ACMEdoc Developer Manual

Table of Contents

Copyright © 2018 ACMEdocOps Collective

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

Welcome to the ACMEdoc Developer Manual!

Here you will learn to extend and customize ACMEdoc as a developer.

> 💡 The content above appears in all editions, but some of it is variable. Many of the strings (e.g., *ACMEdoc* and even *extend and customize*, etc) are variables (`{portal_product}` and `{user_verb}`) are dynamically substituted with AsciiDoc attributes at build time. The content is included from a common file, like so:
>
> ```
> include::includes/intro_portal_common.adoc[]
> ```

As a **developer**, you will have access to a sandboxed version of the product for configured for development and testing. Your Manual includes documentation of all Basic and Premium features as well as information required to extend the product.

> 💡 The above text is from a dynamically included file using:
>
> ```
> include::portal/intro_portal_{portal_slug}.adoc[]
> ```
>
> Where `{portal_slug}` is a variable defined in a block of a YAML file (`_configs/portals.yml`) that is only passed to one designated portal during the build config file (`_configs/build-docs.yml`), where it is designated as `data: _data/portals.yml:portal-3`.

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

Product Overview

This is an overview of ACMEdoc for developers.

> The terms *ACMEdoc* and *developer* 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-table.adoc` and `framework-structure-dlist.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)

How this glossary was sourced

The contents of this appendix were drawn from a semi-structured "small data" file (`_data/terms.yml`), which looks a little something like this:

terms.yml

```
- term: Asciidoctor
  def: Suite of open source tools used to process AsciiDoc markup into various
rendered output formats.
  ref: http://asciidoctor.org
- term: build
  def: The (usually automated) series of actions necessary to compile and package
software or documentation.
  aliases: [build procedure]
- term: Liquid
  def: Open source templating markup language maintained by Shopify
  ref: https://help.shopify.com/themes/liquid
```

This data is combined with the template at `_templates/liquid/terms.asciidoc` to build an AsciiDoc formatted file (`build/includes/built/terms.adoc`).

GETTING META!

You wanna know what's really friggin cool? The example YAML snippet above is drawn directly from the *actual file* at build time. That means the above will always display the exact, actual content of the YAML it is referencing.

Appendix B: NOTICE of Packaged Dependencies

The following open source packages are fully or partially included with ACMEdoc.

Package	Jekyll Documentation Theme
License	MIT
Author	Tom Johnson
Website	https://github.com/tomjoht/documentation-theme-jekyll

Package	M+ OUTLINE FONTS (M+ TESTFLIGHT 058)
License	unlimited
Author	M+ Fonts Project
Website	http://mplus-fonts.osdn.jp/about-en.html

Package	Noto Fonts
License	SIL OFL
Author	Google i18n
Website	https://www.google.com/get/noto/

Package	Font Awesome
License	SIL OFL 1.1
Author	Fonticons, Inc
Website	https://fontawesome.com/

How this appendix is sourced

This file is not sourced as YAML but rather as plain text for a couple of reasons. First of all, this file is traditionally named `NOTICE` and kept prominently in the root directory in open-source projects.

More importantly, since this file *must* be present in this open-source project where it is distributed (GitHub), we do not want to duplicate the source. So we store it in a very human-readable plaintext format, then we extract data from it during the parsing step of the build routine.

Here is the NOTICE file in its entirety:

ACMEdoc NOTICE file

```
The following permissively licensed software is packaged with ACMEdoc.

Jekyll Documentation Theme | MIT | (c) Tom Johnson |
https://github.com/tomjoht/documentation-theme-jekyll

M+ OUTLINE FONTS (M+ TESTFLIGHT 058) | unlimited | (c) M+ Fonts Project |
http://mplus-fonts.osdn.jp/about-en.html

Noto Fonts | SIL OFL (https://github.com/googlei18n/noto-
fonts/blob/master/LICENSE) | (c) Google i18n | https://www.google.com/get/noto/

Font Awesome | SIL OFL 1.1
(http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL) | (c)
Fonticons, Inc | https://fontawesome.com/
```

And here is the section from `_configs/build-docs.yml` that instructs the data extraction:

From build-docs.yml

```
# Generate structured NOTICE output
- action: parse
  data:
    file: NOTICE
    type: regex
    pattern: ^(?<name>.*)\s\|\s(?<license>[a-zA-Z0-
9\.\s]+)\s(\(http(?<license_url>.*\)))\s)?\|\s\(c\)\s(?<author>.*)\s\|\s(?<package
_url>.*)$
  builds:
    - template: _templates/liquid/dependencies.asciidoc
      output: build/includes/built/dependencies_table.adoc
```