

# Short Attention Span Docbook Toolchain – Setup Guide

New Auburn Personal Computer Services LLC

Eau Claire  
611 Davis Avenue  
54703  
USA

<info@napcs.com>

Copyright © 2011 Brian P. Hogan

## Table of Contents

1. Overview .....	1
2. Installation .....	1
3. Creating Books and Articles .....	4
4. Generating Output from Docbook .....	6
5. Additional Customizations .....	7

## 1. Overview

This short guide will walk you through everything you need to use this toolchain to author your own books, articles, and other documents using the open-source Docbook markup language.

### Package Contents

This package contains all of the tools you need to start working with Docbook, including

- The Docbook stylesheets
- An XML validation tool
- The Saxon XSLT processor (version 6.5.3)
- The Apache FOP FO processor for making PDFs
- Scripts that generate new book and article projects, complete with the option to create samples so you have a good starting point.

## 2. Installation

In order to build books with this toolchain, you need to get it installed on your platform.

## Windows setup

Windows users should use the installer <sup>1</sup> which sets up the necessary files and also sets up the path.

### Manual installation on Windows

The easiest way to use this package is to use the application installer, but if you need to do a manual install, you can download the docbook zip file from <http://www.napcsweb.com/files/docbook/docbook.zip> and extract it to `c:\docbook`. Next, you'll want to add this folder to your PATH. To do that, press **Windows+Pause** to bring up the *System Properties* window. Select the Advanced tab and then select the Environment Variables button. In the User variables section, locate the Path variable and edit its contents. Append `;c:\docbook` to the end of the value. If there currently is no Path variable, create a new one called *PATH* and use `c:\docbook` for the value. Press OK to commit these changes. Press OK on the *System Properties* window to finish up.

Open a new command prompt and type **SET**. The PATH variable now reflects the changes you made.

## Installing a Java Runtime Environment

Download the Java Runtime Environment from Sun <sup>2</sup> and get it installed on your system if you don't already have it. Once installed, open a new command prompt and type

```
java -version
```

You'll see something similar to this:

```
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05-237)
Java HotSpot(TM) Client VM (build 1.5.0_13-119, mixed mode, sharing)
```

## Installing Ruby

Download and install the Ruby Installer for Windows <sup>3</sup> and install it, accepting the defaults. To verify the installation, open a new command prompt and type

```
ruby -v
```

You'll see something like this:

```
ruby 1.8.6 (2007-03-13 patchlevel 0) [i386-mswin32]
```

At this point, everything is installed and you're ready to begin. Test it out by creating a new project:

```
generate book my_test_project with_sample
```

<sup>1</sup>Docbook setup for Windows: <http://www.napcsweb.com/files/docbook/docbook-setup.exe>

<sup>2</sup>Java Runtime: <http://java.sun.com/javase/downloads/index.jsp>

<sup>3</sup>Ruby: <http://www.rubyinstaller.org>

```

Creating docbook project...
- my_test_project
- my_test_project/images
- my_test_project/images/src
- my_test_project/cover
- my_test_project/xsl
- my_test_project/w3centities-f.ent
- my_test_project/xsl/pdf.xsl
- my_test_project/xsl/html.xsl
- my_test_project/xsl/epub.xsl
- my_test_project/xsl/epub.css
- my_test_project/xsl/html.css
- my_test_project/xsl/rtf.xsl
- my_test_project/Rakefile
- my_test_project/images
- my_test_project/book.xml
- my_test_project/chapter01.xml
Done

```

Now you have a simple sample project with a book file and a chapter. Convert this to a PDF to make sure everything works.

```
cd my_test_project
```

```
rake book.pdf
```

You'll get a brand-new PDF of the sample book.

## Mac OS X Setup

Setup on Mac OS X 10.5 or above is quite easy. You have Java and Ruby already installed on the machine in most cases. All you need to do is download the files from <http://www.napcs.com/products/docbook/> and extract to `docbook` in your home folder. Once extracted, you should have a list of files that resembles this in your `~/docbook` folder:

```

drwxr-xr-x  18 brianhogan  brianhogan  612B Apr  2 11:56 .
drwx-----+ 63 brianhogan  brianhogan  2.1K Apr  2 10:12 ..
-rwxr-----  1 brianhogan  brianhogan   53K Apr  2 10:23 docbook.pdf
-rwxr-----@ 1 brianhogan  brianhogan   5.1K Apr  2 13:58 generate
-rwxr-----  1 brianhogan  brianhogan  165B Apr  2 10:52 generate.bat
drwxr-xr-x  35 brianhogan  brianhogan  1.2K Mar 31 10:15 jars
drwxr-xr-x  11 brianhogan  brianhogan  374B Apr  3 21:05 template
drwxr-xr-x  41 brianhogan  brianhogan  1.4K Mar 25 22:19 xsl

```



### Permission Denied

You may need to set the executable bit for the `generate` file so you can use it.

```
chmod +x ~/docbook/generate
```

To create a new project, type

```
~/docbook/generate book my_test_project with_sample
```

This generates the project skeleton:

```

Creating docbook project...
- my_test_project
- my_test_project/images
- my_test_project/images/src
- my_test_project/cover
- my_test_project/xsl
- my_test_project/w3centities-f.ent
- my_test_project/xsl/pdf.xsl
- my_test_project/xsl/html.xsl
- my_test_project/xsl/epub.xsl
- my_test_project/xsl/epub.css
- my_test_project/xsl/html.css
- my_test_project/xsl/rtf.xsl
- my_test_project/Rakefile
- my_test_project/images
- my_test_project/book.xml
- my_test_project/chapter01.xml
Done

```

Now try generating a PDF.

```
cd my_test_project
```

```
rake book.pdf
```

You'll get a brand-new PDF of the sample book.



### Tip

Add the `docbook` folder to your path to make the `generate` command available without having to specify the full path. Modify your `.bash_profile` to change the path.

## 3. Creating Books and Articles

You've already seen how to generate a sample book, but the generator can create empty books and can also create articles, which are designed to be shorter and more like individual documents.

### Creating books

`generate book my_book sample` generates a new book with some sample content so you can see how you might construct your book. It also generates a sample external chapter file as an example of how you would split your book into multiple files.

`generate book my_book` generates a `book.xml` file, a `chapter01.xml` file, and a PDF XSLT customization layer file, as well as the directory structure. This sets you up to start your book from scratch.

### Chapters

You can generate chapters as well so you don't have to remember what exactly goes into the chapter template.

`generate chapter chapter02` generates a `chapter02.xml` file in your current directory.

`generate chapter chapter2/some_stuff` generates a `some_stuff.xml` file in a `chapter2` folder within your current directory.

## Articles

An article is just like a book, except it doesn't have chapters. It's ideal for simple projects where you don't need the complex structure of a book. The document you're reading is marked up as a Docbook article.

`generate article foo` creates a new article called `article.xml` in the `foo` subfolder of your current folder.

```
- foo/images
- foo/xsl
- foo/xsl/pdf.xml
- foo/Rakefile
- foo/w3centities-f.ent
- foo/article.xml
```

## Exploring the Project structure

The sample project contains the following files:

```
Creating docbook project...
- my_test_project
- my_test_project/images
- my_test_project/images/src
- my_test_project/cover
- my_test_project/xsl
- my_test_project/w3centities-f.ent
- my_test_project/xsl/pdf.xsl
- my_test_project/xsl/html.xsl
- my_test_project/xsl/epub.xsl
- my_test_project/xsl/epub.css
- my_test_project/xsl/html.css
- my_test_project/xsl/rtf.xsl
- my_test_project/Rakefile
- my_test_project/images
- my_test_project/book.xml
- my_test_project/chapter01.xml
Done
```

`xsl/` contains XSLT Customization Layers for the various types of outputs. By default, the `Rake` task only uses a customization layer for the PDF output. A customization layer is simply a way for you to define your own style properties for your output. The customization layer imports the original Docbook XSLT file, and then you redefine your own transformations.

`images/` will contain any images you plan to reference in your document. It's a good idea to organize this by chapter.

`cover/` holds the cover for your book. Put a file called `cover.pdf` file there and you can use a special task to prepend the cover file to your book.

`book.xml` is a sample starter template that shows you how to compose a simple book. You can build this into a PDF right away, and you can modify it and use it as the basis for your book, or just leave it around as an example.

`chapter01.xml` is an example chapter for a book. It's included within `book.xml` so you can see how you might break apart a book into multiple files.

`Rakefile` is the script used to transform your document into an actual publication. See Section 4, “Generating Output from Docbook” on page 6 to see how that works.

`xsl/*.xml` is the aforementioned customization layers for the various output formats.

Open up the `book.xml` file and modify its contents to write your book. Then render a new output, as discussed in Section 4, “Generating Output from Docbook” on page 6.

## 4. Generating Output from Docbook

Docbook is designed to keep the content separate from the presentation. Using XSLT, we can transform any Docbook document into a PDF file, an ePub file for ebook readers, or even HTML files for use on a web site. This package provides easy access to the most popular transformation methods

### PDF and RTF documents

Generating a PDF or an RTF document is a two-step process. The XML document needs to be processed and transformed into an XML-FO document which is ready for print. Then it needs to be converted into either RTF or PDF by another process. This package uses Apache FOP to handle this step.

### Using Rake to build a book

When you generate a new project with `generate book mybook` you get a `Rakefile` script written in the Ruby programming language. This file includes recipes that we can use to build books.

### Path to the Build Chain

When you generate a project, the generator will put the absolute path to the Docbook build chain in your Rakefile. However, you can override this by setting an environment variable. On Linux or Mac OS, this would be:

```
export SHORT_ATTENTION_SPAN_DOCBOOK_PATH=/usr/local/docbook
```

On Windows, you'd do

```
SET SHORT_ATTENTION_SPAN_DOCBOOK_PATH=c:/docbook
```

Or you would add the Environment variable permanently through the the Control Panel, or alternatively, use the Windows installer for the Short Attention Span Docbook.

### Building

By default, this script can generate HTML, PDF, and RTF outputs. Windows users can also use it to generate Windows Help (CHM) documents if they have the `hhc.exe` file from the HTML Help Workshop installed on their system.

The build script is easy to use. You simply specify the name of the output file you want to generate. To make things simple, the script uses a file naming convention that matches your original docbook xml file.

For example, if you have a file called `book.xml`, you'd be able to do the following transformations:

`rake book.html` to generate an HTML file of your book.

`rake book.rtf` to generate an RTF file of your book.

`rake book.pdf` to generate a PDF file of your book.

### Validation

By default, the script validates your file against the RelaxNG schema definition file. Sometimes you may want to work offline or you just know you're that awesome and don't need to validate. The script takes a `VALIDATE=false` option that skips the validation step.

```
rake pdf VALIDATE=false
```

If you have problems with your document, you may get unexpected results, so use this with caution.

## 5. Additional Customizations

There are some additional customizations you can take advantage of in this build chain as your needs change.

### Drafts

As you're writing, you may want to inform your readers and reviewers that you are in Draft mode.

```
rake pdf DRAFT=true
```

This turns on a background overlay and enables display of any remark tags in your document.

### Covers

Covers are supported for PDFs and ePubs. Unfortunately, the process is slightly different for each.

#### PDF covers

Create a book cover in your favorite graphics program, export it as a PDF called `cover.pdf`. Place this file in your project's `covers/` folder and it will be used when you build the book as a PDF.

#### ePub Covers

To add a cover to an ePub book, you'll need to add a JPG version of your book cover to `covers/cover.jpg`, and then you'll add this markup to your book's front matter:

```
1 <cover>
2   <mediaobject>
3     <imageobject>
4       <imagedata fileref="cover/cover.jpg"/>
5     </imageobject>
6   </mediaobject>
7 </cover>
```

The processor will pick up the cover file and build it into the book.

### Preprocessing

If you need to perform some work before building your book, you can define a `:preprocess` task in your project's `Rakefile`.

To make things easier, several environment variables are available to you in your task.

**Table 1. Preprocessor variables**

Variable	Description
ENV["SOURCE_FILENAME"]	The name of the source XML document (book.xml)

Variable	Description
ENV["TEMP_FILENAME"]	The temporary XML file created by the build chain. It's copied from the source file so you can preprocess it and not alter your original file.
ENV["OUTPUT_FILENAME"]	The name of the output file. (book.pdf)
ENV["FORMAT"]	The format. "pdf", "html", "rtf", "epub"

So, you can preprocess your document easily.

```
task :preprocess do
  `xmllint #{ENV["TEMP_FILENAME"]}`
end
```

## Postprocessing

Postprocessing works exactly the same way. You have access to the same environment variables as the preprocessor task. You could use this to automatically publish the book to a web site, for example.

```
task :postprocess do
  Net::SCP.start("remote.host", "username", :password => "passwd") do |scp|
    # synchronous (blocking) upload; call blocks until upload completes
    scp.upload! "book.pdf", "files/my_awesome_book/awesome.pdf"
    scp.upload! "book.epub", "files/my_awesome_book/awesome.epub"
  end
end
```

When you run `rake book.pdf`, you'll automatically get the cover attached to the book.

## Forcing a Build

If you change a chapter file but not the book.xml file, the Rake task may not work as it hasn't detected a change in the file. This is just how Rake works, but you can force things to work by either deleting your existing PDF file or "touching" the `book.xml` file. If you're making a book called `book.pdf` you can use the command `rake build` which will *always* build, no matter what.