DocBook V5.0

The Transition Guide

28 October 2007

This version:
http://docbook.org/docs/howto/2007-10-28/
Latest version:
http://docbook.org/docs/howto/
Previous versions:
http://docbook.org/docs/howto/2006-10-22/
http://docbook.org/docs/howto/2006-05-16/
http://docbook.org/docs/howto/2006-03-01/
Authors and other credited contributors:
Jirka Kosek, <jirka@kosek.cz>
Norman Walsh, <ndw@nwalsh.com>
Dick Hamilton, <rlhamilton@frii.com>
Michael(tm) Smith, <smith@sideshowbarker.net> (contributor)

Table of Contents

Introduction 2
Finally in a namespace 2
Relaxing with DocBook 2
Why switch to DocBook V5.0? 3
Schema jungle 4
Tool chain 5
Editing DocBook V5.0 5
Validating DocBook V5.0 10
Processing DocBook V5.0 11
Markup changes 13
Improved cross-referencing and linking 13
Renamed elements 14
Removed elements 15
Converting DocBook V4.x documents to DocBook V5.0 15
What About Entities? 16
Customizing DocBook V5.0 17
DocBook RELAX NG schema organization 17
General customization considerations 18
Elements 18
Attributes 23
Naming and versioning DocBook customizations 25
FAQ 26
Bibliography 31

This document is targeted at DocBook users who are considering switching from DocBook V4.x to DocBook V5.0. It describes differences between DocBook V4.x and V5.0 and provides some suggestions about how to edit and process

DocBook V5.0 documents. There is also section devoted to conversion of legacy documents from DocBook 4.x to DocBook V5.0.

At the time of this writing the current version of DocBook V5.0 was 5.0CR7. However almost all information in this document is general and it is applicable to any newer version in DocBook V5.0 series.

Introduction

The differences between DocBook V4.x and V5.0 are quite radical in some aspects, but the basic idea behind DocBook is still the same and almost all element names are unchanged. Because of this it is very easy to become familiar with DocBook V5.0 if you know any previous version of DocBook. You can find a complete list of changes in [DB5SPEC], here we will discuss only the most fundamental changes.

Finally in a namespace

All DocBook V5.0 elements are in the namespace `http://docbook.org/ns/docbook`. XML namespaces are used to distinguish between different element sets. In the last few years, almost all new XML grammars have used their own namespace. It is easy to create compound documents that contain elements from different XML vocabularies. DocBook V5.0 is following this design rule. Using namespaces in your documents is very easy. Consider this simple article marked up in DocBook V4.5:

```
<article>
  <title>Sample article</title>
  <para>This is a really short article.</para>
</article>
```

The corresponding DocBook V5.0 article will look very similar:

```
<article xmlns="http://docbook.org/ns/docbook" …>
  <title>Sample article</title>
  <para>This is a really short article.</para>
</article>
```

The only change is the addition of a default namespace declaration (`xmlns="http://docbook.org/ns/docbook"`) on the root element. This declaration applies the namespace to the root element and all nested elements. Each element is now uniquely identified by its local name and namespace.

Note

The namespace name `http://docbook.org/ns/docbook` serves only as an identifier. This resource is not fetched during processing of DocBook documents and you are not required to have an Internet connection during processing. If you access the namespace URI with a browser, you will find a short explanatory document about the namespace. In the future this document will probably conform to (some version of) RDDL and provide pointers to related resources.

Relaxing with DocBook

For more than a decade, the DocBook schema was defined using a DTD. However DTDs have serious limitations and DocBook V5.0 is thus defined using a very powerful schema language called RELAX NG. Thanks to RELAX NG, it is now much easier to create customized versions of DocBook, and some content models are now cleaner and more precise.

Using RELAX NG has an impact on the document prolog. The following example shows the typical prolog of a DocBook V4.x document. The version of the DocBook DTD (in this case 4.5) is indicated in the document type declaration (!DOCTYPE) which points to a particular version of the DTD.

Example 1. DocBook V4.5 document

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE article PUBLIC '-//OASIS//DTD DocBook XML V4.5//EN'
                         'http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd'>
<article lang="en">
  <title>Sample article</title>
  <para>This is really very short article.</para>
</article>
```

In contrast, DocBook V5.0 does not depend on DTDs anymore. This mean that there is no document type declaration and the version of DocBook used is indicated with the `version` attribute instead.

Example 2. DocBook V5.0 document

```
<?xml version="1.0" encoding="utf-8"?>
<article xmlns="http://docbook.org/ns/docbook" version="5.0" xml:lang="en">
  <title>Sample article</title>
  <para>This is really very short article.</para>
</article>
```

As you can see, DocBook V5.0 is built on top of existing XML standards as much as possible, for example the `lang` attribute is superseded by the standard `xml:lang` attribute.

Another fundamental change is that there is no direct indication of the schema used. Later in this document, you will learn how you can specify a schema to be used for document validation.

Note

Although we recommend the RELAX NG schema for DocBook V5.0, there are also DTD and W3C XML Schema versions available (see the section called “Where to get the schemas”) for tools that do not yet support RELAX NG.

Why switch to DocBook V5.0?

The simple answer is “because DocBook V5.0 is the future”. Apart from this marketing blurb, there are also more technical reasons:

- *DocBook V4.x is feature frozen.* At the time of this writing DocBook V4.5 is the last version of DocBook in the V4.x series. Any new DocBook development, like the addition of new elements, will be done in DocBook V5.0. It is only matter of time before useful, new elements will be added into DocBook V5.0, but they are not likely to be back ported into DocBook V4.x. DocBook V4.x will be in maintenance mode and errata will be published if necessary.

- *DocBook V5.0 offers new functionality.* Even the current version of DocBook V5.0 provides significant improvements over DocBook V4.x. For example there is general markup for annotations, a new and flexible system for linking, and unified markup for information sections using the `info` element.

- *DocBook V5.0 is more extensible.* Having DocBook V5.0 in a separate namespace allows you to easily mix DocBook markup with other XML based languages like SVG, MathML, XHTML or even FooBarML.

- *DocBook V5.0 is easier to customize.* RELAX NG offers many powerful constructs that make customization much easier than it would be using a DTD.

Schema jungle

Schemas for DocBook V5.0 are available in several formats at http://www.oasis-open.org/docbook/xml/5.0CR7/ (or the mirror at http://docbook.org/xml/5.0CR7/). Only the RELAX NG schema is normative and it is preferred over the other schema languages. However, for your convenience there are also DTD and W3C XML Schema versions provided for DocBook V5.0. But please note that neither DTDs nor XML schemas are able to capture all the constraints of DocBook V5.0. This mean that a document that validates against the DTD or XML schema is not necessarily valid against the RELAX NG schema and thus may not be a valid DocBook V5.0 document.

DTD and W3C XML Schema versions of the DocBook V5.0 grammar are provided as a convenience for users who want to use DocBook V5.0 with legacy tools that don't support RELAX NG. Authors are encouraged to switch to RELAX NG based tools as soon as possible, or at least to validate documents against the RELAX NG schema before further processing.

Some document constrainst can't be expressed in schema languages like RELAX NG or W3C XML Schema. To check for these additional constraints DocBook V5.0 uses Schematron. It is recommended to validate your document not only against RELAX NG schema but also against Schematron schema.

Where to get the schemas

The latest versions of schemas can be obtained from http://docbook.org/schemas/5x.html. At the time of this writing the latest version is 5.0CR7 and individual schemas are available at the following locations:

RELAX NG schema	http://docbook.org/xml/5.0CR7/rng/docbook.rng
RELAX NG schema in compact syntax	http://docbook.org/xml/5.0CR7/rng/docbook.rnc
DTD	http://docbook.org/xml/5.0CR7/dtd/docbook.dtd
W3C XML Schema	http://docbook.org/xml/5.0CR7/xsd/docbook.xsd
Schematron schema with additional checks	http://docbook.org/xml/5.0CR7/sch/docbook.sch

These schemas are also available from the mirror at http://www.oasis-open.org/docbook/xml/5.0CR7/.

DocBook documentation

Detailed documentation about each DocBook V5.0 element is presented in the reference part of *DocBook: The Definitive Guide*[1].

Note

Other parts of the book have not yet been updated to reflect the changes made in DocBook V5.0. Please do not be confused by this.

[1] http://docbook.org/tdg5/en/html/pt02.html

Tool chain

This section briefly describes tools and procedures to edit and process content stored in DocBook V5.0.

Editing DocBook V5.0

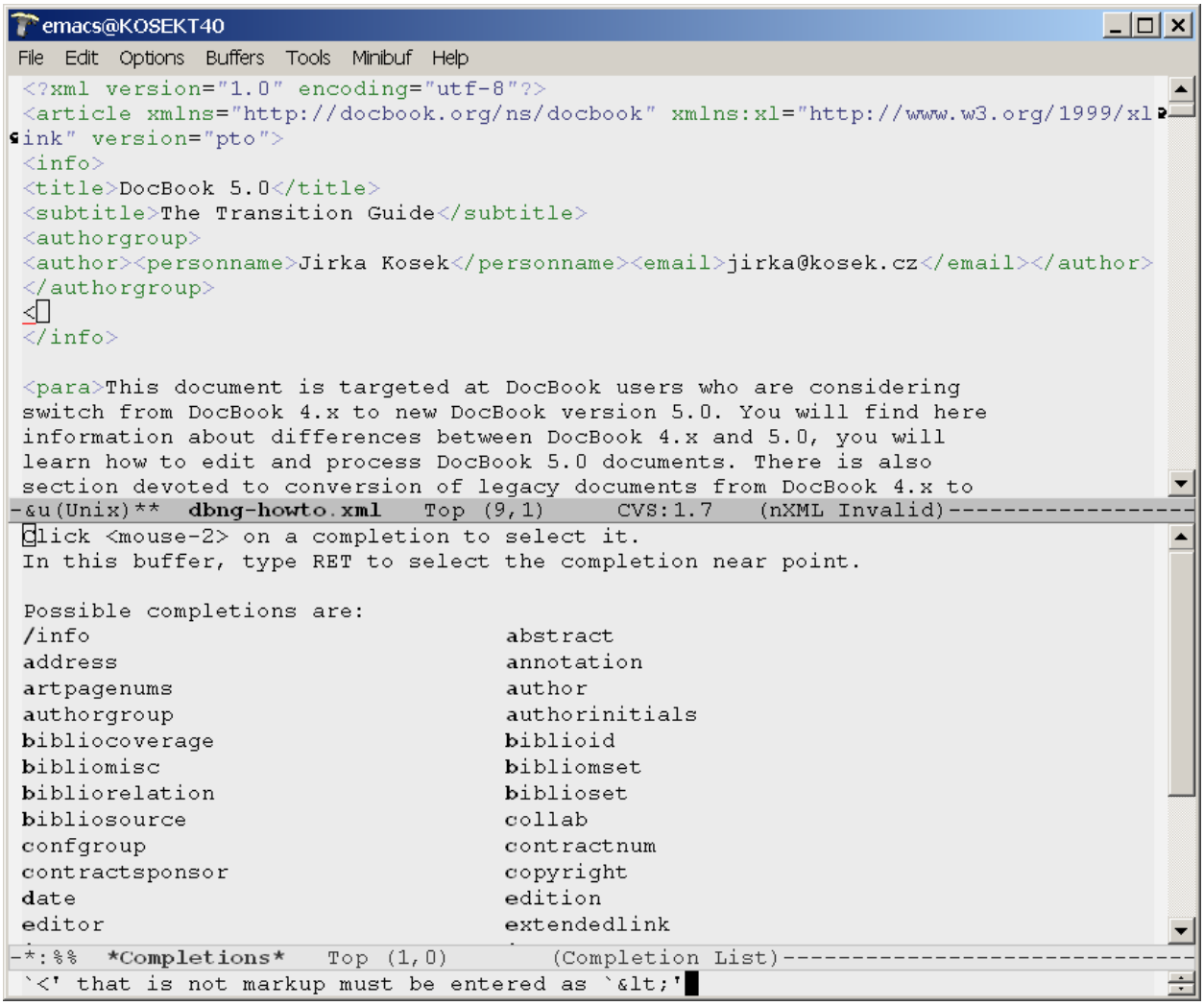
Because DocBook is an XML based format and XML is a text based format, you can use any text editor to create and edit DocBook V5.0 documents. However using “dumb” editors like Notepad is not very productive. You will do better if you use an editor that supports XML. Although there are DTD and W3C XML Schemas available for DocBook V5.0, which means you can use any editor that works with DTDs or W3C XML Schemas, we recommend that you use the RELAX NG grammar with DocBook V5.0. The rest of this section contains an overview of XML editors (listed in alphabetical order) that are known to work with RELAX NG schemas and that offer guided editing based on the RELAX NG schema.

Emacs and nXML

nXML mode[2] is an add-on for the GNU Emacs[3] text editor. By installing nXML you can turn Emacs into a very powerful XML editor that offers guided editing and validation of XML documents.

[2] http://www.thaiopensource.com/nxml-mode/
[3] http://www.gnu.org/software/emacs/emacs.html

Figure 1. Emacs with nXML mode provides guided editing and validation



nXML uses a special configuration file named `schemas.xml` to associate schemas with XML documents. Often you will find this file in the directory `site-lisp/nxml/schema` inside the Emacs installation directory. Adding the following line into the configuration file, will associate DocBook V5.0 elements with the appropriate schema:

```
<namespace ns="http://docbook.org/ns/docbook" uri="/path/to/docbook.rnc"/>
```

Note

Please note that nXML ships with a file named `docbook.rnc`. This file contains the RELAX NG grammar for DocBook V4.x. Be sure that you associate the DocBook V5.0 namespace with the corresponding DocBook V5.0 grammar.

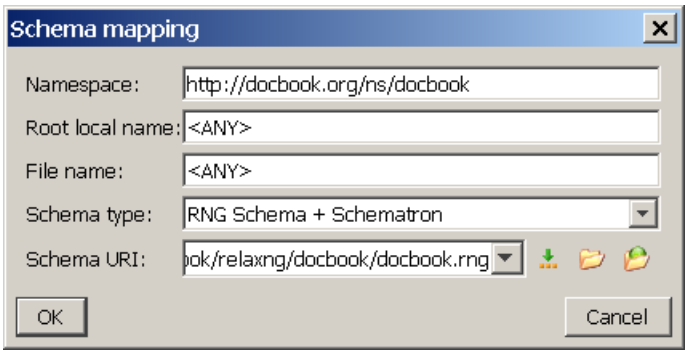
If you can't edit the global `schemas.xml` file, you can create this file in a directory with your document. nXML will find associations placed there also. In this case you must create a complete configuration file like:

```
<locatingRules xmlns="http://thaiopensource.com/ns/locating-rules/1.0">
  <namespace ns="http://docbook.org/ns/docbook" uri="/path/to/docbook.rnc"/>
</locatingRules>
```

oXygen

oXygen[4] is a feature rich XML editor. It has built-in support for many schema languages including RELAX NG. If you want to smoothly edit and validate DocBook 5.0 documents you should associate the DocBook namespace with the corresponding schema. Go to Options → Preferences… → Editor → Default Schema Associations. Then click the New button to add a new association. Type in the DocBook namespace and the RELAX NG schema location, choose the RNG Schema + Schematron type of schema as, and confirm your choice by clicking the OK button.

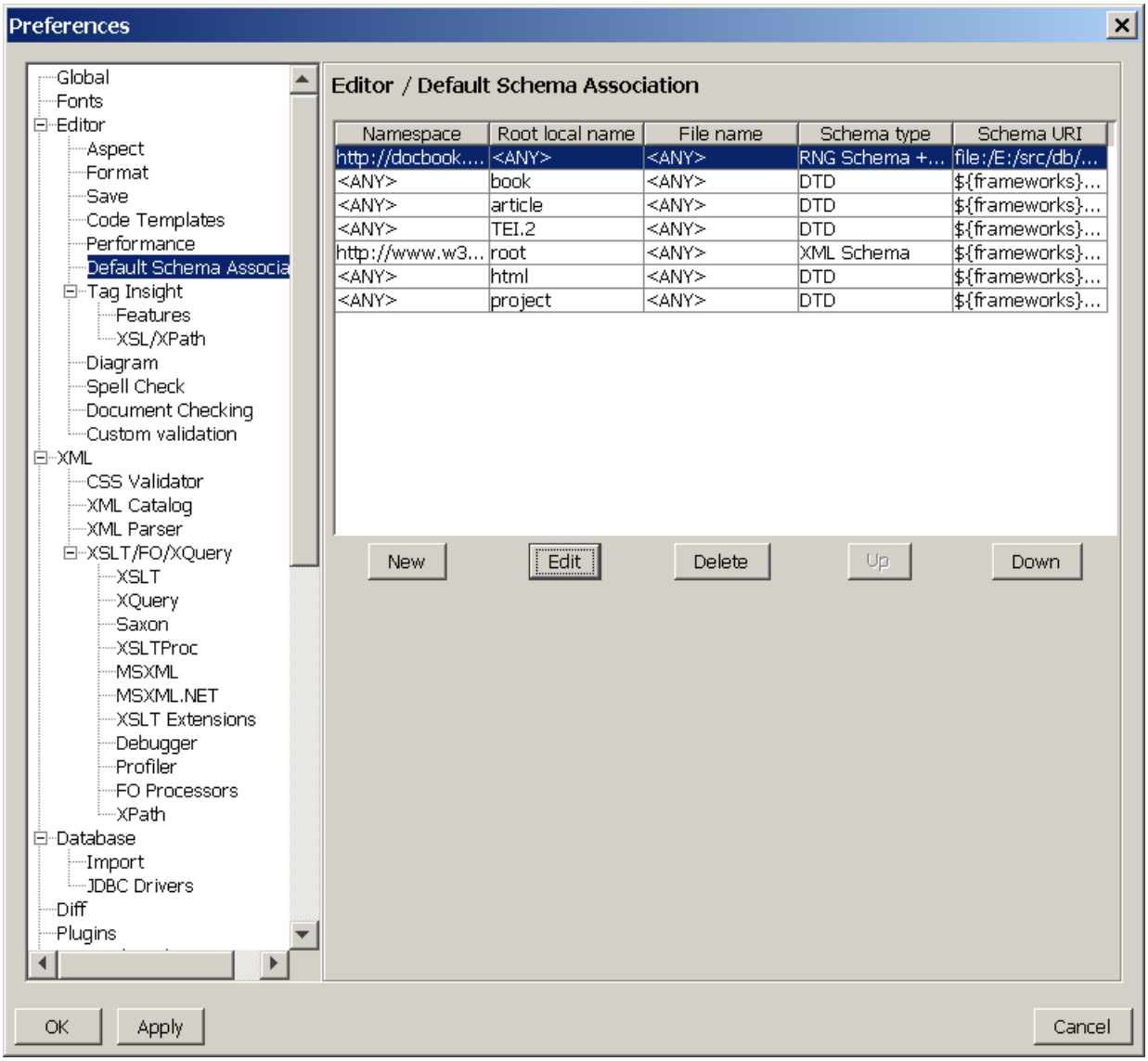
Figure 2. Adding a new schema association in oXygen



Because oXygen comes with preconfigured associations for DocBook V4.x, you must move your newly added configuration to the top of the list (using the Up button). That way you will be able to use oXygen with both DocBook V4.x and DocBook V5.0.

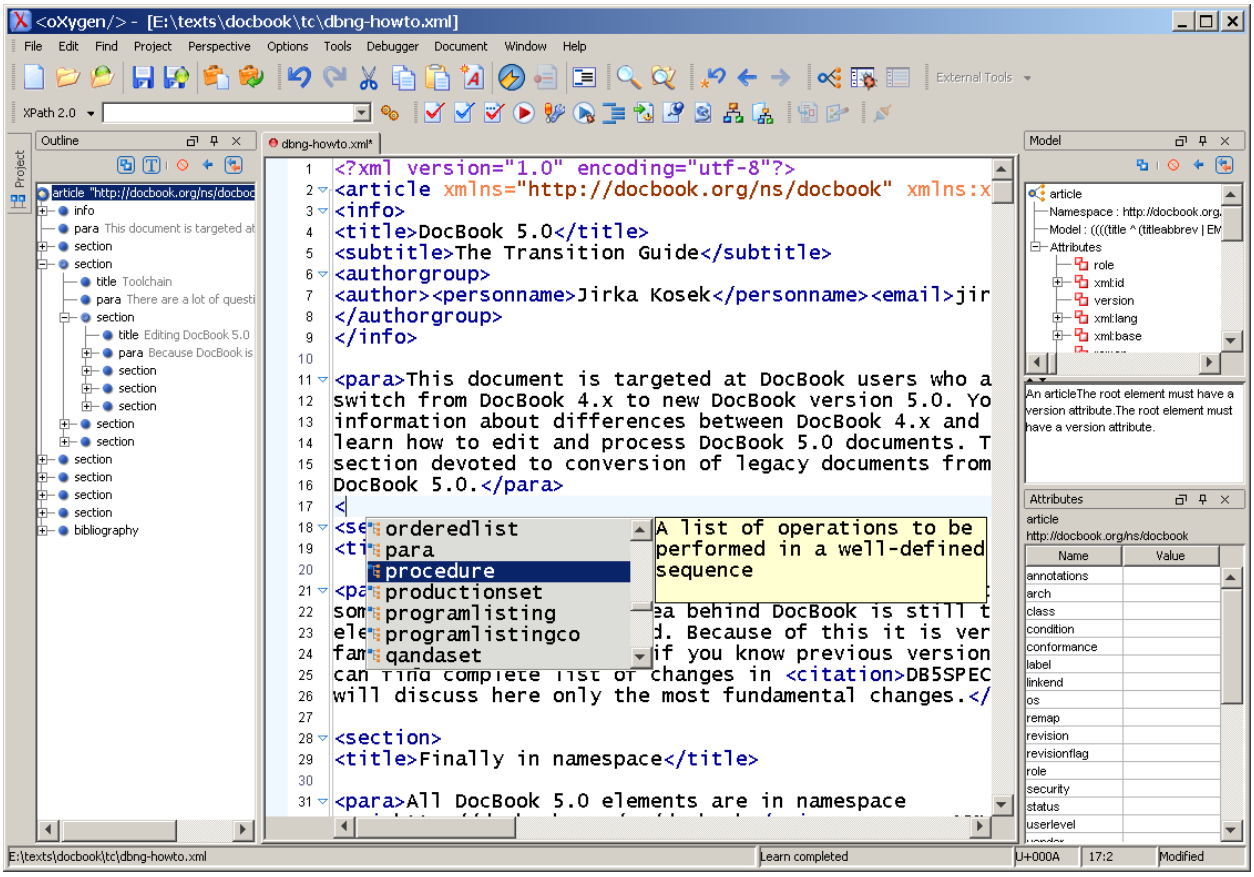
[4] http://www.oxygenxml.com/

Figure 3. DocBook V5.0 association must precede associations for DocBook V4.x



Now you can close the preference box by clicking on the OK button. From this time oXygen will assist you with writing DocBook V5.0 content and you will be able to validate your documents against both RELAX NG and Schematron schemas.

Figure 4. DocBook V5.0 document opened in oXygen

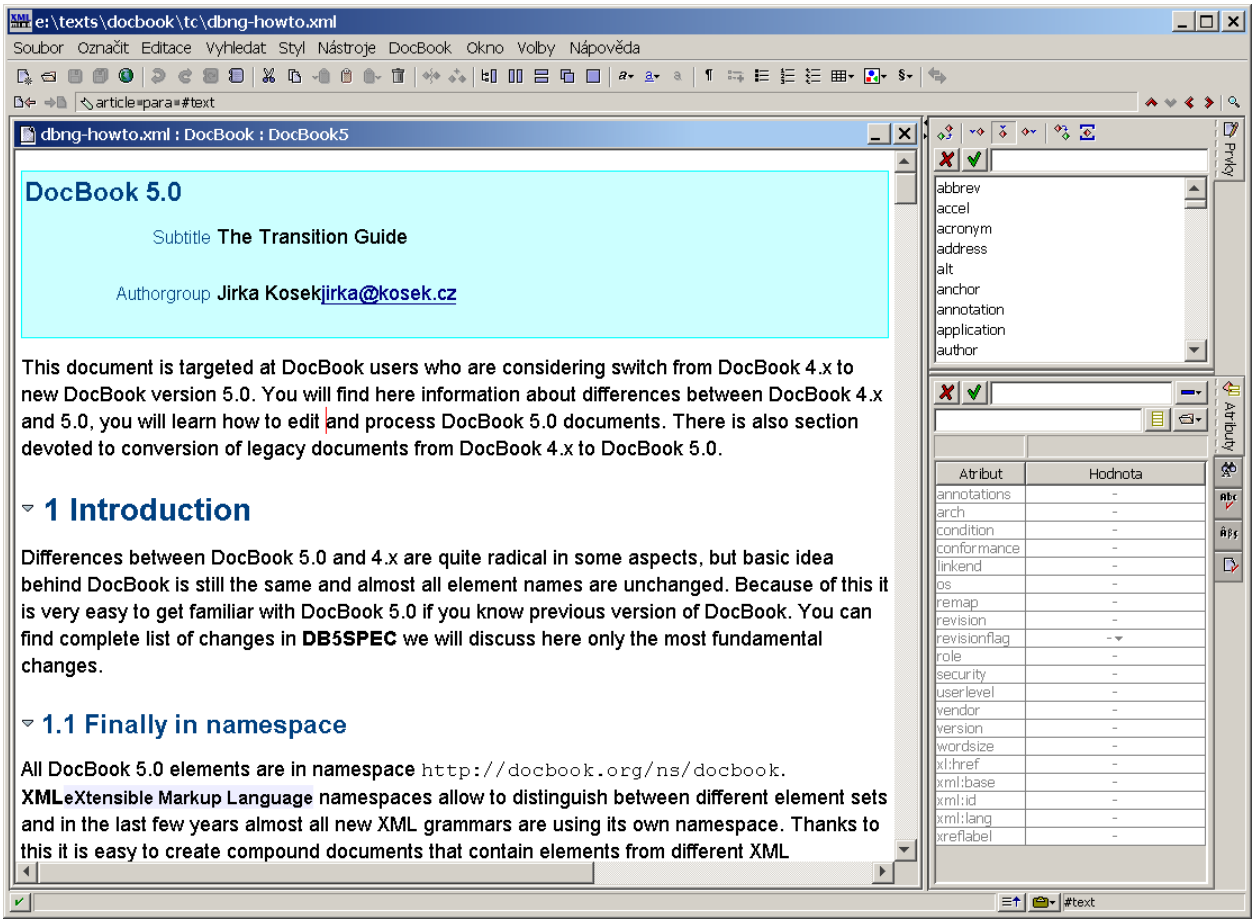


XML Mind XML editor

XML Mind XML editor[5] (XXE) is a visual validating XML editor that provides a wordprocessor-like interface to users. It is available in two versions, Standard and Professional. The Standard version is free and provides everything you need to edit DocBook V5.0 documents.

[5] http://www.xmlmind.com/xmleditor/

Figure 5. XML Mind XML Editor – feels almost like MS Word but real DocBook V5.0 markup is created



In order to use DocBook V5.0 in XXE you have to install correspoding add-on. Go to Options → Install Add-ons…. Then choose DocBook 5 configuration and press OK button. After restart XXE is ready to work with DocBook V5.0 documents.

Validating DocBook V5.0

If you are not using a RELAX NG based validating editor when you create documents, we strongly recommend that you validate your documents against RELAX NG and Schematron schemas before processing them. Only after successful validation you can be sure that your document is really DocBook V5.0 and that processing tools will be able to process it correctly.

For validation you can use tools that support simultaneous RELAX NG and Schematon validation or you can use NVDL to orchestrate validation to those two schemas.

Using RELAX NG and Schematron

You can find a list of RELAX NG validators at http://relaxng.org/#validators. It is best to use validators with support for embedded Schematron rules inside RELAX NG schemas. Schematron is a rule-based validation language which is used to impose additional constraints on DocBook documents. Schematron rules assert conditions which cannot be expressed in a pure RELAX NG schema.

Sun Multi-Schema XML Validator (MSV)[6] is able to validate an XML document against a RELAX NG schema and Schematron rules at the same time. To install and use MSV follow these steps:

1. Download `relames.zip` from https://msv.dev.java.net/servlets/ProjectDocumentList?folderID=101.
2. Unpack the downloaded file into an arbitrary directory.
3. Validate your document using the following command:

   ```
   java -Xss512K -jar /path/to/relames.jar /path/to/docbook.rng document.xml
   ```

   Note

   The switch `-Xss512K` increases the stack size of the Java virtual machine. This is necessary because the DocBook schema is quite large. If you get stack overflow errors from MSV, increase this value. You may get spurious error messages if the value is too small, so if you get a stack overflow error, ignore other error messages and try a larger value for the stack size. If you are not using Sun's Java implementation, please consult the documentation for your virtual machine to learn how to increase the stack size.

There is also an on-line DocBook V5.0 validator[7] that validates DocBook V5.0 documents against the normative RELAX NG schema with embedded Schematron rules.

Using NVDL

NVDL is a metaschema language which can send one document to validation against several schemas. DocBook V5.0 comes with NVDL schema which defines that DocBook documents should be validated against both RELAX NG and Schematron schema.

You can find a list of NVDL validators at http://nvdl.org/. The following procedure shows how to install and use oNVDL[8] validator.

1. Download `onvdl-20070517.zip`[9].
2. Unpack the downloaded file into an arbitrary directory.
3. Validate your document using the following command:

   ```
   java -jar /path/to/oNVDL/bin/onvdl.jar /path/to/docbook.nvdl document.xml
   ```

Processing DocBook V5.0

Part of DocBook's great success can be attributed to the availability of free tools that can be used to transform DocBook content into various target formats including HTML and PDF. The DocBook XSL Stylesheets are very popular tools.

DocBook XSL Stylesheets

The DocBook stylesheets are designed to process content written in different versions of DocBook (for example 3.1 and 4.2). Recent versions of the stylesheets are also able to process DocBook V5.0 with some limitations.

[6] https://msv.dev.java.net/
[7] http://relaxed.vse.cz/docbookvalidator/
[8] http://www.oxygenxml.com/onvdl.html
[9] http://www.oxygenxml.com/InstData/onvdl/onvdl-20070517.zip

You can process DocBook V5.0 documents with the DocBook XSL stylesheets exactly the same way as you process DocBook V4.x documents. You do not need special software, you can stick to your preferred XSLT processor, be it Saxon, xsltproc, Xalan or whatever else.

During document processing, the stylesheets strip namespaces from DocBook V5.0 to get a document which will be very similar to DocBook V4.x. This is necessary because from the XSLT point of view elements from different namespaces are distinct and can not be easily processed by the same set of templates. This process is completely transparent to the user. If you are processing DocBook V5.0 documents, the only difference is that you will see the following additional message:

```
Note: namesp. cut : stripped namespace before processing
Note: namesp. cut : processing stripped document
```

Although you can successfully use the existing stylesheets to process DocBook V5.0, there are some limitations and unsupported features. The unsupported features include:

- general annotations;
- general XLink links on all elements.

Note

During namespace stripping, the base URI of the document is lost. This means that in rare situations, relatively referenced resources like images or programlistings can be processed incorrectly. The stylesheets attempt to compensate for this problem, but it is possible that there are corner cases where they will fail.

If you want to use HTML Help, JavaHelp or Eclipse stylesheets with DocBook V5.0 you have to use profiling variants of those stylesheets (their name is starting with `profile-`).

DocBook XSL-NS Stylesheets

As you can see from reading the previous section, namespace stripping has limitations that will cause trouble in some situations. To overcome those limitations, Bob Stayton created a build system for taking the non-namespace-aware DocBook XSL stylesheets and generating namespace-aware versions from them. The DocBook XSL-NS stylesheets[10] are the result.

The DocBook XSL-NS stylesheets are released side-by-side with the DocBook XSL stylesheets, as a separate docbook-xsl-ns[11] package. They are the recommended XSLT 1.0 stylesheets to use for transforming namespaced (DocBook 5) documents.

XSLT 2.0 based re-implementation

XSLT 1.0 is missing some important features. To work around these missing features, the current DocBook XSL stylesheets use some implementation-specific extensions. XSLT 2.0 adds many new and previously missing features into the language. A new set of DocBook stylesheets is being implemented based on XSLT 2.0 to take advantage of these features and to fully support DocBook V5.0.

The XSLT 2.0 based stylesheets have many new features, including:

- seamless integration of profiling (conditional documents) with external bibliographies and glossaries;
- no need for (most) external extensions;

[10] http://docbook.sourceforge.net/release/xsl-ns/current/
[11] https://sourceforge.net/project/showfiles.php?group_id=21935&package_id=219178

- internationalized indexes;
- easy to customize titlepage templates;

The XSLT 2.0 based stylesheets are still under development. At this writing, they only support HTML and chunked HTML output. As time permits, the stylesheet developers will be adding other formats. Since the stylesheets are developed in the limited free time the developers have, there's no specific schedule.

There are not very many XSLT 2.0 implementations available. But, if you want to try the new stylesheets, grab a snapshot of the development version from http://docbook.sourceforge.net/snapshots/docbook-xsl2-snapshot.zip and unpack it somewhere. Then download and install Saxon 8 from http://saxon.sf.net.

To transform a DocBook V5.0 document to a single HTML page use the command:

```
java -jar /path/to/saxon8.jar -o output.html document.xml /path/to/docbook-xsl2-snapshot/html/docbook.xsl
```

To transform a DocBook V5.0 document to a set of chunked HTML pages use the command:

```
java -jar /path/to/saxon8.jar document.xml /path/to/docbook-xsl2-snapshot/html/chunk.xsl
```

Markup changes

This section describes the most common markup changes between DocBook V4.x and V5.0. You can find a complete list of changes in [DB5SPEC].

Improved cross-referencing and linking

In DocBook V4.x the attribute `id` is used to assign a unique identifier to an element. In DocBook V5.0 this attribute is renamed `xml:id` in order to comply with [XMLID].

Now you can use almost any inline element as the source of a link, not just `xref` or `link`. For example, the following DocBook 4.x content:

```
<section id="dir">
  <title>DIR command</title>
  <para>...</para>
</section>

<section id="ls">
  <title>LS command</title>
  <para>This command is a synonym for <link linkend="dir"><command>DIR</command></link> command.</para>
</section>
```

is written in DocBook V5.0 as:

```
<section xml:id="dir">
  <title>DIR command</title>
  <para>...</para>
</section>

<section xml:id="ls">
  <title>LS command</title>
  <para>This command is a synonym for <command linkend="dir">DIR</command> command.</para>
</section>
```

The `linkend` attribute was added to all inline elements together with the `href` attribute from the XLink namespace. This means that you can use any inline element as the source of a hypertext link. To use XLinks you have to declare the XLink namespace (most often on the root element of your document):

```
<article xmlns="http://docbook.org/ns/docbook"
         xmlns:xl="http://www.w3.org/1999/xlink" version="5.0">
  <title>Test article</title>

  <para><application xl:href="http://www.gnu.org/software/emacs/emacs.html">Emacs</application>
    is my favourite text editor.</para>
  …
```

The `ulink` element was removed from DocBook V5.0 in favor of XLink linking. Instead of the DocBook V4.x `ulink` element:

```
<ulink url="http://docbook.org">DocBook site</ulink>
```

you can now use `link`

```
<link xl:href="http://docbook.org">DocBook site</link>
```

XLink links may contain a fragment identifier, which you can use instead of `linkend` to form cross-references inside a document; for example:

```
<command xl:href="#dir">DIR</command>
```

However XLink links are not checked during validation, while `xml:id`/`linkend` links are checked for ID/IDREF consistency. One place where the XLink-based, fragment identifier scheme is useful is when XInclude is being used, since XML ID/IDREF links cannot span XInclude boundaries. You can use whichever approach better suits your needs.

Renamed elements

Some elements were renamed to better express their meaning or to reduce the total number of elements available in DocBook.

Table 1. Renamed elements

Old name	New name
`sgmltag`	`tag`
`bookinfo`, `articleinfo`, `chapterinfo`, `*info`	`info`
`authorblurb`	`personblurb`
`collabname`, `corpauthor`, `corpcredit`, `corpname`	`orgname`
`isbn`, `issn`, `pubsnumber`	`biblioid`
`lot`, `lotentry`, `tocback`, `tocchap`, `tocfront`, `toclevel1`, `toclevel2`, `toclevel3`, `toclevel4`, `toclevel5`, `tocpart`	`tocdiv`
`graphic`, `graphicco`, `inlinegraphic`, `mediaobjectco`	`mediaobject` and `inlinemediaobject`
`ulink`	`link`
`ackno`	`acknowledgements`

Removed elements

The following elements were removed from DocBook V5.0 without direct replacements: `action`, `beginpage`, `highlights`, `interface`, `invpartnumber`, `medialabel`, `modespec`, `structfield`, `structname`. If you use one or more of these elements, here are some suggestions as to how to re-code them in DocBook V5.0.

Table 2. Recommended mapping for removed elements

Old name	Recommended mapping
`action`	Use `<phrase remap="action">`.
`beginpage`	Remove: `beginpage` is advisory only and has tended to cause confusion. A processing instruction or comment should be a workable replacement if one is needed.
`highlights`	Use `abstract`. Note that because `highlights` has a broader content model, you may need to wrap contents in a `para` inside `abstract`.
`interface`	Use one of the "gui*" elements (`guibutton`, `guiicon`, `guilabel`, `guimenu`, `guimenuitem`, or `guisubmenu`).
`invpartnumber`	Use `<biblioid class="other" otherclass="medialabel">`. The `productnumber` element is another alternative.
`medialabel`	Use `<citetitle pubwork="`*`mediatype`*`">`, where *`mediatype`* is the type of media being labeled (e.g.,`cdrom` or `dvd`).
`modespec`	No longer needed. The current processing model for `olink` renders `modespec` unnecessary.
`structfield`, `structname`	Use `varname`. If you need to distinguish between the two, use `<varname remap="`*`structname or structfield`*`">`. In some contexts, it may also be appropriate to use `property` for `structfield`.

Converting DocBook V4.x documents to DocBook V5.0

The DocBook V5.0 schema ships with an XSLT 1.0 stylesheet that is designed to transform valid DocBook V4.x documents to valid DocBook V5.0 documents.

To convert your document, `doc.xml` in the examples below, follow these steps:

1. Check the validity of your DocBook XML V4.x document. The conversion tool assumes that the input document is valid. If the input document contains markup errors, the results will be unpredictable at best.

2. Transform `doc.xml` to `newdoc.xml` with the `db4-upgrade.xsl` stylesheet included in the DocBook V5.0 distribution that you are using.

3. Check the validity of your DocBook XML V5.0 document against the DocBook V5.0 RELAX NG grammar.

In the vast majority of cases, the resulting document should be valid and your conversion process is finished.

If the document is not valid, please report the problem. (Over time, we'll have more experience with the sorts of things that can go wrong and we'll update this document to reflect that experience.)

What About Entities?

Using XSLT to transform existing documents to DocBook V5.0 has one potential disadvantage: it removes all entity references from your document.

If preserving entities is an important aspect of your production work flow, you will have to engage in a semi-manual process to preserve them.

1. Open your existing document using your favorite editing tool. You must use a tool that *is not* XML-aware, or one that allows you to edit markup "in the raw".

2. Replace all occurrences of the entity references that you want to preserve with some unique string. For example, if you want to preserve "`&Product;`" references, you could replace them all with "`[[[Product]]]`" (assuming that the string "[[[Product]]]" doesn't occur anywhere else in your document).

3. Copy the document type declaration off of your document and save it some place. The document type declaration is everything from "`<!DOCTYPE`" to the closing "`]>`".

4. Perform the conversion described in the section called "Converting DocBook V4.x documents to DocBook V5.0".

5. Open the new document using your favorite editing tool. Replace all occurrences of the unique string you used to save the entity references with the corresponding entity references.

6. Paste the document type declaration that you saved onto the top of your new document.

7. Remove the external identifier (the `PUBLIC` and/or `SYSTEM` keywords) from the document type declaration. A document that begins:

   ```
   <!DOCTYPE book [
   <!ENTITY someEntity "some replacement text">
   ]>
   ```

   is perfectly well-formed. If you don't remove the references to the DTD, then your parser will likely try to validate against DocBook V4.0 and that's not going to work. Alternatively, you could refer to the DocBook V5.0 DTD.

Tip

Steps 2 and 5 from previous procedure can be automated using the cloak script[12] written by Michael Smith.

External Parsed Entities

External parsed entities, entities which load part of a document from another file, are a special case. These can often be replaced with XInclude elements.

The Perl script `db4-entities.pl`, also included in the DocBook V5.0 distribution attempts to perform this replacement for you. To use the script, perform the following steps:

1. Process your document with `db4-entities.pl`. The script expects a single filename and prints the XInclude version on standard output.

2. Process the XInclude version as described in the section called "Converting DocBook V4.x documents to DocBook V5.0".

[12] http://docbook.svn.sourceforge.net/viewvc/docbook/trunk/contrib/tools/cloak

Customizing DocBook V5.0

It's much easier to customize DocBook V5.0 than it was to customize earlier releases. This is partly because RELAX NG provides better support for modifications than DTDs and partly because the DocBook schema is designed to take full advantage of the capabilities RELAX NG provides. This section describes the organization of the RELAX NG schema for DocBook, methods and examples for adding, removing, and modifying elements and attributes, and conventions for naming and versioning DocBook customizations. It assumes some familiarity with RELAX NG. If you are unfamiliar with RELAX NG, you can find a tutorial introduction in [RNCTUT].

DocBook RELAX NG schema organization

The DocBook RELAX NG schema is highly modular, using named patterns extensively. Every element, attribute, attribute list, and enumeration has its own named pattern. In addition, there are named patterns for logical combinations of elements and attributes. These named patterns provide "hooks" into the schema that allow you to do a wide range of customization by simply redefining one or more of the named patterns.

An important design characteristic of the schema is that duplication is minimized. This is done through the use of named patterns for common groupings that can be re-used. For example, the `imagedata` and `videodata` elements each have an `align` attribute that takes the same set of enumerated values. Rather than repeating those values, a single pattern, `db.halign.enumeration` is referenced by the `db.videodata.align.enumeration` and `db.imagedata.align.enumeration` patterns, which are in turn referenced by the `db.videodata.align.attribute` and `db.imagedata.align.attribute` patterns. While this may seem like overkill, it allows a customizer to modify the allowed enumerations for these two attributes separately or together, or to completely re-define the allowed content of either or both, by redefining one or more of these named patterns.

Pattern Names

Because named patterns are used extensively, the RELAX NG schema uses several naming conventions. These are:

- Names have two or more parts, separated dots "."
- The first part of each name is the prefix "db"
- Each element has a named pattern in the form `db.`*`elementname`*. Elements that have different content models in different contexts will also have patterns in the form `db.`*`context.elementname`*. For example, `db.figure.info` defines the content model for the `info` element when it appears as a child of the `figure` element. *`Context`* may have several parts. For example, `db.cals.entrytbl.thead`.
- Most attributes have a named pattern in the form `db.`*`attributename`*`.attribute`. Attributes that have different content models in different contexts will also have patterns in the form `db.`*`context.attributename`*`.attribute`. For example, `db.olink.localinfo.attribute` defines the content model of the `localinfo` attribute when it appears in `olink`. There are a few attributes that do not have individual named patterns. For example, the effectivity attributes are grouped into `db.effectivity.attributes` and not identified separately.
- Each element has a named pattern for its attribute list in the form `db.`*`elementname`*`.attlist` that defines the list of attributes for that element. Elements that have different attribute lists in different contexts will also have patterns in the form `db.`*`context.elementname`*`.attlist` For example, `db.html.table.attlist` defines the attribute list for the html `table` element and `db.cals.table.attlist` defines the attribute list for a cals `table` element.
- Each attribute that has enumerated values has a named pattern in the form `db.`*`[context.]attributename`*`.enumeration`. If the enumeration for a particular attribute depends on context, optional context is provided. For example, `db.verbatim.continuation.enumeration` defines the enumeration values for the `continuation` attribute that is used in verbatim contexts like `screen`. Unlike elements and attributes, there is not necessarily a named pattern for enumerated attributes outside their context. For example, there is no `db.class.enumeration` because the `class` attribute has a broad and non-intersecting range of uses.
- There are several different groupings of elements and attributes. Here are the major ones:

inlines	Combinations of inline elements, for example, `db.error.inlines`, which contains `db.errorcode`, `db.errortext`, etc.
blocks	Combinations of block elements, for example, `db.verbatim.blocks`, which contains `db.programlisting`, `db.screen`, etc.
attributes	Combinations of attributes, for example, `db.effectivity.attributes`, which contains the attributes `arch`, `condition`, `conformance`, etc.
components	High level components of the schema, for example, `db.navigation.components`, which contains `db.glossary`, `db.bibliography`, `db.index`, and `db.toc`, and is used inside the content model for `chapter`, `appendix`, and `preface`.
contentmodel	Shared content models, for example, `db.admonition.contentmodel`, which contains the content model for `tip`, `warning`, `note`, etc.

There are a couple of other groupings designed to minimize duplication, but these are the most important.

General customization considerations

Creating a customized schema is similar to creating a customization layer for XSL. The schema customization layer is a new RELAX NG schema that defines your changes and includes the standard docbook schema. You then validate using the schema customization as your schema.

Example 3 is an empty RELAX NG customization that does nothing except define the name spaces and include the standard DocBook schema. The `href` attribute of the `include` element points to the location of the standard DocBook V5.0 schema.[13] All of the examples are given in both RNG and RNC form.

Example 3. Empty customization file

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>

  <!-- redefinitions of named patterns -->

</grammar>
```

RNC

```
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc" inherit = db
# redefinitions of named patterns
```

Elements

Adding elements

Adding an element typically takes two definitions. The first defines the new element and its content model, and the second adds the new element into the schema. We'll show two examples.

[13] Examples below use `docbook.rng` as schema location. If you want to create portable schema customization you should use standard web-accessible location like `http://docbook.org/xml/5.0CR7/rng/docbook.rng` and then use XML catalogs [http://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html] to resolve this location to your local copy of the schema for improved perfomance. Unfortunatelly at the time of this writing not all RELAX NG validators support XML catalogs.

Example 4 adds a new element, `person`, with the same content model as `author`. The new element will be allowed to appear wherever `author` can appear.

The `db.author` pattern is copied and renamed `dbx.person`, defining a new element called `person`. Then, the `db.author` pattern is redefined to be a choice of the current value or `dbx.person`. The `combine` attribute tells RELAX NG to combine this pattern with the existing named pattern. In this case, the value of the `combine` attribute is “choice”, which tells the parser that either the original pattern or this new pattern is a valid match.

Example 4. Adding a new element by duplicating an existing one

RNG
```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>
  <!-- define the new element -->
  <define name="dbx.person">
    <element name="person">
        <ref name="db.author.attlist"/>
        <ref name="db.credit.contentmodel"/>
    </element>
  </define>
  <!-- redefine the db.author pattern to allow db.person in
       the same places as db.author -->
  <define name="db.author" combine="choice">
    <ref name="dbx.person"/>
  </define>
</grammar>
```

RNC
```
default namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc"
# define the new element
dbx.person =
  element person { db.author.attlist, db.credit.contentmodel }
# redefine the db.author pattern to allow db.person in
# the same places as db.author
db.author |= dbx.person
```

The preceding method works well when you'd like a new element to be a clone or near-clone of an existing element. It gives you complete control over the content model, but only limited control over where the element is allowed. It works well when you want to allow the element in the same places as an existing element, and for this example that works nicely, since `author` is allowed in four different named patterns, each of which would have had to be redefined to allow `person`. But, if you can't find an existing element that is allowed in exactly the places you need, this method doesn't work as well.

Example 5 adds two new elements by combining them into a higher level pattern. In this example, we'll add two new inline elements for writing about assembly language, `register` and `instruction`. We will allow them whereever programming inlines or operating system inlines are allowed. Example 5 defines the two elements, creates a new named pattern (`dbx.asm.inlines`) that contains them, and adds that pattern to `db.programming.inlines` and `db.os.inlines`. Since these two patterns don't have any elements in common, the strategy used in Example 4 would require selecting two different elements to “clone”, which would be messy.

Example 5. Adding new inline elements

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>
  <!-- define the new elements -->
  <define name="dbx.register">
    <element name="register">
      <text/>
    </element>
  </define>
  <define name="dbx.instruction">
    <element name="instruction">
      <text/>
    </element>
  </define>
  <!-- create a new pattern that contains the new inlines -->
  <define name="dbx.asm.inlines">
    <choice>
      <ref name="dbx.register"/>
      <ref name="dbx.instruction"/>
    </choice>
  </define>
  <!-- add the new inlines to programming and os inlines -->
    <define name="db.programming.inlines" combine="choice">
      <ref name="dbx.asm.inlines"/>
    </define>
    <define name="db.os.inlines" combine="choice">
      <ref name="dbx.asm.inlines"/>
    </define>
</grammar>
```

RNC

```
default namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc"
# define the new elements
dbx.register = element register { text }
dbx.instruction = element instruction { text }
# create a new pattern that contains the new inlines
dbx.asm.inlines = dbx.register | dbx.instruction
# add the new inlines to programming and os inlines
db.programming.inlines |= dbx.asm.inlines
db.os.inlines |= dbx.asm.inlines
```

Deleting elements

Deleting elements is straightforward, but takes some care and planning. Example 6 deletes the `important` admonition element by redefining it with a content model of `notAllowed`. Note that in this example, the redefinition is inside the `include` element. This is required for redefinitions that completely replace an existing pattern.

Be careful; If you delete an element that is a required part of another element's content model, you can make it impossible to create a valid document. For example, if you delete the `title` element, you won't be able to validate a `book` because a `book` requires a `title`.

Example 6. Deleting an element

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng">
    <!-- redefine important element as notAllowed -->
    <define name="db.important">
      <notAllowed/>
    </define>
  </include>
</grammar>
```

RNC

```
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc" inherit = db {
  # redefine important element as notAllowed
  db.important = notAllowed
}
```

Customizing the content model of existing elements

Example 7 expands the definition of `author` to include two new elements, `born` and `died`. The `author` element allows two content models, `db.person.author.contentmodel`, which defines an author who is a person, and `db.org.author.contentmodel`, which defines an author that is an organization. We will modify `db.person.author.contentmodel` so that only authors who are persons can have the new elements.

Example 7. Modifying the content model of an element

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>

  <define name="db.person.author.contentmodel" combine="interleave">
    <interleave>
      <optional>
        <element name="born">
          <ref name="db.date.contentmodel"/>
        </element>
      </optional>
      <optional>
        <element name="died">
          <ref name="db.date.contentmodel"/>
        </element>
      </optional>
    </interleave>
  </define>
</grammar>
```

RNC

```
default namespace = "http://docbook.org/ns/docbook"
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc"

db.person.author.contentmodel &=
  element born { db.date.contentmodel }?
  & element died { db.date.contentmodel }?
```

This modification will allow instances like this:

```
<author>
  <personname>Babe Ruth</personname>
  <born>02/06/1895</born>
  <died>08/16/1948</died>
</author>
```

but because we only modified the content model for authors who are human, it won't allow an instance like this, which uses `db.org.author.contentmodel`:

```
<!-- INVALID -->
<author>
  <orgname>Boston Red Sox</orgname>
  <died>1919</died>
  <born>2004</born>
</author>
```

Attributes

Adding attributes

The simplest way to add an attribute to a single element is to add it to the attlist pattern for that element. Example 8 adds the optional attributes `born` and `died` to the attribute list for `author`. The `db.author.attlist` named pattern is redefined with the `combine` attribute set to “interleave”, which interleaves the two new optional attributes with the existing attributes on the list.

Example 8. Adding attributes

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>

  <define name="db.author.attlist" combine="interleave">
    <interleave>
      <optional>
        <attribute name="born">
          <ref name="db.date.contentmodel"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="died">
          <ref name="db.date.contentmodel"/>
        </attribute>
      </optional>
    </interleave>
  </define>
</grammar>
```

RNC

```
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc" inherit = db

db.author.attlist &=
  attribute born { db.date.contentmodel }?
  & attribute died { db.date.contentmodel }?
```

Unlike Example 7, Example 8 allows the new attributes to appear on any `author` element, not just those using the person content model.

Example 9 shows how you could limit the use of these attributes to authors who are persons. In this example, the new attributes are interleaved with the `db.person.author.contentmodel`. The only difference between this example and Example 7 is that the added patterns are identified as attributes rather than elements. This shows some of the flexibility of RELAX NG, which treats attributes and elements very consistently.

Example 9. Adding attributes; alternate method

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>
  <!-- redefinitions of named patterns -->
  <define name="db.person.author.contentmodel" combine="interleave">
    <interleave>
      <optional>
        <attribute name="born">
          <ref name="db.date.contentmodel"/>
        </attribute>
      </optional>
      <optional>
        <attribute name="died">
          <ref name="db.date.contentmodel"/>
        </attribute>
      </optional>
    </interleave>
  </define>
</grammar>
```

RNC

```
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc" inherit = db
# redefinitions of named patterns
db.person.author.contentmodel &=
  attribute born { db.date.contentmodel }?
  & attribute died { db.date.contentmodel }?
```

There is one difference in the treatment of attributes and elements that is worth noting. By the XML 1.0 definition, the relative order of attributes is not significant. Therefore, the `interleave` block is not required for attributes, though it does no harm.

Deleting attributes

Deleting an attribute is similar to deleting an element, except that you use the RELAX NG `empty` pattern rather than `notAllowed`. Example 10 deletes the linking attributes, which are collected in the `db.common.linking.attributes` pattern, by defining that pattern as `empty`.

Example 10. Deleting an attribute

RNG
```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng">
    <define name="db.common.linking.attributes">
      <empty/>
    </define>
  </include>
</grammar>
```

RNC
```
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc" inherit = db {
  db.common.linking.attributes = empty
}
```

Generally, `empty` is used when deleting attributes and `notAllowed` is used when deleting elements.

Changing permitted content of attributes

Example 11 modifies `db.spacing.enumeration` to add the additional value “large”. Note that to remove a value from an enumeration, you need to redefine the entire enumeration, minus the values you don't need.

Example 11. Deleting an attribute

RNG
```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns:db="http://docbook.org/ns/docbook"
         ns="http://docbook.org/ns/docbook"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="docbook.rng"/>
  <!-- add value to an enumeration -->
  <define name="db.spacing.enumeration" combine="choice">
    <value>large</value>
  </define>
</grammar>
```

RNC
```
namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc" inherit = db
# add value to an enumeration
db.spacing.enumeration |= "large"
```

Naming and versioning DocBook customizations

DocBook V5.0 is not tightly coupled with some particular validation technology like DTDs. This also means that DocBook V5.0 documents don't have to (and usually don't) start with a document type declaration (<!DOCTYPE…>) to specify the schema (DTD) to use. Instead, DocBook V5.0 instances can be easily distinguished from other XML vocabularies by using elements in the `http://docbook.org/ns/docbook` namespace. This namespace is enough to distinguish DocBook from other XML based formats. But the DocBook schema evolves over time and there are several

versions of DocBook (e.g. 3.1, 4.2, 4.5 and 5.0). Since DocBook version 5.0, the actual version used is indicated in the `version` attribute on a root element.

```
<book xmlns="http://docbook.org/ns/docbook"
      version="5.0">
  …
</book>
```

Future versions of DocBook documents will start with the same markup, except the version number will be raised, for example to 5.1 or 6.0. The namespace will remain the same until the semantics of the elements change in a backward incompatible way, which is very unlikely to happen.

If you create a DocBook schema customization you must change the `version` attribute to distinguish your customization from the “official” DocBook. Changing the namespace is not recommended because that would break the processing tools. Remember that changing namespaces is the same as renaming all elements in the namespace.

When you customize the schema, use the following syntax to identify your DocBook derivation:

base_version-[subset|extension|variant] [*name*[-*version*]?]+

For example:

```
5.0-subset simplified-1.0
5.0-variant ASMBook
5.0-variant ASMBook-2006
5.0-extension MathML-2.0 SVG-1.1
```

The first part of the version identifier is the version number of the DocBook schema from which you derived your customization.

If your schema is a proper subset, you can advertise this status by using the `subset` keyword in the description. If your schema contains any markup model extensions, you can advertise this status by using the `extension` keyword. If you'd rather not characterize your variant specifically as a subset or an extension, use the `variant` keyword.

After these keywords you may add a whitespace separated list of customization identifiers. Each name may be optionally followed by its version number.

FAQ

1. Authoring

1.1. How do I attach a schema to a DocBook V5.0 document when I do not want to use DTDs and !DOCTYPE?

There is no standard way of associating a RELAX NG schema with a document. Most tools provide some mechanism for performing this association, consult the documentation for your application. In some tools you must specify schema manually each time you want to edit/process your document.

1.2. How do I use entities like `&ndash;` in DocBook V5.0?

Modern schema languages (including RELAX NG and W3X XML Schema) do not provide any means to define entities that can be used for easier typing of special characters. Some editors provide functions or special toolbars that allow you to easily pick necessary character and insert it into document as a raw Unicode character or a numeric character reference.

Another possibility is to include entity definitions in the prolog of your document. Entity definition files[14] are now maintained by W3C. You can reference definition files with entity definitions you are interested in and then reference imported entities. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE article [
<!ENTITY % isopub SYSTEM "http://www.w3.org/2003/entities/iso8879/isopub.ent">
%isopub;
]>
<article xmlns="http://docbook.org/ns/docbook" version="5.0">
<title>DocBook V5.0 &ndash; the superb documentation format</title>
…
```

1.3. How to modularize documents?

You can use XInclude[15] for this task. There is an alternative schema for DocBook V5.0 that contains XInclude elements. This is necessary to make some XML editors happy. This schema can be found in files that end with letters “xi”, e.g. `docbookxi.rnc` instead of `docbook.rnc`.

2. Stylesheets

2.1. Will the current DocBook XSL stylesheets (XSLT 1.0 based implementation) be maintained and improved in the future since work on a new XSLT 2.0 based implementation has started?

Yes, the current stylesheets (like 1.69.1) will be supported and improved further because they are very widely deployed and work with many existing XSLT processors.

Surely there will be a point in a future when all new development will be switched to the XSLT 2.0 based implementation. But this will not happen until all features of the current stylesheets are implemented in the new stylesheets and until there is more than one usable XSLT 2.0 processor available.

3. Schema customizations

3.1. How can I extend the DocBook schema with MathML elements?

The basic DocBook schema allows elements from the MathML namespace to appear inside the `equation` element. This means that you can validate a DocBook+MathML document, but MathML content will be ignored during the validation. You will also not be able to use guided editing for the MathML content.

If you need strict validation of MathML content or guided editing for MathML, you can easily extend the base DocBook schema with the MathML schema.

Procedure 1. Extending the DocBook schema with the MathML schema

1. Download the MathML RELAX NG schema from http://yupotan.sppd.ne.jp/relax-ng/mml2.html and unpack it somewhere (e.g. into a `mathml` subdirectory).

2. Create a schema customization in compact syntax—`dbmathml.rnc`:

RNC

```
namespace html = "http://www.w3.org/1999/xhtml"
namespace mml = "http://www.w3.org/1998/Math/MathML"
```

[14] http://www.w3.org/2003/entities/
[15] http://www.w3.org/TR/xinclude/

```
namespace db = "http://docbook.org/ns/docbook"

include "/path/to/docbook.rnc" {
  db._any.mml = external "mathml/mathml2.rnc"
  db._any =
    element * - (db:* | html:* | mml:*) {
      (attribute * { text }
       | text
       | db._any)*
    }
}
```

Or, alternatively, you can use the XML syntax of RELAX NG—`dbmathml.rng`:

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<include href="/path/to/docbook.rng">
  <define name="db._any.mml">
    <externalRef href="mathml/mathml2.rng"/>
  </define>

  <define name="db._any">
    <element>
      <anyName>
        <except>
          <nsName ns="http://docbook.org/ns/docbook"/>
          <nsName ns="http://www.w3.org/1999/xhtml"/>
          <nsName ns="http://www.w3.org/1998/Math/MathML"/>
        </except>
      </anyName>
      <zeroOrMore>
        <choice>
          <attribute>
            <anyName/>
          </attribute>
          <text/>
          <ref name="db._any"/>
        </choice>
      </zeroOrMore>
    </element>
  </define>
</include>

</grammar>
```

3. Now use the customized schema (`dbmathml.rnc` or `dbmathml.rng`) instead of the original DocBook schema.

3.2. How can I extend the DocBook schema with SVG elements?

The situation is the same as with MathML support. You can use elements from the SVG namespace inside the `imageobject` element.

Procedure 2. Extending the DocBook schema with the SVG schema

1. Download the SVG RELAX NG schema from http://www.w3.org/Graphics/SVG/1.1/rng/rng.zip and unpack it somewhere (e.g. into an `svg` subdirectory).

2. Create a schema customization in compact syntax—`dbsvg.rnc`:

RNC

```
namespace html = "http://www.w3.org/1999/xhtml"
namespace db = "http://docbook.org/ns/docbook"
namespace svg = "http://www.w3.org/2000/svg"

include "/path/to/docbook.rnc" {
  db._any.svg = external "svg/svg11.rnc"
  db._any =
    element * - (db:* | html:* | svg:*) {
      (attribute * { text }
       | text
       | db._any)*
    }
}
```

Or, alternatively, you can use the XML syntax of RELAX NG—`dbsvg.rng`:

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<include href="/path/to/docbook.rng">
  <define name="db._any.svg">
    <externalRef href="svg/svg11.rng"/>
  </define>

  <define name="db._any">
    <element>
      <anyName>
        <except>
          <nsName ns="http://docbook.org/ns/docbook"/>
          <nsName ns="http://www.w3.org/1999/xhtml"/>
          <nsName ns="http://www.w3.org/2000/svg"/>
        </except>
      </anyName>
      <zeroOrMore>
        <choice>
          <attribute>
            <anyName/>
          </attribute>
          <text/>
          <ref name="db._any"/>
        </choice>
      </zeroOrMore>
    </element>
  </define>
</include>

</grammar>
```

3. Now use the customized schema (`dbsvg.rnc` or `dbsvg.rng`) instead of the original DocBook schema.

3.3. Is it possible to use the previous two customizations for MathML and SVG together?

Yes, you can create a special schema customization that combines both MathML and SVG with the DocBook schema. In compact syntax, the merged schema is:

RNC

```
namespace html = "http://www.w3.org/1999/xhtml"
namespace mml = "http://www.w3.org/1998/Math/MathML"
namespace db = "http://docbook.org/ns/docbook"
namespace svg = "http://www.w3.org/2000/svg"

include "/path/to/docbook.rnc" {
  db._any.mml = external "mahtml/mathml2.rnc"
  db._any.svg = external "svg/svg11.rnc"
  db._any =
    element * - (db:* | html:* | mml:* | svg:*) {
      (attribute * { text }
       | text
       | db._any)*
    }
}
```

Or alternatively in the full RELAX NG syntax:

RNG

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">

<include href="/path/to/docbook.rng">
  <define name="db._any.mml">
    <externalRef href="mathml/mathml2.rng"/>
  </define>

  <define name="db._any.svg">
    <externalRef href="svg/svg11.rng"/>
  </define>

  <define name="db._any">
    <element>
      <anyName>
        <except>
          <nsName ns="http://docbook.org/ns/docbook"/>
          <nsName ns="http://www.w3.org/1999/xhtml"/>
          <nsName ns="http://www.w3.org/1998/Math/MathML"/>
          <nsName ns="http://www.w3.org/2000/svg"/>
        </except>
      </anyName>
      <zeroOrMore>
        <choice>
          <attribute>
            <anyName/>
          </attribute>
          <text/>
```

```
          <ref name="db._any"/>
        </choice>
      </zeroOrMore>
    </element>
  </define>
</include>

</grammar>
```

3.4. Are there any other examples of schema customization available?

Sure. Some of the are listed bellow:

- Sample customization of ITS and DocBook[16]

4. Tool specific problems

4.1. I'm using Altova XMLSpy to validate DocBook V5.0 instances against the W3C XML Schema (`docbook.xsd`). XMLSpy complains about undefined `xml:id` attributes?

XMLSpy always uses its own bundled version of `xml.xsd` which unfortunately doesn't define the `xml:id` attribute. The bundled version of `xml.xsd` is hardwired into the program and cannot be replaced by a newer version. To solve this problem you must upgrade to version 2006 SP1.

Bibliography

[RNCTUT] Clark, James – Cowan, John – MURATA, Makoto: RELAX NG Compact Syntax Tutorial. Working Draft, 26 March 2003. OASIS. http://relaxng.org/compact-tutorial-20030326.html

[NVDLTUT] Nálevka, Petr: NVDL Tutorial. http://jnvdl.sourceforge.net/tutorial.html

[XMLID] Marsh, Jonathan – Veillard, Daniel – Walsh, Norman: xml:id Version 1.0. W3C Recommendation, 9 September 2005. http://www.w3.org/TR/xml-id/

[DB5SPEC] Norman, Walsh: The DocBook Schema. Working Draft 5.0a1, OASIS, 29 June 2005. http://www.docbook.org/specs/wd-docbook-docbook-5.0a1.html

[16] http://www.w3.org/TR/xml-i18n-bp/#docbook-plus-its