ACMEdoc Fab User Manual

Table of Contents

You are seeing a TOC and copyright line above because this is the PDF/book edition.

Welcome to the ACMEdoc Fab User Manual!

Here you will learn to develop great docs ACMEdoc Fab as a user.

The content above appears in all editions, but some of it is variable. Many of the strings (e.g., *ACMEdoc Fab* and even *develop great docs*, etc) are variables (`{portal_product}` and `{user_verb}`) are dynamically substituted with AsciiDoc attributes at build time.

Introduction to the Standard Package

Our standard offering is pretty simple, quite by design. It has some feature limitations that we'll tease you with throughout this manual, but there's still quite a bit you can do with it.

The above text is from by a dynamically included file using:

```
include::portal/intro_portal_{portal_slug}.adoc[]
```

This document contains lots of jargon. See the Jargon Guide if you get lost.

Product Overview

This is an overview of ACMEdoc Fab for users.

The terms *ACMEdoc Fab* and *user* are variables (`{portal_product}` and `{user_role}`).

The above content is drawn from a file generated by LiquiDoc. It starts as semi-structured data (see `_data/products.yml`). This AsciiDoc file (`src/portal/product-overview.adoc`) selectively calls sections of that generated file (`build/includes/built/product-info.adoc`), depending on which products the current user role needs to see.

Framework Structure

An implementation of LiquiDoc CMF such as ACMEdoc starts with a core architecture made up of the directories and files listed below.

_configs/	This is where we tell our tooling how to do what. The `_configs/` directory usually contains at least one `build\*.yml` file that instructs a build routine as well as an `asciidoctor.yml` file for establishing global AsciiDoc/Asciidoctor settings, and optionally a `jekyll.yml` file to stand in as the `_config.yml` for Jekyll operations.
_data/	Where we store small-data files.
_templates/	This directory is for storing templates used in preprocessing.
build/	A directory created and written to during the build process.
src/	This is where *most* of our content will come from.
src/*index*.adoc	The defining file for book builds and the homepage and something of a map for site builds. This file sould have `index` in its filename but should also be descriptive of what it produces.
src/includes/	Snippets and partials go in here. This is where we put highly reusable and often partly dynamic text content. Snippets should not have headers. They may be collected into files by type or subject, distinguished by AsciiDoc include tags in their comments.
src/topics/	Topics are chunks of content that could stand alone, for instance as a web page, but that also might be reused in different contexts, including as a component of a larger page or other document, such as a book. All topic files should have a level-one header (`=`).
src/[content]/	A freely named directory for your main content, however organized. Often called `pages/` to comport with Jekyll conventions.
theme/	Files for defining layout and style for *publishing* go here. Not to be confused with `_layouts/`, which holds similarly purposeful files for structuring content in preprocess operations.

In the PDF edition of this document, the above appears as a table; in the website edition, it is a definition list. Both are drawn from the same source (`_data/framework-structure.adoc`), which is parsed using the generic templates `terms-to-table.asciidoc` and `terms-to-dlist.asciidoc`, each generating a file in the ephemeral directory `build/includes/built/` (those being `framework-strcture-dlist.adoc` and `framework-structure-table.adoc`).

Appendix A: Jargon Guide

This is the full list of specialized terms used in this product documentation. They are also generated as JSON at `/data/terms.json` so we can highlight them in the text when we get to it. This is just to show the power of storing data in flat files reusable throughout product docs.

artifact

A digital package (file or archive) representing a discrete component of a product. Here we use *artifact* to describe a descrete instance of output, such as a single HTML or PDF file, or a Jekyll website or Deck.js slide presentation.

AsciiDoc

Dynamic, lightweight markup language for developing rich, complex documentation projects in flat files. (Resource)

Asciidoctor

Suite of open source tools used to process AsciiDoc markup into various rendered output formats. (Resource)

build

The (usually automated) series of actions necessary to compile and package software or documentation.

Liquid

Open source templating markup language maintained by Shopify (Resource)

YAML

a slightly dynamic, semi-structured data format for nested data (Resource)