

## Writing Docbook Files with Open Office

# Writing Assistant

Docbook is the de-facto standard for technical documentation. Open Office helps to prevent XML format documents from turning into a coding nightmare.

BY ROLF STRATHEWERD

The Docbook format is a good choice for program manuals. Websites, books, manpages and an integrated help system can all be generated from a central XML file. And this would seem to save you a lot of work at first glance. Open Source projects like Gnome and KDE, and companies such as HP or Sun, all use Docbook for this reason.

But, unfortunately, a closer look reveals how verbose the Docbook format really is. Longer text passages without any formatting are simple enough, but tables, lists, interfaces and other elements might tax your typing skills. XML editors can help you with this work, but the actually coding can be bothersome. Also, learning a special language just for documentation is a lot of work. If the manual is to be written by mere mortals – as opposed to programmers – it makes sense to keep it simple.

One practical and obvious approach would be to allow technical authors to use a normal word processor. This would take some of the load off authors without the required Docbook skills, although it does assume automatic conversion of the results.

### Open Office as a Docbook Editor

Open Office has quite a few of the prerequisites that make this approach sound

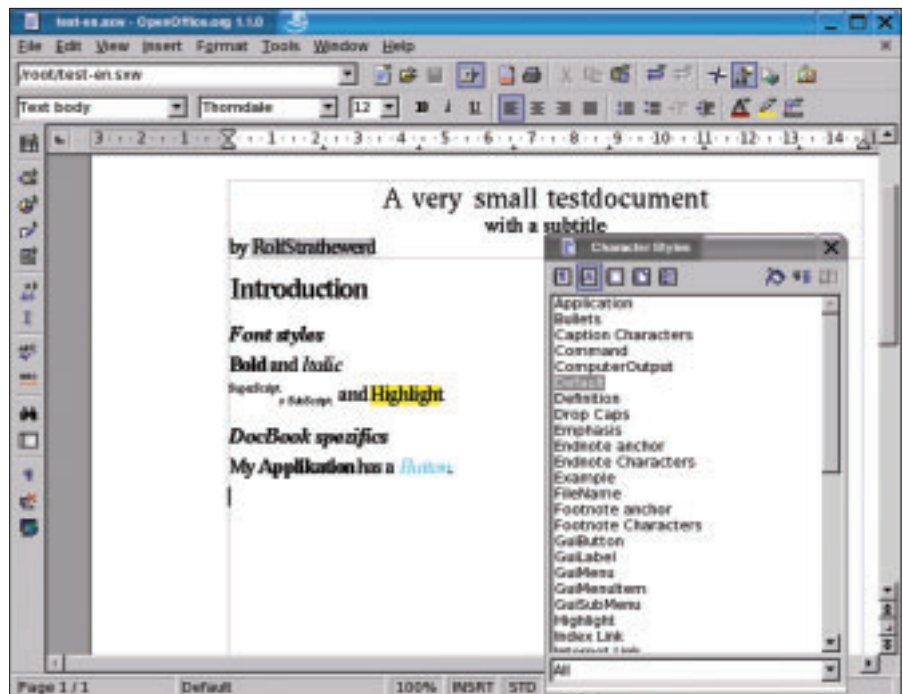


Figure 1: The latest Open Office versions (1.1 RC3 in our example) can handle Docbook files. Styles are assigned to various XML elements to provide this functionality

realistic. Just like Docbook, the native Open Office format is based on XML. Just don't be fooled by the *sxw* suffix on Open Office text files. SXW files are simple ZIP archives. Among other components (such as embedded images), they include the text itself in a file called *content.xml* that contains the XML formatted document. As the Open Office framework supports filter based file importing and exporting, it should be simple to integrate the Docbook format. But actually, it is not quite that simple.

The first implementation of this idea was by Éric Bellot. Éric adopted an usual approach for his OOo2sDBK [4] (Open Office org to simplified Docbook) tool, opting to use Python and Saxon as conversion tools, rather than native Open Office tools. As Éric was only interested in an Open Office to Docbook converter – Éric did not envisage a round trip – and, being French himself, only provided French language documentation, the program did not prove overly popular.

The second incarnation tried to use only native Open Office tools, and to read Docbook files in the word processor. The developers cautiously described their implementation as a proof of concept, and it failed to achieve similar results to OOo2sDBK.

Open Office 1.1 includes a development based on this variant, and thus supports Simplified Docbook (a "lite" version of the standard). RC3 (Release Candidate 3) already provides beta standard Docbook functionality, although the documentation does not quite match the software. The basic statements are appropriate, but some detail is not quite up to date, which can lead to confusion at times.

### XML Variants

Although both Open Office and Docbook use the XML data format, there is one major difference between the two. Docbook has nothing to do with the document layout. Its tags only specify the function of a text passage (the name

of the author, headings, or body text), without actually specifying the appearance. This task is handled by stylesheets that help produce the final layout of the document. Referring to the stylesheet allows attributes such as italicized authors' names and headings in larger font sizes to be applied.

In contrast to this, word processing systems are not typically concerned with the content and its structure, but with the appearance of the text. Formatting instructions directly influence the appearance of the text. If you select Courier, 8 point, bold and green for a text passage, these attributes will be lost during conversion to Docbook format.

Indirect style-based formatting allows these two worlds to co-operate. Authors can use styles to mark up a text passage as a heading, for example. The style contains details of the layout applicable to headings.

### Practical Applications

In addition to Open Office 1.1 (at least RC3) [1] you will need a current Java

package. The latter is essential for converting from and to Docbook. The typical point of departure is a document style called *DocBookStyle.stw*, which resides in *path to Open Office/share/xslt/docbook/*. This style document includes all the styles required to work with Docbook.

The Stylist provides an initial overview. You can press [F11] to display a list of styles. The list includes some default styles and the new additions for Docbook. Figure 1 shows a typical view, based on a sample text file.

Selecting *Save as...* and then *DocBook (simplified)* as the format, tells Open Office to generate an XML file. Figure 2 shows the structure of the file. If you look closely, you will note that the attributes bold and italics have disappeared. The scripts do not convert these attributes to Docbook tags, although the text is retained.

You can use two methods with XML files created in this way: online formatting with CSS (Cascading Style Sheets) or XSL (Extensible Stylesheet Language)

transformation. The first technique requires a stylesheet, such as the sample at [3]. The XML file additionally needs the `<?xml-stylesheet href="sdocbook.css" type="text/css"?>` tag adding to the `<!DOCTYPE...` line. The quick and elegant CSS technique assumes a state-of-the-art browser.

The second approach allows you to choose the target format. One of the simplest transformation types produces a large HTML file:

```
saxon example.xml /Path/docbook2
/html/docbook.xsl > example.html
```

Sourceforge [5] provides a number of useful transformation scripts. There are also alternative processing tools to the Saxon utility [2].

Some of you may be wondering why you should go to all this trouble, just to produce HTML. Open Office can do that anyway, and can even generate PDF documents as of version 1.1. Although this objection is valid, Docbook does offer some advantages when you need to sep-

# DUAL AMD OPTERON LINUX WORKSTATION



- ✓ Dual Opteron processors
- ✓ 64-bit platform
- ✓ Up to 16.0GB of RAM
- ✓ Dual channel memory
- ✓ 64-bit PCI-X bus
- ✓ NVIDIA graphics
- ✓ Gigabit Ethernet
- ✓ 5.1 digital audio
- ✓ 64-bit Red Hat OS
- ✓ 3 year warranty

Prices from **£1250** + VAT

**OUR FASTEST WORKSTATION** to date is now available for less than £1300. Powered by AMD's new flagship Opteron 64-bit processors, it is available as a dual processor number cruncher with lots of RAM. It makes an excellent development system or a powerful graphics workstation.

At Digital Networks, we specialise in servers, storage, workstations and desktops designed specifically for Linux deployment.

Visit **www.dnuk.com** and find out why corporate customers, small and medium businesses and most UK universities choose us for their IT requirements.

Above specification is an example, and is fully configurable. Prices correct as of 16/9/03. Please check [www.dnuk.com](http://www.dnuk.com) for current prices.

 Digital Networks  
United Kingdom

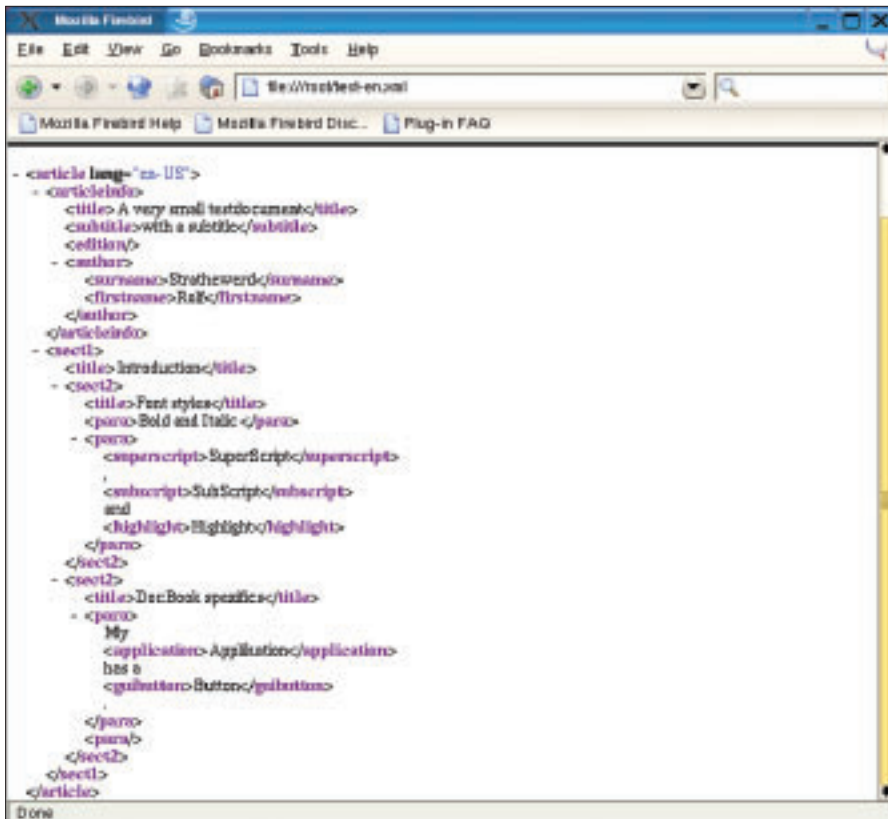


Figure 2: The Docbook XML code generated from the test document and viewed in Mozilla Firebird

arate the content from the presentation, for example, when several people are working on a project where a uniform appearance is required.

## Limits

Éric Bellot's OOo2sDBK encountered some difficulties which have not quite been resolved so far. For example, Open Office does not provide any native means of producing nested tags, although there are situations where Docbook needs them; such as metadata in the *articleinfo* tag. The author info for a document needs to be nested as shown in Listing 1.

Although Open Office is now capable of generating some of these tags, the documentation describes a completely different method. According to the documentation, the author is required to use bookmarks, specifying the inner ele-

ments of the nested structure first, and using the stylist to format them, before going on to convert the whole area into a bookmark with a specific name. You can assume that this approach will be adopted by the final Open Office 1.1 release, unless a superior method is found in the meantime. The current release (RC4) does not implement this facility.

The developers have devised an interim solution for the *articleinfo* area, using several global variables to store the appropriate information. Each variable has a serial number starting with *articleinfo.author\_0.firstname\_0*, which expects the first name of the first author of the current document. These variables are hidden in the menu below *Insert | Fields | Other*.

## Pitfalls

Take care when loading an existing Docbook file. As Open Office only supports a subset of all tags, the results may not be what you expect, if you save the file. Avoid saving files under their original names. Cross-check the XML code with Table 1 to find out where and when you are likely to run into any issues.

### Listing 1: Docbook File Author

```
<author>
  <firstname>Rolf</firstname>
  <surname>Strathewerd</surname>
</author>
```

### Listing 2: Export Extension

```
<xsl:when test="@text:style-
name='Source Text'">
  <xsl:element
name="programlisting">
  <xsl:value-of select="."/>
</xsl:element>
</xsl:when>
```

Just like an Open Office text, a Docbook file is subdivided into sections. You can define a section explicitly by selecting *Insert | Section* or implicitly by applying styles to a text. The styles *Heading 1* through *Heading 4* are examples of the latter. There are three additional sections that an author can specify explicitly if required: *ArticleInfo*, *Abstract* or *Appendix* (these names are case-sensitive!).

Graphics are another minor pitfall. If the author checks the *Link* checkbox when inserting a graphic, Open Office will not display the results correctly, unless the graphic is declared as an *inlinegraphic*. Make sure that you retain the original external image file in this case. In contrast, embedded graphics are placed in the SXW file. The same applies to other media objects.

## More Tags for Open Office

A glance at the list of supported tags in Table 1 shows that some of these tags are not part of the Simplified Docbook format: such as the interface elements *guibutton* or *guimenu*. These tags are specified by the full Docbook standard. On the other hand, important elements, such as *programlisting* are missing. Luckily, it is quite simple to resolve this issue.

Open Office uses only one document style and two XSL stylesheets to support

### Listing 3: Import Extension

```
<xsl:style
match="programlisting">
  <xsl:element name="text:span">
    <xsl:attribute
name="text:style-name">Source
Text</xsl:attribute>
    <xsl:apply-styles/>
  </xsl:element>
</xsl:style>
```

the whole range of Docbook functions. Both stylesheets (for the data export and import facilities) are stored with the style below `/Path to Open Office/share/xslt/docbook`.

To add a tag, first edit the `sofftodocbookheadings.xsl` stylesheet. This stylesheet provides the export function. The `Source Text` paragraph style is a useful starting point for the `programlisting` tag. Listing 2 shows the lines you need to add. Use the text block as the section.

Open Office will now save `programlisting`, but you need to tell Open Office to load the tag to provide a genuine round trip. This is just as easy, as can be seen in Listing 3.

This enhancement was quite simple because `Source Text` is already defined as an Open Office style. To define other styles, you would also need to add the styles to the style. But again, doing so is not difficult. Selecting `Format | Styles | Catalog` will take you where you need to go.

## Conclusion

It does look like the Open Office team was bragging when it promised full support for Simplified Docbook. At a second glance, the support provided is less complete than might be expected. The functionality turns out to be a mixture of Docbook and Simplified Docbook, which was obviously compiled with software documentation in mind. This is a pragmatic approach designed to tempt anyone looking to write documentation into risking typing a few lines.

However, the current version cannot be recommended for more complex documentation projects. The fact that stylesheets are easily extensible does auger well for the future – maybe a few generous code donors can help the Open Office project on its way. ■

## INFO

- [1] Open Office: <http://www.openoffice.org/>
- [2] Saxon: <http://saxon.sourceforge.net/>
- [3] Sample stylesheet:  
<http://www.oasis-open.org/docbook/xml/simple/4.1.2.5/sdocbook.css>
- [4] OOo2sDBK: <http://www.chez.com/ebellot/ooo2sdbk/>
- [5] Transformation scripts for Docbook:  
<http://sourceforge.net/projects/docbook/>

Table 1: Compatibility

XML-Tag	Supported	XML-Tag	Supported
abbrev		link	✓
abstract	✓	listitem	✓
acronym		literal	
appendix	✓	literallayout	
articleinfo	✓	mediaobject	✓
attribution		note	✓
affiliation	✓	objectinfo	
application <sup>1</sup>	✓	option	
article	✓	olink <sup>1</sup>	✓
audioobject	✓	orderedlist	✓
audiodata		orgname	✓
authorblurb		othercredit	
authorinitials		othername	
authorgroup		para	✓
author	✓	phrase	
biblio..		programlisting	
blockquote	✓	pubdate	
caption	✓	publishername	
citetitle		quote	
colspec	✓	releaseinfo	✓
command	✓	replaceable	
computeroutput	✓	revdescription	
copyright	✓	revhistory	
corpauthor		revision	
date		revnumber	
edition	✓	revmark	
editor		row	✓
email		section	✓
emphasis	✓	sectioninfo	
entry	✓	sidebar	
epigraph		subject	
example		subjectset	
figure		subjectterm	
filename	✓	subtitle	✓
firstname	✓	subscript <sup>1</sup>	✓
footnote	✓	superscript <sup>1</sup>	✓
guibutton <sup>1</sup>	✓	surname	✓
guimenu <sup>1</sup>	✓	systemitem	✓
guisubmenu <sup>1</sup>	✓	table	✓
highlight <sup>1</sup>	✓	tbody	✓
holder		term	✓
honorific		textobject	
imageobject	✓	tgroup	✓
imagedata	✓	thead	✓
informaltable	✓	title	✓
inlinemediaobject		titleabbrev	
issuenum		trademark	
itemizedlist	✓	ulink	✓
inlinegraphic	✓	userinput	✓
jobtitle		variablelist	✓
keyword		varlistentry	✓
keywordset		videodata	
keycap <sup>1</sup>	✓	videoobject	
keycombo <sup>1</sup>	✓	volumenum	
keysym <sup>1</sup>	✓	xref	✓
legalnotice		year	✓
linage			

<sup>1</sup> These tags are not simplified Docbook tags, although they are specified in the full standard.