DAPS AsciiDoc Quick Start Guide

Table of Contents

AsciiDoc is a widely used markup language which, among other features, natively supports conversion to DocBook. Starting with the DAPS 3.0, DAPS supports AsciiDoc sources. AsciiDoc sources are converted to DocBook and then processed the same way as DocBook sources. Projects with AsciiDoc sources are handled the same way as regular DocBook projects. Therefore, the full range of output formats supported by DAPS is supported for AsciiDoc sources (HTML, single HTML, PDF, EPUB, Mobipocket, plain text). This quick start guide focuses on using DAPS with AsciiDoc sources and requires basic DAPS knowledge and AsciiDoc writing experience. For More Information contains links to documentation on DAPS and AsciiDoc.

The complete DAPS documentation is available online at https://opensuse.github.io/daps/.

1. Motivation

Over the past years, a lot of software documentation projects have moved from DocBook to AsciiDoc. With products and solutions becoming increasingly complex, documentation needs to rely more and more on external contributors. The hurdle to contribute is significantly lower when using AsciiDoc as it is with DocBook. Such a move not only requires converting the DocBook sources to AsciiDoc, but also changing the project setup, the toolchain and writing new stylesheets. The AsciiDoc support in DAPS saves you from switching to a new toolchain and new stylesheets. Whether you convert an existing DAPS project from DocBook to AsciiDoc, or whether you have used DAPS before and are starting an AsciiDoc project from scratch, DAPS lets you:

- use existing XSLT stylesheets (for converting DocBook into PDF, HTML, etc.)
- use the same `daps` commands as with DocBook projects
- use the same project setup as with DocBook projects

However, DAPS cannot handle the conversion to AsciiDoc as the new source format.

2. Requirements

To use AsciiDoc sources with DAPS, you need to install the following packages:

- DAPS 3.0 or better
- `asciidoctor`

To install DAPS follow the DAPS installation instructions. For installing `asciidoctor` we recommend to use packages available for your Linux distribution.

3. Setting up a Project

An AsciiDoc project has got the same directory structure as a regular DocBook project managed with DAPS (see Directory Structure for more information. However, there is one exception: For a DocBook project, images need to be saved in a directory structure where each image format has its own directory. To be able to also easily process AsciiDoc sources from a DAPS project with `asciidoctor` or to properly display the contents on, for example, GitHub, an AsciiDoc project can store all images in a single directory.

3.1. The Directory Structure

The following directory structure is recommended for AsciiDoc projects:

```
PROJECT_ROOT/ ①
     |--DC-* ②
     |--adoc/ ③
     |   |--MAIN.adoc ④
     |   |--\*.adoc
     |   |--images/ ⑤
     |   |  |--\*.dia/
     |   |  |--\*.fig/
     |   |  |--\*.jpg
     |   |  |--\*.png/
     |   |  |--\*.svg/
```

① The project's root directory (working directory) for the respective documentation project.

② One or more Doc Config files (DC files) defining your deliverables. Each DC file represents one book, article, or man page.

③ The directory containing all AsciiDoc source files. It nust be named `adoc/`. AsciiDoc files in subdirectories are ignored.

④ The MAIN file of the documentation project. It may be the only file, but may also include other AsciiDoc files via the `include` directive.

⑤ The Directory containing image files. Place all images in this directory. Subdirectories are ignored. Either specify the path to this directory in the DC-file with `ADOC_IMG_DIR` (recommended) or on the command line with `--adocimgdir`.

3.2. The Doc Config File (DC File)

A configuration file in the project directory. Each book, article, or man page in a project can have its own documentation configuration file (DC file). It defines several parameters for your documentation deliverable (for example, the MAIN file, layout variants, or whether to render the sources as article or book). Although not mandatory, creating a DC file for each deliverable is recommended. You can store deliverable-specific configuration options in the DC-file that you otherwise need to enter on the command line each time you call DAPS.

DC files need to be placed in the project root. The file names need to start with `DC-`. The rest of the file name is used to name directories and files created by DAPS. For example, if a DC-file is named `DC-adoc_test`, file names and directories will start with `adoc_test`.

Of the multiple parameters that can be set in the DC file, the only one required is MAIN. It points to the MAIN AsciiDoc file that you want to process. Create at least one DC file per book or article.

Example: A Typical DC File for an AsciiDoc Project

```
## Doc config file for the DAPS example document
## See /etc/daps/config for documentation of the settings below
##

# MAIN file (mandatory) ①
#
MAIN="MAIN-example.adoc"

#----- optional parameters -----

# Image directory  ②
#
ADOC_IMG_DIR="adoc/images"

# Type ③
#
ADOC_TYPE="book"

# Turn on postprocessing  ④
#
ADOC_POST="yes"

# Stylesheet directory  ⑤
#
STYLEROOT="/usr/share/xml/docbook/stylesheet/daps2013"

# XSLT Parameters for customizing the stylesheets ⑥
#
XSLTPARAM="--stringparam homepage=https://github.com/openSUSE/daps"
XSLTPARAM+="--param variablelist.as.blocks=1"
```

① Name of the main AsciiDoc file from the `adoc/` directory. You do not need to specify the full path, DAPS will automatically find it.

② Path to the image directory containing images referenced in the AsciiDoc sources. If you specify a relative path (recommended for portability), it needs to be specified relative to the directory containing the DC file.

③ Document type (`article`, `book`, `manpage`).

④ Parameter for applying the (default) post-processing XSLT stylesheet. It cleans up some of the DocBook XML constructs created by `asciidoctor`.

⑤ For a custom layout, use the STYLEROOT parameter to specify the path to the directory containing the custom XSLT stylesheets. If not specified, DAPS will use the default DocBook stylesheets. For an introduction on how to create custom stylesheets, refer to Customizing the DocBook Stylesheets (requires XSLT knowledge).

⑥ If the stylesheets allow customization via parameters, you can specify these parameters here.

The example above only shows a few options you can set. For a complete reference, refer to the

DAPS configuration file at `/etc/daps/config`. An overview of the AsciiDoc-specific configuration options is also available at [config-settings].

3.3. Image Linking and Processing

One of the core features of DAPS is automatic image conversion. Whether images need to be converted to grayscale for a printable PDF, whether `.dia` images need to be converted to SVG, or whether `.svg` images need to be converted to PNG—DAPS automatically takes care. You only need to make sure that the image is available in a supported format in the AsciiDoc images directory. Refer to Image Handling for details.

When linking an image in the AsciiDoc sources, use the attribute `:imagesdir:` in the document header. Always use the file name only. The path for `:imagesdir:` is best specified relatively (relative to the `adoc/` directory). Specifying `:imagesdir:` is optional, but recommended. It allows you to also process your AsciiDoc sources with tools other than DAPS.

Example: Linking Images in AsciiDoc

```
= Linking images

:author: John Doe
:imagesdir: images/

This is an example of how to link images in AsciiDoc documents processed with DAPS.

image::test.png[Test Image]
```

For the example above, the images need to reside in `adoc/images`. Possible sources for the images in `adoc/images` are:

- `adoc/images/test.dia`
- `adoc/images/test.fig`
- `adoc/images/test.jpg`
- `adoc/images/test.png`
- `adoc/images/test.svg`

The AsciiDoc source example requests a PNG file. If it exists, it will be used as is. If it does not exist, DAPS will search for an image with the base name `test` (`test.svg`, `test.dia`, or `test.fig`) and automatically convert it to PNG.

Unique Image Names

Because of the procedure described above, DAPS requires unique base names for images. If you would have two files with different content but the same base name, for example `test.svg` and `test.dia`, the outcome of an automatic conversion would be uncertain (could be either one of the two files). Therefore make sure to always use unique base names. To test for non-unique images, use the following daps command: `daps -d <DCFILE> list-images-multisrc`.

4. Creating Output with DAPS

After you have set up the directory structure, created AsciiDoc documents, and images (optionally), you can use DAPS to create PDFs, HTML, EPUB and other output formats. The command line syntax is the same as with DocBook projects:

Example: Creating HTML output

```
daps -d DC-_MYADOC_  html
```

Example: Creating Single Page HTML Output

```
daps -d DC-_MYADOC_  html --single
```

Example: Creating a PDF Directly from an AsciiDoc Source File

```
daps -m adoc/_MYADOC_.adoc pdf
```

For more information, refer to `daps --help`. Help on the DAPS subcommands is available with the command `daps <SUBCOMMAND> help`.

4.1. Creating Partial Output

If you want to create output for parts of the documents (for example, to send a single chapter or section for review), you do not need to create the whole document and cut out the parts you need. DAPS supports creating partial documents on section titles for any of the supported output formats. All that is required is an anchor preceding the section title:

Example: Section title with an Anchor

```
[[new_chapter]]
== Brand New Chapter

This chapter is brand new
```

This anchor, `new_chapter` in the example above, can be passed to DAPS with the `--rootid` parameter. The following example shows how to build a PDF only containing the chapter "Brand New

Chapter":

```
daps -d DC-_MYADOC_ pdf --rootid="new_chapter"
```

4.2. Spell Checking

DAPS also supports spell checking your sources. This is especially convenient, when your sources contain include files. A spellcheck with DAPS automatically checks all included files. DAPS supports the backends aspell and hunspell for spellchecking.

```
daps -d DC-_MYADOC_ spellcheck --lang=en_GB --spell-checker=hunspell
```

5. Common Errors and Pitfalls

Before generating output formats, DAPS internally converts AsciiDoc to DocBook XML. Because of that you may receive two different kind of errors when processing AsciiDoc sources:

AsciiDoc Errors

The conversion to DocBook XML is done by `asciidoctor`. In case there are syntax or structural errors in the AsciiDoc sources, you will get an appropriate error or warning message. It usually contains the number of the line where the error occured. These errors need to be fixed before any further processing can be done, DAPS will only proceed in case `asciidoctor` no longer produces errors or warnings.

DAPS Errors

After the internal conversion to DocBook XML is done, DAPS validates the resulting XML file to make sure it can properly be processed. Although this happens rarely, the conversion to DocBook XML might produce invalid XML. In this case, you will receive an error message claiming that "validation failed." The error message also contains the path to the file and the line number where the error occurred.

To fix the error, look at the text in the XML file at the given line and locate that text in the AsciiDoc sources. The error might be the result of some unsupported special `asciidoctor` macro or of an overly complex structure. Simplify the AsciiDoc code and try again.

5.1. Avoiding Mistakes

Section Titles (Headings)

- DAPS only allows a single Level 0 section at the beginning of the document
- Section titles need to be in correct order. It is not possible to skip a level. So `== Level 2` followed by `==== Level 4` will not work.

Multimedia

- DAPS currently does not support embedding videos

6. Linking Between Multiple Books

The `asciidoctor` built-in way to do inter-document cross references (via `link:file.html#ID[TITLE]`) does not work when generating the output via DocBook as DAPS does (and it only works for HTML output).

DAPS provides a way to allow links between multiple books (inter-document cross references) by converting the AsciiDoc sources into a DocBook set.

Requirements for the AsciiDoc sources

- The AsciiDoc sources need to combine all books into a single "master book" (with the doctype book) by making each book a "part" of the master book. In AsciiDoc, a part is introduced by a level 0 headline (`= HEADLINE`) that follows the initial headline of the master book. See the Asciidoctor documentation for details.
- The master book must only contain parts and no additional content outside these parts.
- Each part needs to begin with a unique custom ID (`[[UNIQUE_ID]]`)
- You need to be extra careful not to mess up the headline hierachy throughout the master book. If you are using prefix, appendix, glossary, etc. make sure you apply the special rules for parts regarding the headline levels (refer to the Asciidoctor documentation for details)
- Abstracts for chapters do not seem to work in parts (is this a bug?)
- All IDs you use need to be unique. Duplicated IDs will cause errors.
- Create a DC-file for the master book to be able to easily build it with DAPS (name it, for example, `DC-multipart`). MAIN needs to point to the top level AsciiDoc file that defines the master book.

Since tha AsciiDoc source now combines everything into a single book, you can link to all IDs anywhere in the book using `<<ID>>`.

Once the "master book" builds without errors and you can see a part for each book you want to build, proceed as follows to create stand-alone books in HTML, PDF, etc:

1. Copy the DC-file for the master book (name the copy, for example `DC-set`)
2. Add the line `ADOC_SET="yes"` to the copy of the DC file
3. Create a copy of the new DC-file (`DC-set`) for the first book represented by a part in the master book. Add the ID of the respective part to the DC-file by adding a line `ROOTID=<ID>` (where you replave `<ID>` with the respective part ID)
4. Repeat the previous step for each book represented by a part.

Using the respective DC-files you can now build the individual books. Using `DC-set` allows you to build the complete set.

Restrictions

- the individual books cannot contain parts
- the `article` document type is not supported with this setup

7. GitHub Integration

One of the advantages of using AsciiDoc as a source for documentation is its seamless integration with GitHub. GitHub not only renders AsciiDoc sources, but also allows to edit them directly in the Web interface. After having edited a document via the built-in editor, GitHub even automatically creates a pull request, depending on the repository setup. This improves the flow for external contributors.

To enable a smooth GitHub integration, a few adjustments in the AsciiDoc source code are required. Using the `ifdef` preprocessor macro, you can set directives specific to GitHub. The following adjustments are required:

Includes

In addition to this, including documents via the `include` directive needs to be worked around, since this is not supported by GitHub for security reasons. Create an anchor ( ) at the top-level of the documents you want to include and use the following syntax:

```
// Links for GitHub
ifdef::env-github[]
 <<FILE1.adoc#ANCHOR, TITLE1>>
 <<FILE2.adoc#ANCHOR, TITLE2>>
endif::[]
// includes for AsciiDoc processing
ifndef::env-github[]
 include::FILE1[]
 include::FILE2[]
endif::[]
```

Admonition Images

Admonition boxes (warning, tip, etc.) usually come with an icon. To display these icons, you need to tell GitHub where to find them.

```
ifdef::env-github[]
//Admonitions
:tip-caption: :bulb:
:note-caption: :information_source:
:important-caption: :heavy_exclamation_mark:
:caution-caption: :fire:
:warning-caption: :warning:
endif::[]
```

8. For More Information

Find links to useful online resources here.

8.1. DAPS

- User Guide
- Project Page
- Discussion

8.2. AsciiDoc

- AsciiDoc Cheatsheet
- AsciiDoctor Syntax Quick Reference
- AsciiDoctor Writer's Guide
- Getting Started with Asciidoc (SUSE Syntax and Best practice for Contributors)

8.3. Example AsciiDoc Code

- This document was written in AsciiDoc
- The DAPS test documents contain example with complex, nested structures

8.4. Converting DocBook to AsciiDoc

- DocBookRx

Appendix A: Appendix

A.1. Command Line Options and Configuration Settings for AsciiDoc

DAPS supports the following AsciiDoc-specific command line options. These are global options and need to be specified before the subcommand. Additional configuration settings can be made in the DAPS configuration files (for example in a DC file).

Table 1. Command Line Options/Configuration Settings for AsciiDoc

CLI Option	Config Setting	Explanation

`--adocattr`	`ADOC_ATTRIBUTES`	Define or delete AsciiDoc document attributes. To overwrite an attribute already defined in the AsciiDoc document, use NAME=VALUE, or just NAME for attributes without a value. To delete a value set in the document use NAME!. To set a value that is not already set in the document, use NAME=VALUE@. Refer to the AsciiDoc documentation for more information.
		
		Examples:
		
		`daps -d DC-adoc --adocattr "author=John Doe" \`
		
		`--adocattr "revision=beta1"`
		
		You may specify this option multiple times to set more than one attribute.
		
		`ADOC_ATTRIBUTES="--attribute author=myself"`
		
		`ADOC_ATTRIBUTES+="--attribute "revision=beta1"`
		
		Note that when using the config file option, you always have to prefix the KEY=VALUE pair with `--attribute`. To specify more than one value, use the `+=` notation for subsequent values.

`--adocimgdir`	`ADOC_IMG_DIR`	Specify a directory for the images used in the AsciiDoc sources. Must contain all images, subdirectories are ignored. Examples:
		
		`daps -d DC-adoc --adocimgdir "adoc/images"`
		`ADOC_IMG_DIR="adoc/images"`

CLI Option	Config Setting	Explanation
	ADOC_BACKEND	Specify whether to use asciidoc or asciidoctor. Specifying this parameter is usually not necessary, because DAPS will automatically check which program is installed. If both are installed, asciidoctor will be preferred. Only required if the back-end binary is not in your path or if you prefer asciidoc over asciidoctor. Examples: ADOC_BACKEND="/home/doc/asciidoctor/asciidoctor" ADOC_BACKEND="/usr/bin/asciidoc"
	ADOC_POST	If set to "yes", the XML produced from the AsciiDoc sources will be processed again before DAPS generates output. This can be used to change or clean up the XML. The stylesheet to be used can be specified via ADOC_POST_STYLE. By default this is set to "no". Example: ADOC_POST="yes"
	ADOC_POST_STYLE	Stylesheet to post-process the XML produced from the AsciiDoc sources. Requires ADOC_POST to be set to "yes" (will be ignored otherwise). By default it is set to a stylesheet shipped with DAPS that does some cleaning up.

CLI Option	Config Setting	Explanation
	`ADOC_SET`	If set to "yes" a multipart book in AsciiDoc will be converted to a set in DocBook. Each part of the original sources will becaome a book in DocBook. Require the AsciiDoc sources to only contain parts (one for what is to become a book) and no extra contant. By default this is set to "no". Example: `ADOC_SET="yes"`
	`ADOC_TYPE`	Same option as you would set by --doctype with `asciidoc` or asciidoctor. Valid values are "article", "book", and "manpage". Do not use "inline" as it will not work with DAPS. Setting "manpage" requires manpage-specific content (refer to the AsciiDoc documentation). Otherwise processing the source will fail. This setting will override the :doctype: definition in the AsciiDoc source document.

The following subcommands support AsciiDoc-specific commands:

`list-srcfiles`

This subcommand lists all files that are used to build the document, including images and the DC file. It supports several options for filtering the output. To restrict the results to only AsciiDoc files, use the option `--adoconly`. To exclude AsciiDoc files from the results, use the option `--noadoc`. Examples:

```
tux > gdaps -d DC-adoc_test list-srcfiles
/home/doc/documents/DC-adoc_test
/home/doc/documents/adoc/appendix.adoc
/home/doc/documents/adoc/book.adoc
/home/doc/documents/adoc/part_block.adoc
/home/doc/documents/adoc/part_inlines.adoc
/home/doc/documents/images/src/dia/dia_example.dia
/home/doc/documents/images/src/fig/fig_example.fig
/home/doc/documents/images/src/jpg/jpg_example.jpg
/home/doc/documents/images/src/png/png_example.png
/home/doc/documents/images/src/png/png_example2.png
/home/doc/documents/images/src/svg/svg_example.svg

tux > gdaps -d DC-adoc_test list-srcfiles --adoconly
/home/doc/documents/adoc/appendix.adoc
/home/doc/documents/adoc/book.adoc
/home/doc/documents/adoc/part_block.adoc
/home/doc/documents/adoc/part_inlines.adoc

tux > gdaps -d DC-adoc_test list-srcfiles
/home/doc/documents/DC-adoc_test
/home/doc/documents/images/src/dia/dia_example.dia
/home/doc/documents/images/src/fig/fig_example.fig
/home/doc/documents/images/src/jpg/jpg_example.jpg
/home/doc/documents/images/src/png/png_example.png
/home/doc/documents/images/src/png/png_example2.png
/home/doc/documents/images/src/svg/svg_example.svg
```