure to use to render a `important` block. This parameter is added to allow new latex styles to use their own figures in admonitions.

  figure.warning Figure to use to render a `warning` block. This parameter is added to allow new latex styles to use their own figures in admonitions.

  figure.caution Figure to use to render a `caution` block. This parameter is added to allow new latex styles to use their own figures in admonitions.

- XML source files with any extension are correctly handled. Previously one needed to give XML files with extension .xml.
- Better `footnote` support: it can be used in section titles and in `term`s.
- Some latex rendering aspects are removed from the XSL stylesheets (they should never have been in these stylesheets): \parindent value, \parskip value, \thispagestyle{fancy} for pages containing chapters.
- Bug fixes.

2.4.2 Changes of version 0.1.4.1

The changes between release 0.1.4.1 and release 0.1.4 are:

- Significant `imagedata` improvement: almost all the attributes (align, valign, depth, width, scale, scalefit, contentdepth, contentwidth) are correctly managed. However percentage used for both contentdepth and contentwidth is not managed (only contentwidth percentage is then applied).
- Dblatex tries to automatically detect the image file formats of the included graphics, and convert them if necessary (and if possible) to be compatible with the TeX backend driver. It is usefull when several image formats are used within the same document, in which case the `-f` *`fig_format`* cannot be used.

- The dblatex option `-P` *`param=value`* is added. One can then set XSL parameter values directly from the command line. This is an alternative to the `-p` *`custom.xsl`* option.
- The `align` attribute is now managed for table cells spanned on several columns (i.e. row entries with `nameend` or `spanname` attributes).

2.4.3 Changes of version 0.1.4

The changes between release 0.1.4 and release 0.1.3 are:

- Deep code cleanup.
- Better table support
  - `Multicolumn` support (use of the attributes `namest`, `nameend`, `spanname`).
  - Better `frame`, `rowsep`, `colsep` attributes inheritance.
- Better bibliography support
  - `Bibliography` can be nested under any section.
  - `Biblioset` support.
  - Basic `bibliomixed` support.
- `Indextermsortas` and `class` attributes support added.
- `Imagedatawidth`, `depth`, `scale` attributes support improved. In previous releases, `scale` was used instead of `width`. Now, you should use `width` or `scale` properly.
- `Programlistinglinenumbering` attribute support added.
- Basic `glossary` support added.
- Better `reference` support. `Refnamediv` title is no more hard-coded (use of $refnamediv.title if not empty, or name automatically generated according to the lang).
- `Qandaset` improved. `Qandadiv` can be nested under any section.
- Better `xref` support. Now `xreflabel` and `endterm` work.
- The latex hyperref package is now automatically included in the dbk_core package. A customized LaTeX style package shouldn't include hyperref anymore.
- `Link` now works.
- `Trademarkclass` attribute managed (except class='service').
- A `keyword` is not displayed but is inserted in the index entries.
- Some bug fixes.

2.5 Publishing Principles

Dblatex transforms a DocBook XML/SGML document to LaTeX. Once transformed into LaTeX, standard LaTeX tools are used to produce DVI, Postcript or PDF files.

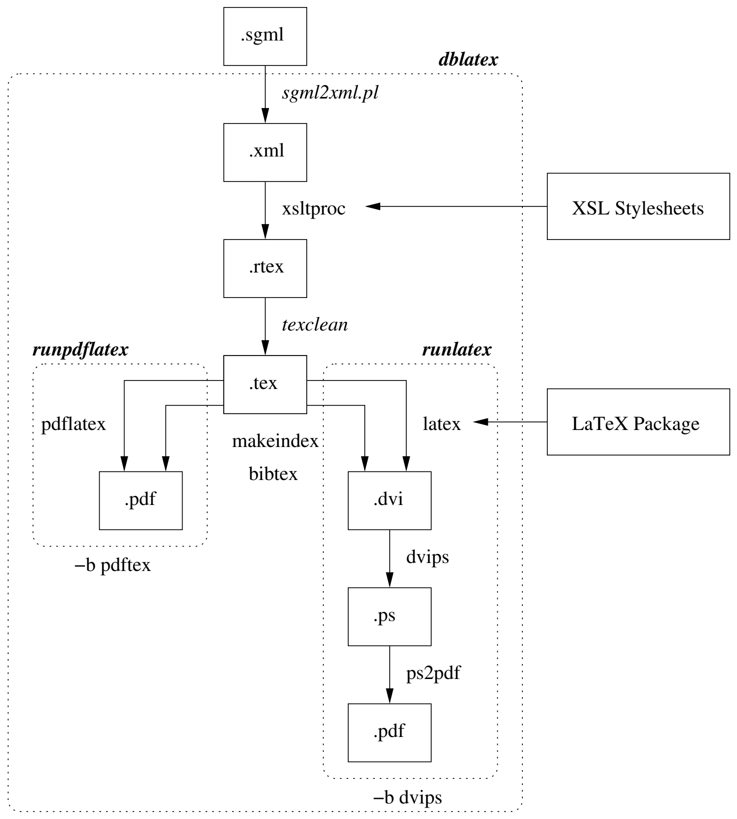
Figure 2.1 explains the process applied. It shows the tools used and the steps. The emphasized tools are provided by the package.

2.5.1 Backend Drivers

The main script allows to use two LaTeX backend drivers:

- The "dvips" driver calls **latex**, and produces DVI, Postscript and at the end PDF files. The drawback is that converting to PDF can take a while.
- The "pdftex" driver calls **pdflatex**, to produce directly PDF files. The conversion is fast, the file size is smaller, but only PDF graphics are accepted.

Figure 2.1 Transforming Process



2.5.2 XSL Stylesheets

The XSL stylesheets located under `xsl/` are used to transform from XML to “raw” LaTeX. The main file is `latex_book_fast.xsl`, that includes the other stylesheets of the directory.

2.5.3 Perl Post Processing

Actually the XSL stylesheets doesn’t produce valid LaTeX. The reason is that some DocBook processing is too complex or too time-consuming for XSL transforming. Besides, some extra actions need sometimes to be done such like figure conversion. Here are the main actions done by Perl Post processing:

- Transform the entities to valid LaTeX characters (e.g. &nbsp; is transformed to ’ ’). Perl is suited and performant for this task.
- Build the tables. DocBook Tables can be hard to translate into LaTeX (ConTeXt tables are far easier). This is why Perl is really helpfull to implement some algorithms not so easy to do by XSL programming. The main difficulties are about the missing cells, or cells spanned on several columns and/or rows.
- Convert the figures to be compatible with the backend driver. See Section 4.3 for more detail.

2.5.4 LaTeX Style Package

Once valid LaTeX is available, the LaTeX style package (docbook.sty) under `latex/style/` is used to customize the output rendering. It includes the other files of the directory. You can also provide your own LaTeX style (cf. chapter 5).

Chapter 3

Installing the package

3.1 Content

The package contains the following:

docs/ Contains the files of this document.

latex/ Contains all the latex stuff: LaTeX style files, logos, and scripts to compile the LaTeX output.

scripts/ Several scripts, including the main script of the package.

xsl/ XSL stylesheets.

tests/ Test files.

3.2 Dependencies

To work, the following items must be available:

- An XSLT. `xsltproc` is the default XSLT used.
- The XML DocBook DTD.
- A recent LaTeX distribution. The configure script checks that the needed latex packages are available.
- Perl 5.x.
- GNU make.

3.3 Installation

3.3.1 Installing the dependencies

To use the package, install properly the dependencies:

1. Install Perl 5.x and GNU make if necessary.
2. Install LaTeX.
3. Install the XSLT. By default `xsltproc` is used.
4. Install the XML DocBook DTD.
5. Create a catalog file, that defines where to find the DTD. Here is an example:

```
PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
    "file:///usr/local/share/xml/docbook/dtd/4.1.2/docbookx.dtd"
```

If the XML Gnome tools are available, it's a good idea to create an XML catalog by using `xmlcatalog` such like this:

```
% xmlcatalog --noout --create mycatalog
% xmlcatalog --noout --add 'public' '-//OASIS//DTD DocBook XML V4.1.2//EN' \
                           'file://path/to/4.1.2/docbookx.dtd' mycatalog
```

6. Add the catalog path to the SGML_CATALOG_FILES variable:

   ```
   export SGML_CATALOG_FILES=$SGML_CATALOG_FILES:/path/to/mycatalog
   ```

   You can skip this step if you configure the dblatex installation with the `--catalog` option.

3.3.2 Installing the tool

The steps to follow are the following:

1. Untar the ball. For a bzipped release, do as follow:

   ```
   % bzip2 -dc dblatex-x.x.x.tar.bz2 | tar xvf -
   ```

   For a gzipped release, do as follow:

   ```
   % gunzip -c dblatex-x.x.x.tar.gz | tar xvf -
   ```

2. Call the **configure** script, that checks the dependencies and prepares the installation. In the example, the `dblatex` script is installed under `/usr/local/bin` and the other files are installed under `/usr/local/share/dblatex`. Besides, the `--catalog` option tells where to find the catalog.

   ```
   % cd dblatex-x.x.x
   % ./configure --prefix=/usr/local --catalog=/path/to/mycatalog
   ```

3. Install:

   ```
   % gmake install
   ```

Chapter 4

Using dblatex

4.1 Publishing with dblatex

To publish your document, you just need to use the script `dblatex`.

4.1.1 Synopsis

```
dblatex [-t {tex|dvi|ps|pdf}] [-b {dvips|pdftex}] [-o output] [other options] file.{xml|sgml}
```

4.1.2 Description

The script works on an XML or SGML file and can produce LaTeX, DVI, Postscript and PDF output. By default (i.e. without option) a PDF file is produced in the same directory where the input file is, with the same base name.

4.1.3 Options

-t *format* Output format. By default the format is PDF.

-b *driver* Backend driver to use. The available drivers are "dvips" (latex) and "pdftex" (pdflatex). By default the "dvips" driver is selected. See also Section 2.5.1.

-f *fig_format* Input figures format, specified to have on the fly conversion. See also Section 4.3.2.

-I *fig_path* Additional lookup paths of the figures. See Section 4.3.3.

-o *output* Output filename. When not used, the input file name is used, with a suffix related to the output format.

-d Debug mode. It only keeps the temporary directory in which dblatex actually works. [?] explains how you can use it.

-p *config_file* Specify a configuration file. See Section 5.1.

-P *param=value* Set an XSL parameter value from command line. See Section 5.2.

-S *spec_file* Specification file. A specification file can be used to group all the options and customizations to apply. See Section 5.6.

-T *style* Rendering style to use. Several rendering style (also called LaTeX style) are provided by default. See Section 4.2.

-x *xslt_options* Options to pass to the XSLT. The example below passes the options "--timing" and "--profile" to the XSLT. Using this option supposes that you know the supported XSLT options.

```
dblatex -x "--timing --profile" file.sgml
```

4.1.4 Other options

Other options are defined to customize the tool, but they should be used in exceptional cases, for test or debug purpose. These options are defined in section Section 5.6 and should normaly be set through a specification file.

4.2 Output Formatting Style

The output rendering done by **dblatex** can be widely customized like explained in chapter 5. By default several rendering styles are provided, that one can choose by using the option `-T` (see Example 4.2.1). The available styles are:

native The rendering uses the default LaTeX stylesheets. It is the style used by default if dblatex has been configured without using the option `--target`.

simple The rendering is very close to original latex rendering. The wrapper around the default latex packages is very thin.

db2latex The rendering tries to be as close as possible to the DB2LaTeX formatting.

Example 4.2.1 Choosing the DB2LaTeX style

```
dblatex -T db2latex file.xml
```

4.2.1 How it works

The rendering style stuff is under the `latex/`directory. You can see the XSL stylesheets under `xsl/` as the way to produce latex with as little as possible docbook specific things (even if a large amount of latex packages are used to do the work). Then, it's up to LaTeX stylesheets to format the document as you wish.

The organization under `latex/` is as follow:

`contrib` Contains the non-default available LaTeX stylesheets (simple and db2latex).

`graphics` Default graphics used in the admonitions (e.g. `warning`). These graphics are used by the default output formatting.

`scripts` Scripts used to compile with **latex** or **pdflatex**.

`specs` Contains all the specification files describing the available styles. A specification file must have the extension .specs to be detected as a style description, and its basename is the name of the style. For example the style db2latex is described by the specification file `db2latex.specs`.

When **dblatex** is executed with no parameter, the usage is displayed. In particular, the list of the available styles is given, like this:

```
$ dblatex
dblatex [options] file.{sgml|xml}
Options:
-t {pdf|ps|dvi|tex|xml}: output format
...
-T style               : available latex styles (db2latex, native, simple)
```

The list is built by scanning the specs files found under `specs/`. The spec file syntax is described in Section 5.6.

`style` Default LaTeX stylesheets.

4.2.2 Adding a New Formatting Style

To add a new formatting style, do the following steps:

1. Create the latex stylesheets you need. It must define the expected DocBook interface and include some core definitions from the default latex stylesheets (cf. Section 5.4).
2. Put the latex stylesheets under a directory located under `contrib/`.

   ```
   $ mkdir latex/contrib/mystyle
   $ mv mytexstyle.sty latex/contrib/mystyle
   ```

3. If needed, create an XSL parameter file (e.g. `param.xsl`) that tunes the XSL production, and put it under `mystyle/`.
4. Create a specification file under the directory `specs/`. The specification file must point to the new latex stylesheet, and give the specific parameters. Example:

   ```
   $ cat latex/specs/mystyle.specs
   #
   # Dblatex spec file for my new style
   #
   TexInputs: ..//
   PdfInputs: ../contrib/mystyle/graphics//
   TexStyle:  mytexstyle
   XslParam: ../contrib/mystyle/param.xsl
   Options:   -b pdftex
   ```

5. That's it. Try to compile your document with your style, and check the output.

   ```
   $ dblatex -T mystyle file.xml
   ```

4.3 Figure Inclusion

4.3.1 Presentation

The expected format of the included figures depends on the backend driver used:

dvips: EPS format is required.

pdftex: PDF or PNG format is required.

In order to be able to use both backends, it is wise to not write the suffix of the file that references the figure. The suffix will be deduced from the backend used.

The figures must either already exists in the expected format, or must be able to be converted on the fly.

Example 4.3.1 Figure inclusion

```
<figure id="fig-exemple1">
  <title>Components</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="path/figure1" align="center" scale="70"/>
    </imageobject>
  </mediaobject>
</figure>
```

4.3.2 Converting on the fly

When it is needed dblatex tries to automatically convert the figures to the expected format (i.e. EPS or PDF). The principle is to detect the original figure format from the suffix of the fileref attribute. If no suffix is given, the tool checks if a file whose basename is conformant with the fileref attribute and with one of the predefined suffixes exists (that is, ".eps", ".fig", ".pdf", or ".png"). If such a file exists, conversion is done from the original format found.

The option `-f` *`fig_format`* allows to specify the default included figures format (*`fig_format`*), that will be used when automatic format scanning gives no result. Then, the tool converts the figures from the specified format to the expected one.

If the specified format is unknown, no conversion is done. The supported formats are:

fig: native format of the figures produced by XFig.

eps: Encapsulated PostScript format. This format shall be specified only when using the pdftex backend.

Example 4.3.2 Figure conversion

The following command compiles a document that contains figures produced with XFig.

```
% dblatex -f fig mydoc.sgml
```

4.3.3 Paths Lookup

You can use the option `-I "`*`path1path2`*` ..."` to specify where the figures are. The given paths must be absolute. The paths are added to the doculent root path.

Example 4.3.3 Figures lookup

This example shows how figure lookup is done. Let's consider this document source:

```
<figure id="fig-example1">
  <title>Composants</title>
  <mediaobject>
    <imageobject>
      <imagedata fileref="rep1/rep2/figure1" align="center" scale="70"/>
    </imageobject>
  </mediaobject>
</figure>
```

And the document is compiled like this:

```
% dblatex -I "/another/path /last/case" /initial/path/document.sgml
```

The figure1 lookup is done in the following directories, in respect of the order:

- `/initial/path/rep1/rep2` ;
- `/another/path/rep1/rep2` ;
- `/last/case/rep1/rep2`.

4.4 Creating Tables

DocBook tables can be quite complex. This is why the tool doesn't support all the possibilities.

The tool contains two engines to create tables: the default Perl parsing, and a new excellent implementation by David Hedley completely written in XSL. One can select this new implementation with the parameter `newtbl.use` set to 1.

Here is what is supported:

- Columns without specified widths (`colspec` without `colwidth` attribute) have the same size.
- A table width is always equal to the page width, if at least one column doesn't contain a fixed width attribute (e.g. colwidth="12cm").
- Fixed column widths are supported (e.g. colwidth="10cm").
- Fixed column widths must be declared in centimeters in default implementation ("cm"). The new XSL table implementation supports any units.
- Proportional column widths are supported (e.g. colwidth= "5*").
- Proportional and fixed colum width together is not supported (e.g. colwidth="5*+10cm"). The new XSL table implementation has a better support for this, even if it is not (yet) perfect.
- The `morerows` attribute of a table entry (`entry` element) is supported.
- The `namest` and `nameend` attributes of a table entry (`entry` element) are supported. It is possible to have a cell spanned on several columns.
- Mixing column and row spanning is not supported.
- The `orient` table attribute is supported (portrait and landscape).
- It is possible to have missing cell entries in a table.

4.4.1 Tables without colwidth

When none of the `colspec` elements contains the `colwidth` attribute, all the columns have the same size, and the table width is fixed to the maximum available size. Several examples of these tables are given.

Column 1
left aligned
no specified width, so it takes all the page

Column 1	Column 2
left aligned	centered cell
no specified width	idem

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned	left aligned	right aligned	centered cell	centered
no specified width	idem	idem	idem	idem

4.4.2 Tables with mixed colspec

A table can have `colspec` elements containing `colwidth` attribute mixed with `colspec` elements without `colwidth`. The following XML source:

```
<informaltable>
```

```
  <tgroup cols="5" colsep="1" rowsep="1" align="left">
    <colspec colname="c1"/>
    <colspec align="left" colwidth="4cm"/>
    <colspec align="right" colwidth="5cm"/>
    <colspec align="center"/>
    <colspec align="center" colwidth="3cm"/>
    <tbody>
    ...
    </tbody>
  </tgroup>
</informaltable>
```

is rendered like this:

C- o- l- u- m- n 1	Column 2	Column 3	C- o- l- u- m- n 4	Column 5
l- e- ft a- l- i- g- n- e- d (- t- g- r- o- u- p o- r- d- e- r)	left aligned	right aligned	c- e- n- t- e- r- e- d c- e- ll	in the centre

C-o-l-u-m-n 1	Column 2	Column 3	C-o-l-u-m-n 4	Column 5
n-o s-p-e-c-i-f-i-e-d w-i-d-t-h	4 cm column width	5 cm column width	n-o w-i-d-t-h	3 cm column width

4.4.3 Tables with proportional and fixed colwidth

Proportional column widths are supported. The following XML source:

```
<informaltable>
  <tgroup cols="5" colsep="1" rowsep="1" align="left">
    <colspec colname="c1" colwidth="*"/>
    <colspec align="left" colwidth="2*"/>
    <colspec align="right" colwidth="3*"/>
    <colspec align="center"/>
    <colspec align="center" colwidth="3cm"/>
    <tbody>
    ...
    </tbody>
  </tgroup>
</informaltable>
```

gives this table:

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned (tgroup level)	left aligned	right aligned	centered cell	in the centre
proporti-onal column (*)	proportional column (2*)	proportional column (3*)	no specified width	3 cm column width

4.4.4 Tables with fixed colwidths

All the columns can have fixed size, like this:

```
<informaltable>
  <tgroup cols="4" colsep="1" rowsep="1" align="left">
    <colspec colname="c1" colwidth="2cm"/>
    <colspec align="left" colwidth="2.5cm"/>
    <colspec align="right" colwidth="5cm"/>
    <colspec align="center" colwidth="3cm"/>
    <tbody>
    ...
    </tbody>
  </tgroup>
</informaltable>
```

It gives the following table:

Column 1	Column 2	Column 3	Column 4
left aligned (tgroup level)	left aligned	right aligned	centered cell
2 cm column width	2,5 cm column width	5 cm column width	4 cm column width

4.4.5 Tables with morerows

A table can contain entries that cover several lines. The following XML source contains an entry covering 4 lines:

```
<informaltable>
  <tgroup cols="4" colsep="1" rowsep="1" align="left">
    <colspec colname="c1" colwidth="*"/>
    ...
    <tbody>
    <entry morerows="3">it covers 4 lines</entry>
    ...
    </tbody>
  </tgroup>
</informaltable>
```

Here is an example of table containing several entries with morerows attribute:

Column 1	Column 2	Column 3	Column 4
cell on 4 lines	simple cell	cell on 2 lines	cell without morerow attribute
	cell in column 2		cell on 2 lines
	left aligned on 2 lines	cell in line 3, column 3	
		4 cm column width	last cell in column 4

4.4.6 Landscape tables

A table can be displayed in a lanscape format by using the `orient` attribute. The following XML source is an example.

```
<informaltable orient="land">
  <tgroup cols="5" colsep="1" rowsep="1" align="left">
    <colspec colname="c1" colwidth="*"/>
    ...
    <tbody>
    ...
    </tbody>
  </tgroup>
</informaltable>
```

Here is how it is displayed.

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned	left aligned	right aligned	centered cell	centered
no specified width	idem	idem	idem	idem

4.4.7 Smaller tables

For big tables it can be usefull to have smaller text, so that the table is not too large or too long and it can be displayed within a page. It is possible to specify smaller table text by using the `role` attribute of the elements `table` or `informaltable`.

The values and the "role" dedicated to this attribute are specific to dblatex, but it is compliant with the DocBook specification because in general the `role` attribute purpose is never defined.

The available text size definitions supported by `role` are directly taken from LaTeX:

- small,
- footnotesize,
- scriptsize,
- tiny.

Here are examples for each size.

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned	left aligned	right aligned	centered cell	centered
no specified width	idem	idem	idem	idem

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned	left aligned	right aligned	centered cell	centered
no specified width	idem	idem	idem	idem

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned	left aligned	right aligned	centered cell	centered
no specified width	idem	idem	idem	idem

Column 1	Column 2	Column 3	Column 4	Column 5
left aligned	left aligned	right aligned	centered cell	centered
no specified width	idem	idem	idem	idem

4.5 Writing LaTeX mathematical equations

4.5.1 Presentation

DocBook doesn't define elements for writing mathematical equations. Only few elements exist that tell how equation should be displayed (inlined, block):

- `inlineequation` tells that the equation is inlined,
- `informalequation` tells that the equation is displayed as a block, without a title.
- `equation` tells that the equation is displayed as a block, with or without a title.

These tags include a graphic (`graphic` or `mediaobject`) or an alternative text equation, as shown by the example.

Example 4.5.1 Equation taken from TDG

```
<equation><title>Last Theorem of Fermat</title>
  <alt>x^n + y^n &ne; z^n &forall; n &ne; 2</alt>
  <graphic fileref="figures/fermat"/></graphic>
</equation>
```

4.5.2 Implementation choice

The principle is to use only the `alt` element. If initially `alt` contains actually the text to print, it is chosen to use this element to embed LaTeX mathematical equations. This choice has the following advantages:

- The translation done by dblatex is really easy, since the equation is already written in LaTeX.
- LaTeX is one of the best word processor to render mathematical formulas.
- One doesn't need to write the equations in MathML.
- This method isn't specific to this tool (see the following section).

Besides, the implementation is as light as possible. This is why it is up to the writer to properly use the mathematical delimiters ($, \(, \), \[, \]). By this way the writer fully controls how he writes equations.

4.5.3 Compatibility

This implementation is not contradictory nor specific. In partticular, the DBTeXMath proposal to extend the DSSSL stylesheets used by jade follows the same approach, and is integrated in the Norman Walsh XSL stylesheets.

4.5.4 Examples

The following examples show how to write the equations.

Example 4.5.2 Inlined Equation

The formula $C = \alpha + \beta Y^{\gamma} + \epsilon$is inlined in the paragraph. Its SGML source is:

```
<para>The formula
  <inlineequation>
    <alt>$C = \alpha + \beta Y^{\gamma} + \epsilon$</alt>
    <graphic fileref="figures/eq1"/>
  </inlineequation>
is inlined in the paragraph. Its SGML source is:</para>
```

Example 4.5.3 Equation in a block

The following formula:

$$C = \alpha + \beta Y^{\gamma} + \epsilon$$

is displayed in a separate block. The SGML source is:

```
<para>The following formula:
  <informalequation>
    <alt>\[C = \alpha + \beta Y^{\gamma} + \epsilon\]</alt>
    <graphic fileref="figures/eq2"/>
  </informalequation>
is displayed in a separate block. The SGML source is:</para>
```

Example 4.5.4 Equation in a float

The formula Equation 4.5.1 below:

Equation 4.5.1 Simple Formula

$$C = \alpha + \beta Y^{\gamma} + \varepsilon$$

is displayed in a block with a title. Its SGML source is:

```
<para>The formula <xref linkend="eq-with-title"/> below:
  <equation id="eq-with-title">
<title>Simple Formula</title>
    <alt>\[C = \alpha + \beta Y^{\gamma} + \epsilon\]</alt>
    <graphic fileref="figures/eq3"/>
  </equation>
is displayed in a block with a title. Its SGML source is:</para>
```

4.6 Writing MathML equations

You can write MathML equations in a DocBook based document, by using the MathML Module for DocBook XML instead of the DocBook DTD.

dblatex now translates the MathML equations to latex by using the excellent stylesheets of the XSLT MathML Library by Vasil Yaroshevich. A large amount of tests from the W3C MathML Test Suite 2.0 is supported (657 of 712 tests). The test file used to validate the MathML stylesheets is provided in the package.

4.7 Creating an Index

An index is automatically generated if some index entries (`indexterm`), telling the terms to put in the index, are written in the document. The `keyword` elements are not printed but are also added to the index.

Example 4.7.1 Index Entry

```
<para>In this paragraph is described the function
<function>strcpy</function><indexterm><primary>strcpy</primary></indexterm>.
</para>
```

The index is put at the end of the document. It is not possible to put it somewhere else.

4.8 Writing a Bibliography

A bibliography (`bibliography`) can be written and put anywhere in the document. It appears as a chapter or a section and is composed by several divisions (`bibliodiv`) displayed as sections or subsections.

The writer selects information that describes each bibliography entry (`biblioentry`), and chooses the presentation order. The titles and authors are displayed first.

Example 4.8.1 A Bibliography

```
<bibliography><title>Bibliography Example</title>
  <bibliodiv><title>References</title>
    <biblioentry>
      <title>Document title</title>
      <author><firstname>J.</firstname><surname>Duval</surname></author>
      <pubsnumber>DEX000567325</pubsnumber>
    </biblioentry>
  </bibliodiv>
  <bibliodiv><title>White papers</title>
    <biblioentry>
      <title>Technical notes</title>
      <authorgroup>
        <author><firstname>J.</firstname><surname>Duval</surname></author>
        <author><firstname>R.</firstname><surname>Marion</surname></author>
      </authorgroup>
      <pubsnumber>DEX000704520</pubsnumber>
    </biblioentry>
  </bibliodiv>
</bibliography>
```

4.9 Document Revisions

The attribute `revisionflag` is usefull to identify the changes between two revisions of a document. This information is managed by dblatex, that adds revision bars in the margin of the paragraphs changed, such like in this paragraph.

Adding the revision flags can be manual, but its is tedious and error prone. The perl script diffmk by Norman Walsh can do the work for you. It works fine, but it depends on several Perl modules.

NOTE

The revision bars only appear when using the "dvips" driver. It seems to be a limitation of the LaTeX macros defined by the changebar package.

Chapter 5

Customization

The transformation process (and thus the output rendering) can be heavily customized by:

- using a configuration stylesheet,
- using customized stylesheets,
- using a customized LaTeX style package.
- using a LaTeX post process script.

All these customization methods can be used independently and in exceptional cases, but it can also be combined and registered in a master configuration file, called a specification file (cf. Section 5.6) to create a new tool dedicated to your needs.

5.1 Configuration Parameter Stylesheet

The PDF rendering can be customised by using an XSL configuration stylesheet. The configuration file is specified by using the option `-p custom.xsl`. The available configuration parameters are the following:

Parameter	Role	Default value
latex.hyperparam	cf. Section 5.1.1	empty
latex.figure.boxed	If set to 1, put the images into a framed box.	0
latex.babel.use	Set to 1 the babel package corresponding to the document language is included. Set to 0 no babel package is included whatever the document language is.	1
latex.babel.language	This parameter forces the use of the specified babel language whatever the document language is.	Empty
latex.class.options	Options passed to the \documentclass command.	Empty
newtbl.use	Set to 1, use the David Hedley table support.	0
newtbl.format.thead	LaTeX formatting for head table cells.	\bfseries%
newtbl.format.tbody	LaTeX formatting for body table cells.	Empty
newtbl.format.tfoot	LaTeX formatting for foot table cells.	Empty

Parameter	Role	Default value
newtbl.default.colsep	Set to 1, print the column separators when no `colspec` attribute is specified.	1
newtbl.default.rowsep	Set to 1, print the row separators when no `rowspec` attribute is specified.	1
figure.note	Figure to use to render a `note` block. This parameter is added to allow new latex styles to use their own figures in admonitions.	Empty
figure.tip	Figure to use to render a `tip` block. This parameter is added to allow new latex styles to use their own figures in admonitions.	Empty
figure.important	Figure to use to render a `important` block. This parameter is added to allow new latex styles to use their own figures in admonitions.	"warning"
figure.warning	Figure to use to render a `warning` block. This parameter is added to allow new latex styles to use their own figures in admonitions.	"warning"
figure.caution	Figure to use to render a `caution` block. This parameter is added to allow new latex styles to use their own figures in admonitions.	"warning"

5.1.1 latex.hyperparam

This parameter gives the options to pass to the LaTeX hyperref package. No validity check is done.

For instance, the Table of Content rendering (link color, etc.) can be changed. Look at the hyperref.sty documentation to know all the hyperref options available.

Example 5.1.1 Configuring with latex.hyperparam

```
<?xml version='1.0' encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version='1.0'>

<!-- We want TOC links in the titles (not in the page numbers), and blue.
 -->
<xsl:param name="latex.hyperparam">colorlinks,linkcolor=blue</xsl:param>

</xsl:stylesheet>
```

5.2 Setting Parameter values

It is possible to set some XSL parameter values directly from the command line without creating a configuration parameter stylesheet, by using the `-P` *`parameter=value`* option.

The following example set the latex.hyperparam parameter value:

```
dblatex -P latex.hyperparam=colorlinks,linkcolor=blue myfile.xml
```

5.3 Customized stylesheets

If one needs to change some of the translations done by the XSL stylesheets, it is possible to provide user stylesheets to override the templates. To do this, write the stylesheets (e.g. mystyle.xsl) and include them in the configuration file such as shown by the following example.

Example 5.3.1 Using a customized stylesheet in a configuration file

```
<?xml version='1.0' encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version='1.0'>

<!-- Let's import our own XSL to override the default behaviour.
 -->
<xsl:import href="mystyle.xsl"/>

</xsl:stylesheet>
```

5.4 Customized LaTeX style

The actual output rendering is done by the latex style package used, and not by the XSL stylesheets whose role is only to translate to latex. Users can provide their own LaTeX style file, in respect of some rules:

- The LaTeX style package preamble must support all the options that the XSL stylesheets can pass to the package.
- Some packages must be used to make all the thing work.
- The docbook interface must be defined: the XSL stylesheets register some elements information in LaTeX commands. These commands or macro are the only ones specific to DocBook that are explicitely used by the XSL stylesheets. Other specific macros are used but are not intended to be changed by the user. These hidden macros are defined in the dbk_core latex package.

The latex style file to use is specified by using the option `--style` *`latex_style`*. An example of a simple LaTeX DocBook style is provided in the package.

5.4.1 Package options

A compliant LaTeX style package supports the following options. The options are provided by the XSL stylesheets according to the document attributes.

Option	Role
hyperlink, nohyperlink	Indicates if links in the document are provided or not
article, book	The document is an `article` or a `book`

5.4.2 Needed packages

A LaTeX style package must at least include the following packages.

Package	Description
dbk_core	Core LaTeX definitions and macros needed for DocBook

5.4.3 DocBook interface

All the latex commands beginning with DBK are related to elements under `bookinfo` or `articleinfo`.

Command	Description
\DBKreference	mapped to `pubsnumber`
\DBKsite	mapped to `address`
\DBKcopyright	mapped to `copyright`
\DBKdate	mapped to `date`
\DBKedition	mapped to `edition`
\DBKpubdate	mapped to `pubdate`
\DBKsubtitle	mapped to `subtitle`
\DBKreleaseinfo	mapped to `releaseinfo`
\DBKlegalnotice	environment mapped to a `legalnotice`. The legal notices are all put into the \DBKlegalblock command. It is up to the latex stylesheet to decide where to put it in the document.
\DBKlegalblock	wrapper command for the \DBKlegalnotice environments, used by the latex stylesheet to decide where to put the legal notices in the document.
\DBKindexation	This command contains the `othercredit` information translated to latex by the XSL. This command must be placed where the othercredit shall appear in the document.
\DBKindtable	This environnement must be defined by the user to render the `othercredit` list. It can be displayed as a table, listitem, description list, or anything that suits your need.
\DBKinditem	This is an `othercredit` item.
\DBKrevtable	This environnement must be defined by the user to render the `revhistory` table. Untill now it is not really possible to customize it, since it must be a table with four columns, each column for a `revhistory` piece of information.
float example	This float is expected to be defined, and is mapped to `example`. It is not defined by default by the dbk_core package to allow the user to define its rendering (ruled or not, etc.)
float dbequation	This float is expected to be defined, and is mapped to `equation`. It is not defined by default by the dbk_core package to allow the user to define its rendering (ruled or not, etc.)

5.4.4 Debugging your Style

It is not surprising if your first dblatex compilation fails with a fresh LaTeX style. So, how to debug it when used with dblatex?

The following steps can help you:

1. Compile your file in the debug mode (option `-d`). When the compilation is done, the temporary working directory will not be removed.

```
$ dblatex --style mytexstyle -d file.xml
```

```
...
/tmp/tpub-ben-99629 is not removed
```

2. Go under the building temporary directory, and set the environment with the file `env_tex`.

```
$ cd /tmp/tpub-ben-99629
$ . env_tex
```

3. Compile the temporary latex file produced by the XSL stylesheets. Its name has the suffix "_tmp.tex".

```
$ pdflatex file_tmp.tex
$ [ many outputs here ]
```

4. Now latex stops when it encounters an error so that you can debug your stylesheet.

5.5 Latex post process script

Extra user actions can be processed on the latex file produced by the XSL stylesheets or on its temporary working files produced by the latex compilation.

For instance, in the documents I write the cover page must display the number of pages of the document, but written in full letters (e.g. 23 is written "twenty three"). The latex post process script is then helpfull, and in this particular case it patches the .aux file.

The post process script is called just before the last latex compilation, and takes one parameter, the latex file compiled by the tool.

5.6 Specification file

A master configuration file, also called a specification file, can be defined to list all the customizations and options to apply. Such a file is passed by using the option `-S` *`specs`*.

The format of the file is the following:

- Every comment starts with a "#", and is ignored.
- The file must contain one parameter by line.
- The format of a parameter is the following:

```
<keyword>: <value>
```

- Every parameter is mapped to an option that can be passed to **dblatex**.
- An unknown parameter is silently ignored (the whole line is dropped).
- The parameters defining a path (a file or a directory) can take absolute or relative paths. A relative path must be defined from the specification file itself. For instance, a specification file under `/the/spec/directory/` with a parameter describing the file `../where/this/file/is/myfile` points to `/the/spec/where/this/file/is/myfile`.

The following table lists the supported parameters and the corresponding command line option.

Keyword	Value	Corresponding option	Description
TexInputs	Directories	--texinputs	Defines extra path to add to TEXINPUTS
PdfInputs	Directories	--pdfinputs	Defines paths containing PDF graphics needed to compile the latex file with pdflatex.

Keyword	Value	Corresponding option	Description
TexStyle	Latex package name	--style	Defines the LaTeX style package to use.
TexPost	Script file name	--texpost	Defines the LaTeX post process script to use.
XslParam	Parameter file name	-p	Defines the parameter file to use.
FigInputs	Directories	-I	Defines the extra figures path.
Options	Command line options	None	Lists command options to use by default when using the tool. The options specified by the parameter are directly passed to **dblatex**

Here is the specification file used for this manual.

Example 5.6.1 Specification file example

```
#
# Specification file example
#
TexInputs: ../latex//
PdfInputs: ../latex/graphics
TexStyle:  docbook
TexPost:   ../latex/example/postlatex
XslParam:  myparam.xsl
Options:   -b pdftex
```

5.7 Customization order

All the customization queries are translated to the corresponding command line options. Thus, using several customization methods can be unconsistent because each of them override the same option with another value.

For instance, you can specify the use of a specification file in which it is said to use a latex style (parameter TexStyle) and explicitely use the `--style` command line option. So, what is the behaviour?

The options order is the following:

- If a specification file is used (`-S` option), the options are set to the specification file parameters.
- The options explicitely passed override the specification file setting, whatever is the position of the options (i.e. before or after the `-S` option).
- If an option is passed several times, this is the last occurence that is used.

Example 5.7.1 Customization order

Let’s consider the specification file containing the following parameters:

```
XslParam: file3.xsl
Options: -b pdftex
TexStyle: mystyle1
```

And now the command line:

```
dblatex -b dvips -p file1.xsl -p file2.xsl -S file.specs --style mystyle2 mydoc.sgml
```

The setting used is the following:

- “-b dvips” overrides “-b pdftex” set by the spec file.
- “-p file2.xsl” overrides “-p file1.xsl” since it is defined after, and overrides “file3.xsl” set by the spec file.
- “--style mystyle2” override “mystyle1” set by the spec file.

Chapter 6

Thanks

Thanks to the people who contribute to the project: Jean-Yves Le Ruyet, precursory and hard-working user, Julien Ducourthial for his precious help, and all the users for their remarks to improve the product, like the KDE documentation team, or Vincent Hottier who asked for the embedded LaTeX equations support.