Technical Docs

Formalhauthorized

Version dev, 2025-10-13

Table of Contents

Antora

This is practice website to try out antora and asciidoc.

Antora Intro

This is personal notes from docs.antora.org. Just things I find useful.

Content Sources

Symlinks

Thing from here.

*commands for *nix*

```
ln -s target.adoc link.adoc
```

If They are not in the same dir: 1. go to the directory when the link should be. 2. specify the relative form

When the target is not in the same dir

```
ln -s ../../path/to/target.adoc link.adoc
```

Component Versions

These are fmor the `Configure Component Sources` section.

versions

You can use refname as version: refname-projection.

Below is the same example from the docs

refname projection

```
name: colorado
version:
  v(?<version>+({0..9}).+({0..9})).x: $<version>
  feature/(*)/*: $1
```

attributes in antora.yml

Set asciidoc attributes in `antora.yml`:

example from docs.antora.org

```
name: light
title: Data Light
version: '2.3'
asciidoc:
  attributes:
    table-caption: Data Set
    hide-uri-scheme: ''
    page-level: Advanced
    page-category: Statistics; Cloud Infrastructure
nav:
- modules/ROOT/nav.adoc
```

Also it is useful to know one can "soft set" attributes by attaching the `@` modifier to the end. So that they can be overridden from a page.

example of soft set

```
asciidoc:
  attributes:
    table-caption: Data Set@
    hide-uri-scheme: '@'
```

Summary:

1. Hard setting a component version attribute using a string value with no trailing @ modifier (e.g., value or '') prevents a page from changing or unsetting the attribute.
2. Soft setting a component version attribute using a string value with a trailing @ modifier (e.g., value@ or '@') allows a page to change or unset the attribute.
3. Hard unsetting a component version attribute using the tilde symbol (~) prevents a page from setting the attribute.
4. Soft unsetting a component version attribute using a false value allows a page to set the attribute.

Resource ID

resource ID coordinates syntax and order

A resource ID contains five coordinates:

1. version
2. component
3. module
4. family
5. file

They are put together in the following manner:

```
version@component:mudule:family$file.ext
```

examples xref with resource id

```
xref:v0.1@technical-docs::index.adoc[back to v0.1 index]
xref:index.adoc[index of the current version]
```

The above examples will give the link below:

- back to v0.1 index
- index of the current version

Note that the Family coordinate omits "s" from the family’s name, like `page$`, `image$`, `partial$`, etc. Also, the xref does not require if it is from a pages family. When it is required refer to thie docs.

So, if you want to have assets for all the content sources, one way to do it is `assets` content source. Then add it to the `antora-playbook.yml`.

example from other content source

```
 image:assets::favicon-32x32.png[test picture: a green cat alien from formalhaut]
```

The example above .

Antora UI

Supplemental UI

To make a few adjustments to the UI bundle. To see more, visit the antora docs.

The supplemental UI is overlaid onto the files in the loaded UI bundle. (either add or replace)

Under `ui` category, put `supplementa_files` key. The key accepts either (exclusive)

- path of a directory
- a map of virtual files

Supplemental_files with directory

You can specify with a relative or absolute filesystem path.

For relative paths:

- `~`: relative to the user’s home directory
- `.`: relative to the location of the playbook file

- `~+` or no prefix: relative to the current working directory

Example of supplementa_files with directory

```
ui:
  supplemental_files: ./supplemental-ui
```

The files in the specified directory will be added to the loaded UI bundle. The directory is assumed to be the root

Supplemental_files with virtual files

If the given value is an array, it is assumed to be a virtul file.

A virtual file entry has two keys: - `path`: relative path of the file in the UI - `contents`: three possibility - omit the `contents` key: create an empty file - file path: a single line ends with a file extension - the contents of the file will be assigned to the virtual file - contents: - use the given value as the contents

Example for the file path

```
ui:
  supplemental_files:
  - path: partials/head-meta.hbs
    contents: ./supplemental-ui/partials/head-meta.hbs
```

Example for the contents

```
ui:
  supplemental_files:
  - path: partials/head-meta.hbs
    contents: |
      {{#with site.keys.googleSiteVerification}}
      <meta name="google-site-verification" content="{{this}}">
      {{/with}}
```

Static files

Publish files to the root of the published site: static files

The static files are identified using `ui.yml`

Example for favicon

```
ui:
  supplemental_files:
  - path: favicon.ico
    contents: ./supplemental-ui/favicon.ico
  - path: ui.yml
    contents: |
      static_files:
      - favicon.ico
```

github troubleshoot

github pages no css

Github Pages runs all files through `Jekyll`, which will remove all files with `_`. This will remove all the css and js files generated with antora.

To disable this `Jekyll` feature, you have to put `.nojekyll` file to the root of the published site.

For more detailed explanation, checkout this page.

One way suggeted is to use the supplemental UI:

antora-playbook.yml that adds .nojekyll file

```yaml
ui:
  bundle:
    url: <url-of-bundle-goes-here>
  supplemental_files:
  - path: ui.yml
    contents: |
      static_files:
      - .nojekyll
  - path: .nojekyll
```

For a brief explanation:

1. the first item under `supplemental_files` adds `ui.yml` file:

   ui.yml added by supplemental_files

   ```yaml
   static_files:
   - .nojekyll
   ```

   Note that the `static_files` means that the list of files will be placed in the root of the site. So, it will place the `.nojekyll` file on the root of the site

2. the second item under `supplemental_files` adds `.nojekyll` file, which is empty

Pages

Here are all about pages section.

file name

The following files in the pages family directory will be ignore it

- a file name begins with a dot (.)
- a file name ends without a file extension

The following files will be cataloged but will not published:

- a file name starts with an underscore (_):

Recommended (some are required) file name style:

- ends with `.adoc` file extension
- only URL-compliant characters
- all lowercase
- no space (` `)

header and body

```
= Page Title
Optional Author <option@auth.com>; Directly Under Title <optional@email.com>; Output
to MetaTag
:description: A description of the page stored in an HTML meta tag. \
Break description into multiline if too long with `\`.
:keywords: comma-separated values, stored, in an HTML, meta, tag
// comment
// AsciiDoc Attributes
:sectanchors:
:url-repo: https://my-git-repo.com
:page-tags: name of a tag, name of a tag

// a blank line ends the page header
page body
```

- sectanchors, url-repo: optional asciiDoc attributes

page title

About page title:

- can contain text formatting
- page title **must not** contain resource references (xref, image, etc)

asciiDoc document attributes

To circulate information in and out of pages, such as settings, styles, metadata.

Two kinds of attributes:

- built-in attributes

    - defined by asciidoc
    - intrinsic attributes: defined by Antora
- custom attributes
    - defined by author

Page attributes are prefixed with `page-`. They can be accessed using `page.attributes.<name>` in a UI templates.

Types of page attributes:

- custom
- predefined
- intrinsic

Built-in attribute

Here are examples of built-in attributes

Built-in attribute

```
= Page Title
:toc:  // set the table of content
:!toc:  // unset the table of content
:table-caption:
:sectanchors:  // adds an anchor to each section title
:sectanchors!:  // To unset
:!sectanchors:  // To unset
:xrefstyle: short // short, full, basic
```

Some AsciiDoc attributes are **not** applicable in the Antora environment:

- data-uri
- allow-uri-read
- docinfo
- linkcss
- noheader
- nofooter
- webfonts
- outfilesuffix
- imagesdir: set by Antora, cannot be overriden

Custom attribute

To store frequently used text and URLs.

Attribute name:

- only contains word characters (A-Z, a-z, 0-9, or _) and hyphens
    - no dot (.) or a space
- begins with a word character
- recommended
    - to use only lowercase letters
    - avoid starting the name with a number

Attribute values can:

- be any inline content
- contain line breaks, but only if an explicit line continuation (+) is used.

Custom attribute

```
= Page Title
:url-repo: https://gitlab.com/antora/demo/docs-site


Use the set value like {url-repo}
```

Precedence rules

The Attributes which are hard set or hard unset in the playbook or component escriptor (antora.yml) take precedence over attributes defined in a page.

Page attributes

The name of the attribute must begin with `page-`. If you need to access the attribute from the UI template, you need the prefix.

Page attribute examples

```
= Page Title
:page-category: DevOps
:page-edition: Enterprise
:page-custom-attribute: custom value
:product-name: Name of My Product
:page-product-name: {product-name}
// attribute reference is resolved immediately

Main content.
```

Custom page attributes should be defined in the page header and start the name with `page-`.

Page attributes can also be defined site-wide in the Antora playbook or per component version in the component version descriptor.

predifined page attribute

Predefined Page attribute examples

```
= Page Title
// configure alternate resource IDs
// value: comma-separated list of resource IDs
:page-aliases: old-page-name.adoc, v0.1@technical-docs::test-page1.adoc, assets::test-
page2.adoc
// specify which UI template to apply
:page-layout: home
```

Access page attributes from the UI template

Here is an example for `toclevels`

antora-playbook.yml or antora.yml

```
asciidoc:
  attributes:
    page-toclevels: 3@
    // trailing @: to be able to override in a page header
    page-pagination: ''
    // hardset (cannot be turned off by a page)
```

page header

```
= Page Title
:page-toclevels: 2
```

The page attributes can be referred like `tocleves` (without prefix `page-`):

use toclevels

```
{{page.attributes.toclevels}}
```

use pagination

```
{{#unless (eq page.attributes.pagination undefined)}}
<nav class="pagination">
...
</nav>
{{/unless}}
```

access non-page attribute from the UI model

```
{{#with (resolvePage page.relativeSrcPath model=false)}}
{{./asciidoc.attributes.policy-number}}
```

```
{{/with}}
```

intrinsic attributes

Automatically assigned runtime environment information by Antora. Read-only and should not be reassigned. Not visible to the component descriptor (antora.yml)

intrinsic environment attributes:

intrinsic environment attributes:

- env=site
- env-site
- site-gen=antora
- site-gen-antora

Example usage

```
ifndef::site-gen-antora[]
include::local-preview-settings.adoc[]
endif::[]
```

site and config attributes

site and config attributes

- attribute-missing=warn
- !data-uri
- icons=font
- sectanchors
- source-highlighter=highlight.js
- site-title
- site-url

highlight.js is the only syntax highlighter that Antora currently supports.

You can check the list of intrinsic page attributes here.

Access intrinsic page attributes

To access the values:

- In AsciiDoc: use the attribute reference syntax (`technical-docs`)
- In the UI model: use a template variable (`page.attributes.component-name`)

Example in a page

```
This page belongs to the *{page-module}* module in the *{page-component-title} {page-
component-version}* component version.
```

Some useful attributes

Here are some examples of attributes

Set ref text

```
= My Page
:reftext: This text will be used in xref
:navtitle: This will be used for nav
```

If they are not set, the default is page title.

Create a redirect with page-aliases

```
= Title of Target Page
:page-aliases: source-page-filename.adoc, version@component:module:source-page-
filename.adoc
```

Page aliases split across multiple lines

```
= Title of Target Page
:page-aliases: source-page-filename.adoc, \
version@component:module:source-page-filename.adoc
```

One can redirect a deleted page to a custom 404 page.

xref

cross reference

structure of an xref macro

```
 xref:version@component:module:file.adoc#elementID[optional link text]
```

If the version is not specified (and target page and current page belong to different docs components), Antora uses the latest version.

When the fragment (elementID) was specified, the link text is **not** automatically populated. You need to specify the link text for a fragment.

Xrefs in a navigation has one additional feature. Navigation files use the `navtitle` value to populate missing link text. If the `navtitle` is not set, then `reftext` value is used. By default Antora assigns a page’s title to the `reftext` attribute at runtime.

example same component different version different module

```
xref:5.2@get-started:tour.adoc[excursions]
```

The same component (colorado) was omitted, The module `get-started` was specified.

include

Example of inclue

```
 include::version@component:module:file-coordinate-of-target-page.adoc[optional
attribute]
```

The barckets of the include can contain optional attributes, such as lines, tag, or tags.

To include content by line ranges, refer to this page.

Include content by line ranges

```
include::antora-docs.adoc[lines=10..17]
include::antora-docs.adoc[lines=10..17;22..26;120..-3]
```

You may find that the include directive resolves the target page when the ./ token isn't used in the file coordinate. However, we strongly recommend using the ./ token when referencing a target page that is stored in the same subdirectory as the current page.

Include placement

```
A paragraph in the page.

include::resource-id-of-target-page.adoc[tag=value]

A line of content.
include::resource-id-of-target-page.adoc[]
Another line of content.
```

The second include will be a block with the line above and below.

page-partial attribute

Abot the order of process. See docs

In short, the `page-partial` attribute in the included page instructs Antora to retain the AsciiDoc source until all pages have been converted. It is (soft) set by default, so it will just work.

It may, however,increase the peak heap usage. So, one can revert it in playbook file so that the page-partial is no longer set globally:

playbook.yml

```
asciidoc:
  attributes:
    page-partial: false
```

You **must** set the page-partial attibute on any page to be included

Shared page

```
= Shared Page
:page-partial:

Page contents.
```

Other page

```
 include::shared-page.adoc[]
```

Antora snippets

These are just taken from the official docs.antora.org.

images

docs

Let’s say you have an image file in `assets` component.

image location

```
assets
|-- README.md
|-- antora.yml
`-- modules
    `-- ROOT
        `-- images
            `-- example-image.png
```

Image examples

```
 below is a block image example.

 image::assets::example-image.png[]

 You can also put inline image::assets::example-image.png[].

 The image can link using xref: image:assets::example-image.png[The image alt
```

```
text,xref=antora-pages.adoc].

 The image can link using xref: image:assets::example-image.png[The image alt
text,30,40,xref=antora-pages.adoc].

 image::assets::example-image.png[The image alt text,50,50,xref=antora-pages.adoc]
```

The image with link: .

The image macro has positional attributes (optional). Assign them before setting named attributes such as `xref`. The example above is `alt`, `width`, and `height`.

Current page deep link

```
image::panorama.png[xref=#elevation]

The range can be traversed by Cottonwood Pass, Independence Pass, or Hagerman Pass.

[#elevation]
== Pass elevations

The passes usually open in late spring.

One can also link to self:
image::panorama.png[link=self]
```

partials

Reusable

A partial is not required to be an AsciiDoc file.

Example partials

```
[plantuml,my-schema,svg]
....
include::partial$diagrams/my-schema.puml[]
....
```

attachment

Unlike pages, attachments don’t have default reference text, so it’s always good to specify link text to ensure a good reader experience.

Example attachment

```
Download xref:attachment$practice-project.zip[the sample project] to try it out!
```

asciidoc

AsciiDoc primer

AsciiDoc Primer note from antora docs.

header

Header examples

```
= Page Title
:description: A description of this page.
:docrole: home
```

Text emphasis, punctuation, typography

bold example

```
Make a *word* or *a phrase of text* bold in a single set of asterisks.
Bounded characters with a set of double asterisks. Like cha**act**ers, not
cha*act*ers.
```

Make a **word** or **a phrase of text** bold in a single set of asterisks.

Bounded characters with a set of double asterisks. Like cha**act**ers, not cha*act*ers.

The style of bold text depends on the font and the CSS tyles for `<strong>` HTML tag.

example mixed formatting

```
`*_monospace bold italic phrase_*` & ``**__char__**``acter``**__s__**``
```

italic example

```
An italic _word_, and an italic _phrase of text_.

Italic c__hara__cter__s__ within a word.
```

The style of bold text depends on the font and the CSS tyles for `<em>` HTML tag.

monospace example

```
A monospace `word`, and a monospace `phrase of text`.

Monospace c``hara``cter``s`` within a word.
```

The style of bold text depends on the font and the CSS tyles for `<code>` HTML tag.

highlight example

```
Let's #highlight this phrase# and the i and s in th##is##.
```

The style of bold text depends on the font and the CSS tyles for `<mark>` HTML tag.

quotation raw text

```
.quotation mark
* 'normal single quote'
* "normal double quote"
* '`single quote and backtick.`'
* "`double quote and backtick.`"
```

quotation mark

- 'normal single quote'
- "normal double quote"
- ‘single quote and backtick.’
- “double quote and backtick.”

apostrophes raw

```
* The `'80s.
* The students`' books.
* I can't find Joey's house.
* I can't find Joey\'s house.
```

apostrophes

- The ’80s.
- The students’ books.
- I can’t find Joey’s house.
- I can’t find Joey's house.

special characters

Some special characters are replaced with the appropriate character or unicode entity. The three special characters, <, >, and &, are always replaced first.

See the full table here Also this

characters

```
&sect;
&#167;
```

```
&amp;
(C)
(R)
(TM)
...
<=
<-
=>
->
```

- §
- &
- ©
- ®
- ™
- …
- ⇐
- ←
- ⇒
- →

Note that some special characters won’t work for asciidoctor-pdf. It will work, however, if you use the numeric reference. discuss

For example, `&sect;` will not work, but `&#167;` will work fine with asciidoctor-pdf.

escape replacement

If you don’t want to replace into special character, escape with `\`.

ref

For example, `.\...` will show normal four periods `....` instead of 3+1 `⋯.`.

escape

```
`.\\...` `.\...` `....`
`\&sect;` `\&#167;`
```

`.\...` `....` `⋯.` `&sect;` `&#167;`

subscript and superscript

subscript superscript examples

```
The chemical formula for water is H~2~O.
```

```
What is the answer to E=mc^2^?
```

The chemical formula for water is H_2O. What is the answer to $E=mc^2$?

internal xref

Inline reference example here.

create internal xref

```
=== spectial characters

 [#in-page-xref-example]
 .apostrophes raw
 [,adoc]
 ----
 * The `'80s.
 * The students`' books.
 * I can't find Joey's house.
 * I can't find Joey\'s house.
 ----

Inline referenc example [#inline-ref]#here#.
```

create internal xref

```
.How to in-page cross reference:
* <<special characters>>
* <<special-characters>>
* <<special characters,go to the section>>

.Link to above block
<<in-page-xref-example>>
<<in-page-xref-example>>

.Inline reference
* <<inline-ref>>
* <<inline-ref,go to inline example>>
```

Below is the same from above block, but rendered. Note the deferences.

How to in-page cross reference:

- special characters
- [asciidoc-primer:::special-characters]
- this uses section’s title
- this uses implicit ID

Link to above block

- apostrophes raw
- go to the example block

Inline reference

- [asciidoc-primer:::inline-ref]
- go to inline example

URL

link to an external URL

Links that start with `https`, `ftp`, `mailto`, etc. will automatically turned into hyperlinks.

To excape a URL, so that it is not active, prepend it with a backslash `\`.

URL examples

```
Visit the https://chat.antora.org[Antora chat room].

This URL is displayed, \https://gitlab.com, but isn't clickable.
```

Note that a URL with characters like `_` or `^` might not display correctly. See this troubleshooting

troubleshoot url

```
= Page Title
:url-peak: https://www.google.com/maps/place/Antora+Peak/@38.3249976,-
106.2355882,14z/data=!3m1!4b1!4m5!3m4!1s0x871572433f469bd7:0xd2bdf15e615cd269!8m2!3d38
.3249994!4d-106.2180786!5m1!1e4

{url-peak}[This URL] was complicated, but a page attribute came to our rescue!

pass:macros[URL goes between these brackets]
Anyone want to climb this 13er with me?
(pass:macros[https://www.14ers.com/13ers/peak.php?peakkey=4740])
```

Lists

Ordered list example

```
. Step 1
. Step 2
.. Details
... Mini-details
.... Micro-details
..... We're really down in the weeds now.
.. More details
. Step 3
```

```
[lowergreek]
. alpha
. beta
. gamma

[start=4]
. Picking up where we left off.
. Add one more ingredient.
. Mix and serve!
```

1. Step 1
2. Step 2
    a. Details
        i. Mini-details
            A. Micro-details
                I. We’re really down in the weeds now.
    b. More details
3. Step 3

α. alpha

β. beta

γ. gamma

4. Picking up where we left off.
5. Add one more ingredient.
6. Mix and serve!

Unordered list example

```
* Item A
* Item B
** Item B1
*** Details
**** More details
***** Details about the details
** Item B2
* Item C
```

- Item A
- Item B
    - Item B1
        - Details
            - More details

            ▪ Details about the details

    ◦ Item B2

- Item C

Complex ordered and unordered lists

```
* A list item can contain two or more paragraphs or blocks of content.
+
[source,yaml]
----
  artifacts:
    expire_in: 1 week
----
+
====
An example block
====

** A literal paragraph doesn't require a list continuation.

 $ antora antora-playbook.yml

* You can also mix list item types in the same list.
** Unordered list item
... Ordered list item
.... Another ordered list item
+
term:: A description list term and content.
```

- A list item can contain two or more paragraphs or blocks of content.

  ```yaml
    artifacts:
      expire_in: 1 week
  ```

  An example block

    ◦ A literal paragraph doesn’t require a list continuation.

      ```
      $ antora antora-playbook.yml
      ```

- You can also mix list item types in the same list.

    ◦ Unordered list item

        i. Ordered list item

            A. Another ordered list item

term

A description list term and content.

Alternatively, when a list item contains multiple blocks, you can wrap them in a delimited open block (--). Then you only need a single list continuation line to attach the open block to the list item.

example open block

```
* A list item that includes several blocks wrapped in an open block.
+
--
[source,yaml]
----
  artifacts:
    expire_in: 1 week
----

====
An example block
====

Another paragraph
--
```

- A list item that includes several blocks wrapped in an open block.

  ```yaml
  artifacts:
    expire_in: 1 week
  ```

  An example block

  Another paragraph

Description list

```
Keyboard::
Used to enter text or control items on the screen.
Mouse:: Used to point to and select items on your computer screen.




term 1::
This description needs two paragraphs.
To attach them both to term 1, use a list continuation (+) on the line separating the
paragraphs.
+
This is the second paragraph for term 1.
```

```
term 2:: This description includes an admonition block.
Like additional paragraphs, blocks also need to be connected with a +.
+
NOTE: An admonition block that is part of term 2's description.

term 3::
* unordered list item
.. ordered list item
... another ordered list item
```

Keyboard
: Used to enter text or control items on the screen.

Mouse
: Used to point to and select items on your computer screen.

term 1
: This description needs two paragraphs. To attach them both to term 1, use a list continuation (+) on the line separating the paragraphs.

  This is the second paragraph for term 1.

term 2
: This description includes an admonition block. Like additional paragraphs, blocks also need to be connected with a +.

  > An admonition block that is part of term 2’s description.

term 3

- unordered list item
  a. ordered list item
     i. another ordered list item

and so on

More things covered by antora docs' asciidoc primer:

- Lists: Description Lists, Checklists
- Embed a Video
- UI Macros: Things like button, Keybinding and Menu syntax
- Source Blocks
- Admonitions
- Examples
- Sidebars
- Comments

For more asciidoc stuff Read further form the antora doc. Or visit the asciidoc.

AsciiDoc for STEM

This page is just a note for asciidoc stem documentation.

Activate

To activate STEM, you should put the `stem` attribute in the header.

To activate STEM support

```
= My Diabolical Mathematical Opus
Jamie Moriarty
// By default, assumes AsciiMath equations if not specified
:stem:
```

To use LaTeX notation by default

```
= My Diabolical Mathematical Opus
Jamie Moriarty
:stem: latexmath
```

These are just to set the default, so by setting explicitly onc can use both `asciimath` or `latexmath`.

using with Antora

By default, the STEM will **not** work with the default Antora setup and default Antora UI.

Mathjax

One way to use it is, using `mathjax`.

For Detailed instruction:

- the npm package
- For Antora

Here is a short instruction:

1. install Mathjax
   - global installation: `npm i -g @djencks/asciidoctor-mathjax`
   - local installation: add `@djencks/asciidoctor-mathjax` to your dev dependencies
2. add the extension to the playbook

   ```
   asciidoc:
     extensions:
   ```

```
  - '@djencks/asciidoctor-mathjax'
```

Examples

Inline stem

```
stem:[sqrt(4) = 2]

Water (stem:[H_2O]) is a critical component.

A matrix can be written as stem:[[[a,b\],[c,d\]\]((n),(k))].
```

`sqrt(4) = 2`

Water (`H_2O`) is a critical component.

A matrix can be written as `[[a,b],[c,d]]((n),(k))`.

mermaid

use mermaid

Here I show how to set up mermaid chart for antora.

mermaid with antora

It is from antora-mermaid extension.

setup

npm install

```
npm i -D @sntke/antora-mermaid-extension
```

Then update the `antora-playbook.yml`:

antora-playbook.yml

```
antora:
  extensions:
    - require: '@sntke/antora-mermaid-extension' ①
      mermaid_library_url:
https://cdn.jsdelivr.net/npm/mermaid@10/dist/mermaid.esm.min.mjs ②
      script_stem: header-scripts ③
      mermaid_initialize_options: ④
        start_on_load: true
```

① npm package name: @sntke/antora-mermaid-extension (required)

② URL of Mermaid.js library (optional)

③ Stem that exists in the handlebar templates of UI bundle where HTML script element for mermaid.js is placed. (optional)

④ The argument to mermaid.initialize(). (optional) Make sure to convert the Mermaid config keys to snake case, e.g., startOnLoad → start_on_load or themeVariables → theme_variables. Refer to the Antora docs for details.

If you want to specify the version of mermaid, you can do you using <2>. Note that the mermaid version 11 has architecture-beta, but not the mermaid version 10. For example:

antora-playbook.yml

```
antora:
  extensions:
  - require: '@sntke/antora-mermaid-extension'
    mermaid_library_url:
https://cdn.jsdelivr.net/npm/mermaid@11/dist/mermaid.esm.min.mjs
```

If you don't need any options, just simply put as following:

antora-playbook.yml

```
antora:
  extensions:
    - '@sntke/antora-mermaid-extension'
```

Now you can put mermaid diagram and it will show in the browser.

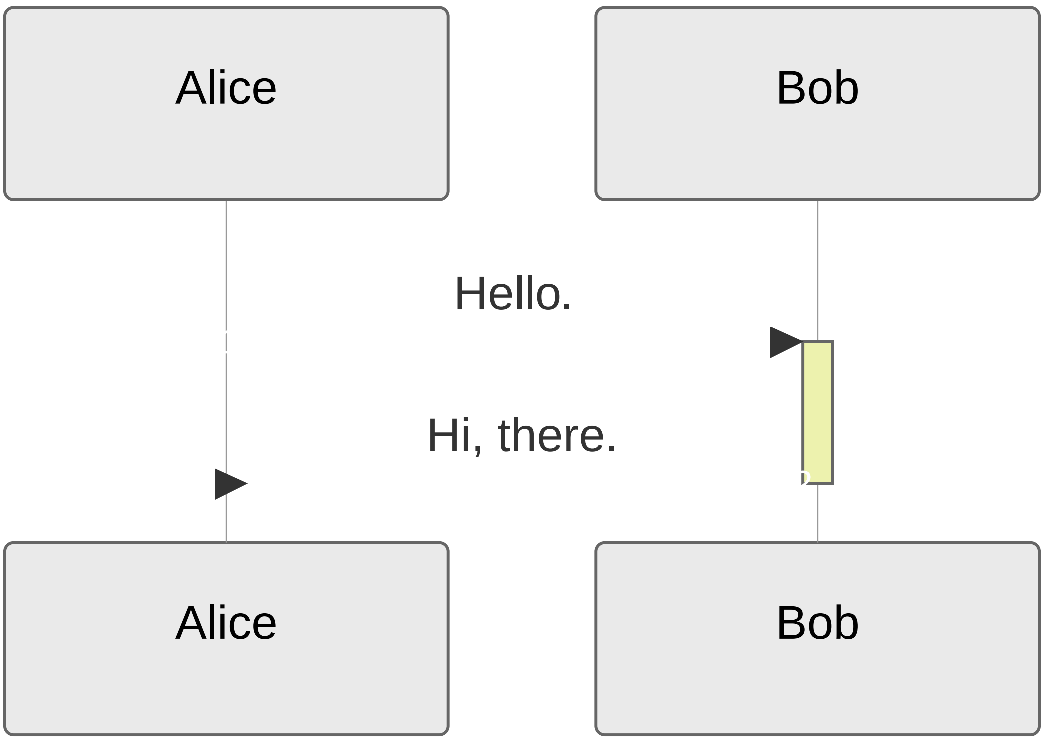
use mermaid

This extension visualized Listing Blocks (`....`) and Literal Blocks (`----`) of `mermaid` on HTML files.

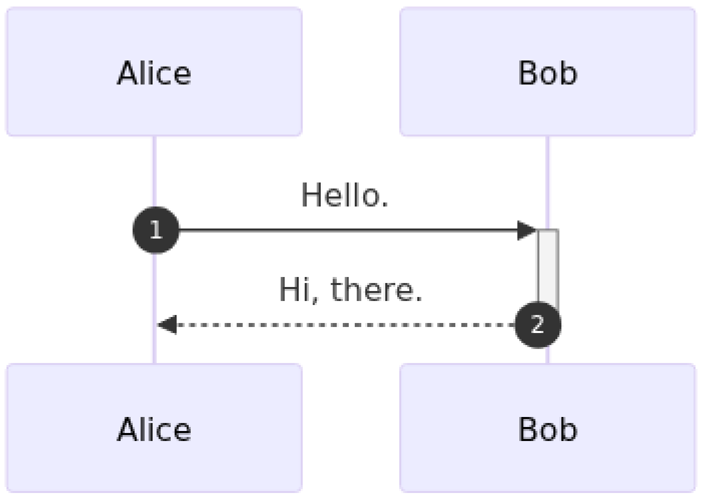
example mermaid

```
[mermaid]
....
sequenceDiagram
autonumber

participant a as Alice
participant b as Bob

a ->>+ b : Hello.
b -->>- a : Hi, there.
....
```

Below is with the `format=svg`:



Below is with the `format=png`:



mermaid to pdf

If you also want to create a pdf with antora, the mermaid diagram will not render out of the box. You need to set up couple of things.

- asciidoctor-diagram
- mermaid-cli: `mmdc` this will render diagram
- update the command for assembler
- `mmdc` uses puppeteer with chromium, so set up for it…
  - either download dependencies
  - or download chromium
- also chromium sandbox (for docker)
  - either no-sandbox (dangerous)
  - or set it up properly

setup

install asciidoctor-diagram

Install asciidoctor-diagram

```
gem install asciidoctor-diagram
```

You can as well use `asciidoctor-kroki`. The difference is that kroki uses kroki server, while `asciidoctor-diagram` renders locally. So if you want to use `asciidoctor-diagram` with mermaid, you need to install `mermaid-cli`.

install mermaid-cli

Here I install mermaid-cli globally.

Install mermaid-cli

```
npm install -g @mermaid-js/mermaid-cli
```

This will enable the command `mmdc`. You can try out with `mmdc -i input.mmd -o output.png` to see it works properly. Note that `mmdc` needs chromium to render (at least for now), so we will install chromium later.

update antora-assmebler.yml

Now you have to tell the antora-assembler, who is making the pdf file, to use the asciidoctor-diagram. So, update the `antora-assembler.yml` file.

Update `antora-assembler.yml`

```
build:
  command: asciidoctor-pdf -r asciidoctor-diagram
```

If you are using Bundler to manage gems (if Gemfile.lock), and you want to customize the command, make sure you prefix the command with bundle exec.

Update `antora-assembler.yml`

```
build:
  command: bundle exec asciidoctor-pdf -r asciidoctor-diagram
```

Depending on your setting, this may be enough.

chromium for mermaid-cli

But if you are using docker, the mermaid-cli `mmdc` might not work, due to lacking libs.

option 1. install libraries for chromium

Here is a possible fix (for debian), (it is from gpt, so might be not 100% correct)

Install dependencies for `mmdc` *and* `chromium`

```
sudo apt-get update
sudo apt-get install -y \
  libglib2.0-0 \
  libnss3 \
  libx11-6 \
  libatk1.0-0 \
  libatk-bridge2.0-0 \
  libcups2 \
  libdrm2 \
  libdbus-1-3 \
  libxcomposite1 \
  libxdamage1 \
  libxfixes3 \
  libxrandr2 \
  libgbm1 \
  libpango-1.0-0 \
  libcairo2 \
  libasound2 \
  libxkbcommon0 \
  libxkbcommon-x11-0
```

option 2. install chromium

Altenatively, just install chromium. This may be better for people with apple silicon. Cause we can specify this chromium for the puppeteer, which will smooth out some problems.

Install chromium

```
sudo apt-get update
sudo apt-get install -y chromium
```

Sandbox problem (docker)

Also, the mermaid-cli might still say that `No usable sandbox!`.

option 1. no-sandbox

The easy way to fix it is adding no-sandbox to the mmdc. how-to-no-sandbox.

Only use this option, if you make your own charts.

First, make `puppeteerConfig.json` file:

example puppeteerConfig.json

```
{
  "executablePath": "/usr/bin/chromium", ①
  "args": ["--no-sandbox", "--disable-dev-shm-usage"] ②
}
```

① you can specify the chromium to use.

② In containers, /dev/shm is tiny, disable-dev-shm-usage avoids crashes

Then specify this config in your `antora-assembler.yml`. Note that puppeteer-config is prefixed with `mermaid-`: ref

example antora-assembler.yml

```
build:
  command: asciidoctor-pdf -r asciidoctor-diagram -a mermaid-puppeteer-config=<path-
to>/puppeteerConfig.json ①
```

① puppeteer path is preferably absolute path, cause it is looking for the file from the `_exports`

option 2. docker run with seccomp profile

To allow the docker to use chromium, you need the right setting.

For more info, refer to this blog post.

Long story short, you have to

1. download the chrome.json
2. run the image with `--security-opt seccomp=chrome.json`

example commands

```
curl -L -o chrome.json
https://raw.githubusercontent.com/jfrazelle/dotfiles/master/etc/docker/seccomp/chrome.
json  ①

# example run
docker run -it \
  --init \
  --security-opt seccomp=<path to seccomp profile>/chrome.json \ ②
  dieters877565/chromedriver:latest

docker run -it \
  --pids-limit=200 \  ③
  --cpus=1 \ ④
  --cap-drop=ALL \ ⑤
  --security-opt seccomp=chrome.json \ ⑥
  --security-opt no-new-privileges \ ⑦
  --tmpfs /tmp:rw,noexec,nosuid,nodev,size=128m \
  --rm -p 8080:8080 -v "$PWD/dropbox":/mnt/dropbox \
  $IMAGENAME:$VERSION bash --login
```

① download the chrome.json

② give the downloaded chrome.json

③ no fork-bombs, stops it from spawning zombie processes

④ throttle CPU greed, keeps it from hogging all your cores

⑤ drop every Linux capability

⑥ specify the path you downloaded chrome.json

⑦ block privilege escalation

In this docker container, you can drop the `--no-sandbox` in the puppeteerConfig.json

mmdc might not work

The second docker run command above with more flags might fail mmdc to run.

example puppeteerConfig.json

```
{
  "executablePath": "/usr/bin/chromium",
  "args": ["--disable-dev-shm-usage"]
}
```

Download PDF

mermaid with custom icons

To use custom icons

with site

This assumes that you use antora-mermaid-extension.

So, adding icon pack is not coming out of the box.

Either you download the antora-mermaid-extension itself and update the code, or add another extension. Here we choose to simply add-on

add-on to the antora-mermaid-extension

example code written with gpt

```
'use strict'

/**
 * Antora extension: register Mermaid icon packs *after* the Mermaid init script.
 *
 * Config example (in antora-playbook.yml):
 *
 * extensions:
 *   - require: ./antora-mermaid.js            # your existing extension
 *   - require: ./antora-mermaid-iconpacks.js  # this file
 *     config:
 *       # Must match the URL used by your Mermaid init extension!
 *       mermaid_library_url:
https://cdn.jsdelivr.net/npm/mermaid@10/dist/mermaid.esm.min.mjs
 *       # Optional; defaults to 'header-scripts' (the {{> header-scripts}} slot).
 *       script_stem: header-scripts
 *       icon_packs:
 *       - name: logos
 *         url: https://unpkg.com/@iconify-json/logos@1/icons.json
 *       - name: devicon
 *         url: https://unpkg.com/@iconify-json/devicon@1/icons.json
 */

const DEFAULT_MERMAID_LIBRARY_URL =
'https://cdn.jsdelivr.net/npm/mermaid@10/dist/mermaid.esm.min.mjs'
const DEFAULT_SCRIPT_STEM = 'header-scripts'

// We deliberately choose a path that sorts AFTER "partials/mermaid-script.hbs"
// (commonly used by mermaid init extensions) so our script executes later.
const ICONPACKS_PARTIAL_PATH = 'partials/zzz-mermaid-iconpacks.hbs'
const MERMAID_INIT_PARTIAL_PATH = 'partials/mermaid-script.hbs' // for presence check
only

module.exports.register = (context, { config }) => {
  context.on('uiLoaded', appendIconPacksScript(config))
}

const appendIconPacksScript = (cfg) => ({ uiCatalog }) => {

  const mermaidLibraryUrl = cfg.mermaidLibraryUrl || DEFAULT_MERMAID_LIBRARY_URL
  const scriptStem = cfg.scriptStem || DEFAULT_SCRIPT_STEM
  const iconPacks = Array.isArray(cfg.iconPacks) ? cfg.iconPacks : []

  // Nothing to do if no packs configured
  if (!iconPacks.length) return

  // Avoid adding twice
  if (uiCatalog.findByType('partial').some(({ path }) => path ===
ICONPACKS_PARTIAL_PATH)) return

  // Best-effort: if a known mermaid init partial is present, we�ll append ours with a
  // lexicographically later path ("zzz-...") so it is rendered after it.
  // Even if the init partial has a different name, adding our partial later and with
  // a "zzz-" path usually preserves the intended order in {{> header-scripts}}.
  const hasMermaidInitPartial = uiCatalog.findByType('partial').some(({ path }) =>
path === MERMAID_INIT_PARTIAL_PATH)
  // (We don�t *require* it; this still works if mermaid is imported again from the
same URL.)

  // Validate and sanitize packs (simple guardrails)
  const packs = iconPacks
    .map((p) => ({
      name: String(p && p.name || '').trim(),
      url: String(p && p.url || '').trim(),
    }))
    .filter((p) => p.name && p.url)

  if (!packs.length) return

  // Build a tiny ESM that imports the SAME mermaid URL and registers all packs.
  // Using `import ... from <same URL>` guarantees the same module instance if the URL
matches
  // the init script; otherwise it still works on its own.
  const packsJson = JSON.stringify(packs)
  const hbsContent = [
    '<script type="module">',
    `  import mermaid from ${JSON.stringify(mermaidLibraryUrl)};`,
    `  const packs = ${packsJson}.map(({ name, url }) => ({`,
    '    name,',
    '    loader: () => fetch(url).then((res) => {',
    '      if (!res.ok) throw new Error(`Failed to load icon pack: ${name}
(${res.status})`);',
    '      return res.json();',
    '    }),',
    '  }));',
    '  try {',
    '    // Mermaid >= 10.9: registerIconPacks exists. Older versions will just skip.
',
    '    if (typeof mermaid.registerIconPacks === "function") {',
    '      mermaid.registerIconPacks(packs);',
    '    } else {',
    '      // Fallback: log a friendly hint.',
    '      console.warn("[antora-mermaid-iconpacks] mermaid.registerIconPacks() not
found — is your Mermaid version new enough?");',
    '    }',
    '  } catch (e) {',
    '    console.error("[antora-mermaid-iconpacks] Failed to register icon packs:",
e);',
    '  }',
    '</script>',
    '',
  ].join('\n')

  uiCatalog.addFile({
    contents: Buffer.from(hbsContent),
    path: ICONPACKS_PARTIAL_PATH, // "zzz-" to bias ordering after common init
partials
    stem: scriptStem,              // {{> header-scripts}} in your UI will include
this
    type: 'partial',
  })
}

module.exports.appendIconPacksScript = appendIconPacksScript
```

Then also update the `antora-playbook.yml`

example antora-playbook.yml

```
antora:
  extensions:
  - require: '@sntke/antora-mermaid-extension'
    mermaid_library_url:
https://cdn.jsdelivr.net/npm/mermaid@11/dist/mermaid.esm.min.mjs ①
  - require: ./_config-mermaid/antora-mermaid-iconpack.js
    mermaid_library_url:
```

```
https://cdn.jsdelivr.net/npm/mermaid@11/dist/mermaid.esm.min.mjs ②
    icon_packs:
    - name: azure
      url:
https://raw.githubusercontent.com/NakayamaKento/AzureIcons/refs/heads/main/icons.json
```

The mermaid_library_url should match the antora-mermaid-extension. (<1> and <2>)

for pdf

This assumes that you use antora assembler with asciidoctor-diagram/mermaid.

As you can see in the list attributes in the above link, the `iconPacks` or `iconPacksNamesAndUrls` for `mmdc` is not possible to pass through using asciidoctor-diagram. (as far as I know of).

To work around, I write a wrapper script:

mmdc-wrapper.sh

```
#!/bin/bash
# Forward all arguments Asciidoctor Diagram gives us
# but inject your preferred flags

echo "$(date) mmdc wrapper called with: $*" >> /tmp/mmdc-wrapper.log

exec mmdc \
  --iconPacksNamesAndUrls
azure#https://raw.githubusercontent.com/NakayamaKento/AzureIcons/refs/heads/main/icons
.json \ ①
  -p <path-to>/puppeteerConfig.json \ ②
  "$@"
```

example mmdc-wrapper.sh

```
#!/bin/bash

echo "$(date) mmdc wrapper called with: $*" >> /tmp/mmdc-wrapper.log

# If the puppeteerConfig.json is in the same dir as mmdc-wrapper.sh (just to be safe)
SCRIPT_DIR="$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )"

exec mmdc \
  --iconPacksNamesAndUrls
azure#https://raw.githubusercontent.com/NakayamaKento/AzureIcons/refs/heads/main/icons
.json \
  -p "$SCRIPT_DIR/puppeteerConfig.json" \ ③
  "$@"
```

Do NOT forget to `chmod +x mmdc-wrapper.sh`

① add the iconPack you want to use here

② may as well put the puppeteerConfig here

③ well `./puppeteerConfig.json` also worked somehow

mmdc --help

```
  --iconPacks <icons...>                        Icon packs to use, e.g. @iconify-
json/logos. These should be Iconify NPM packages that expose a icons.json file, see
https://iconify.design/docs/icons/json.html. These will be downloaded from
https://unkpg.com when needed. (default: [])
  --iconPacksNamesAndUrls <prefix#iconsurl...>    Icon packs to use, e.g.
azure#https://raw.githubusercontent.com/NakayamaKento/AzureIcons/refs/heads/main/icons
.json where the name (prefix) of the icon pack is defined before the "#" and the url
of the json definition after the "#". These should be Iconify json file formatted as
IconifyJson, see https://iconify.design/docs/icons/json.html. These will be downloaded
when needed. (default: [])
```

You can as well add other options for `mmdc` in this wrapper if you want.

Then add the mmdc into `antora-assembler.yml`

example antora-assembler.yml

```
component_version_filter:
  names: ['*@technical-docs', '*@example-docs']
assembly:
  root_level: 0
  insert_start_page: true
  section_merge_strategy: enclose
  doctype: book
asciidoc:
  attributes:
    mermaid-background: 440000
    mmdc: <path-to>/mmdc-wrapper.sh ①
build:
  clean: true
  mkdirs: true
  dir: build-pdf
  publish: true
  command: asciidoctor-pdf -r asciidoctor-diagram
```

① the mmdc path can be relative

To be honest it is not really a pretty solution, but a dirty hack, but it kinda works…

Together

Now we have to specify the iconpack both for the site in `antora-playbook.yml` as well as `mmdc-wrapper.sh`. It is error-prone. So we will make the `mmdc-wrapper.sh` to look for the antora-playbook.yml for the iconpack information.

This is a bit messy code made with gpt…

bit messy update for mmed-wrapper.sh

```bash
#!/bin/bash
# Forward all arguments Asciidoctor Diagram gives us
# but inject your preferred flags from antora-playbook.yml

echo "$(date) mmdc wrapper called with: $*" >> /tmp/mmdc-wrapper.log

set -euo pipefail

# --- locate files relative to this wrapper ---
SCRIPT_DIR="$( cd -- "$( dirname -- "${BASH_SOURCE[0]}" )" &> /dev/null && pwd )"

PUPPETEER_CONFIG="-p${SCRIPT_DIR}/puppeteerConfig.json"
PLAYBOOK="${SCRIPT_DIR}/../antora-playbook.yml"

echo "$PLAYBOOK" >> /tmp/mmdc-wrapper.log

# --- config: which extension block to read, and which flag to emit ---
# Which substring must appear in the 'require' field of the extension?
ICON_EXT_REQUIRE="${ICON_EXT_REQUIRE:-antora-mermaid-iconpack}"
# Most builds use --iconPacks; some forks used --icons. Override via env if needed.
ICON_FLAG_NAME="${ICON_FLAG_NAME:---iconPacksNamesAndUrls}"

# Optional debug (stderr only, no files). Enable with: DEBUG=1
dbg() { if [[ "${DEBUG:-}" == "1" ]]; then printf '%s\n' "$*" >&2; fi; }

dbg "wrapper: using playbook ${PLAYBOOK}"
dbg "wrapper: target extension require=${ICON_EXT_REQUIRE}"

# ----- need yq (mikefarah) for YAML parsing -----
if ! command -v yq >/dev/null 2>&1; then
  echo "$(date) 'yq' not found" >> /tmp/mmdc-wrapper.log
  dbg "wrapper: 'yq' not found; running mmdc without icon packs"
  echo "exec mmdc ${PUPPETEER_CONFIG} $@" >> /tmp/mmdc-wrapper.log
  exec mmdc "${PUPPETEER_CONFIG}" "$@"
fi

# ----- extract icon packs as name=url lines (substring match on 'require') -----
# Handles arrays like:
#   - '@antora/pdf-extension'
#   - { require: './_config-mermaid/antora-mermaid-iconpack.js', icon_packs: [...] }
mapfile -t PACK_ITEMS < <(
  yq -r '
  ((.antora.extensions // .extensions // [])[])?
  | select(has("require")?)
  | select((.require | tostring) | contains("antora-mermaid-iconpack"))
  | (.icon_packs // [])[]?
  | "\(.name)=\(.url)"
  ' "$PLAYBOOK" 2>/dev/null || true
)



if [[ ${#PACK_ITEMS[@]} -eq 0 ]]; then
  echo "$(date) no icon_packs found" >> /tmp/mmdc-wrapper.log
  dbg "wrapper: no icon_packs found; running mmdc without icon packs"
  echo "exec mmdc ${PUPPETEER_CONFIG} $@" >> /tmp/mmdc-wrapper.log
  exec mmdc "${PUPPETEER_CONFIG}" "$@"
fi

# ----- build args: repeat --iconPacksNamesAndUrls for each pack -----
echo "PCAK_ITEMS: ${PACK_ITEMS}" >> /tmp/mmdc-wrapper.log
echo "PCAK_ITEMS: ${#PACK_ITEMS[@]}" >> /tmp/mmdc-wrapper.log
args=()
for pack in "${PACK_ITEMS[@]}"; do
  prefix="${pack%%=*}"; prefix="${prefix##+([[:space:]])}"; prefix="${prefix
%%+([[:space:]])}"
  url="${pack#*=}";     url="${url##+([[:space:]])}";      url="${url%%+([[:space:]]
)}"
  echo "prefix: ${prefix}" >> /tmp/mmdc-wrapper.log
  echo "url: ${url}" >> /tmp/mmdc-wrapper.log
  [[ -n "$prefix" && -n "$url" ]] || continue
  # If your flag variable is --iconPacksNamesAndUrls
  args+=( "$ICON_FLAG_NAME" "${prefix}#${url}" )
done


dbg "wrapper: passing icon packs: ${args[*]}"
echo "$(date) passing icon packs ${args[*]}" >> /tmp/mmdc-wrapper.log
echo "exec mmdc ${PUPPETEER_CONFIG} ${args[@]} $@" >> /tmp/mmdc-wrapper.log

# ----- run mmdc with our flags + original args -----
exec mmdc "${PUPPETEER_CONFIG}" "${args[@]}" "$@"
```

1. need `yq` installed
2. the antora-playbook.yml location is hard coded to be ../ from the mmdc-wrapper.sh
3. it is searching for the `antora-mermaid-iconpack` from the previous section.

openapi

Generate OpenAPI Documentation from Quarkus

This guide explains how to generate OpenAPI documentation from a Quarkus project and convert it to AsciiDoc format.

Generate OpenAPI JSON from Quarkus

Quarkus uses SmallRye OpenAPI to automatically generate OpenAPI specifications from your REST endpoints. To export the OpenAPI schema to a file during the build process, use the following Maven command:

```
./mvnw package -Dquarkus.smallrye-openapi.store-schema-directory=target/openapi
```

This command will:

- Build your Quarkus application
- Generate the OpenAPI specification
- Store it in the `target/openapi` directory

The generated file will typically be named `openapi.json` or `openapi.yaml` depending on your configuration.

Configuration Options

You can also configure this behavior in your `application.properties`:

```
quarkus.smallrye-openapi.store-schema-directory=target/openapi
```

Convert OpenAPI to AsciiDoc

Once you have the OpenAPI specification file, you can convert it to AsciiDoc format using the OpenAPI Generator CLI tool.

Prerequisites

Install the OpenAPI Generator CLI:

```
npm install @openapitools/openapi-generator-cli -g
```

Or use it via npx without installation:

```
npx @openapitools/openapi-generator-cli
```

Generate AsciiDoc Documentation

Run the following command to generate AsciiDoc documentation from your OpenAPI specification:

```
openapi-generator-cli generate -i openapi.yaml -g asciidoc -o output
```

Where:

- `-i openapi.yaml` - Input OpenAPI specification file (use `openapi.json` if your file is in JSON format)
- `-g asciidoc` - Generator type (AsciiDoc)
- `-o output` - Output directory for generated documentation

Complete Workflow Example

Here's a complete example workflow:

```
# 1. Generate OpenAPI specification from Quarkus project
./mvnw package -Dquarkus.smallrye-openapi.store-schema-directory=target/openapi

# 2. Navigate to the output directory
cd target/openapi

# 3. Generate AsciiDoc documentation
openapi-generator-cli generate -i openapi.yaml -g asciidoc -o ../../docs/api
```

The generated AsciiDoc files will include:

- API overview and description
- Endpoint documentation with request/response examples
- Model/schema definitions
- Authentication and security requirements

Integration with Antora

To integrate the generated OpenAPI documentation into your Antora site:

1. Copy the generated `.adoc` files to your Antora module's pages directory
2. Add references to the new pages in your `nav.adoc` file
3. Rebuild your Antora site

```
# Example: Copy generated docs to Antora
```

```
cp docs/api/*.adoc antora-docs/technical-docs/modules/ROOT/pages/api/
```

Additional Resources

- Quarkus OpenAPI Guide
- OpenAPI Generator
- SmallRye OpenAPI

Note

This is written by Claude

example script to generate openapi json

I keep the up-to-date code in the main branch, but keep the documentation on a seperate branch. In that case to use the java's openapi generation, I need the up-to-date code. One can achieve this using git worktree.

Below is an example code written by chatGPT to do that.

example script to generate openapi

```
#!/usr/bin/env bash
set -euo pipefail

# ---- config you can tweak -------------------------------------------------
MAIN_BRANCH="${MAIN_BRANCH:-main}"

DOCS_BRANCH="${DOCS_BRANCH:-documentation}"

MODULE="${MODULE:-mymodule}"

# Path (from repo root) where ./mvnw lives (the Quarkus project root)
# myproj/mymodule from the root of the repo
PROJECT_DIR_REL="${PROJECT_DIR_REL:-myproj/$MODULE}"

echo "${PROJECT_DIR_REL}"

# Where Quarkus writes schemas (relative to each module/project dir)
OPENAPI_DIRNAME="${OPENAPI_DIRNAME:-target/openapi}"

# Destination on the docs branch (relative to repo root)
OUT_DIR="${OUT_DIR:-documentation/generated_openapi}"

# Extra Maven flags
MVN_ARGS="${MVN_ARGS:--q -DskipTests}"
# ---------------------------------------------------------------------------

# Ensure we're at repo root
```

```
if [[ ! -d .git ]]; then
  echo "Run this from the repository root (where .git exists)." >&2
  exit 1
fi

# Ensure mvnw exists in the provided project dir (in the MAIN worktree we□ll build)
if [[ ! -e "$PROJECT_DIR_REL" ]]; then
  echo "Warning: '$PROJECT_DIR_REL' doesn't exist in the current checkout; that's ok
(we'll use a worktree of $MAIN_BRANCH)." >&2
fi

CURRENT_BRANCH="$(git rev-parse --abbrev-ref HEAD)"
if [[ "$CURRENT_BRANCH" != "$DOCS_BRANCH" ]]; then
  echo "Warning: you are on '$CURRENT_BRANCH', not '$DOCS_BRANCH'." >&2
fi

# # Avoid losing local changes
# if ! git diff --quiet || ! git diff --cached --quiet; then
#   echo "Please commit or stash your changes before running." >&2
#   exit 1
# fi

# Prepare a temporary worktree for MAIN
WT="$(mktemp -d -t openapi-main.XXXXXX)"
cleanup() {
  git worktree remove --force "$WT" >/dev/null 2>&1 || true
  rm -rf "$WT" || true
}
trap cleanup EXIT

echo "-> Adding worktree for '$MAIN_BRANCH' at $WT"
# git fetch -q origin "$MAIN_BRANCH" || true
git worktree add -f "$WT" "$MAIN_BRANCH" >/dev/null

# Verify mvnw in the worktree
if [[ ! -x "$WT/$PROJECT_DIR_REL/mvnw" && ! -x "$WT/$PROJECT_DIR_REL/./mvnw" ]]; then
  if [[ -x "$WT/$PROJECT_DIR_REL/../mvnw" ]]; then
    echo "Note: using mvnw at $(realpath "$WT/$PROJECT_DIR_REL/../mvnw")"
  elif [[ -x "$WT/mvnw" ]]; then
    echo "Note: using mvnw at repo root in worktree"
  fi
fi

# Build to generate OpenAPI
echo "-> Building on $MAIN_BRANCH in '$PROJECT_DIR_REL' to generate OpenAPI schemas⋯"
(
  cd "$WT/$PROJECT_DIR_REL"
  ./mvnw package $MVN_ARGS -Dquarkus.smallrye-openapi.store-schema-directory=
"$OPENAPI_DIRNAME"
)
```

```
# Collect schemas (supports single or multi-module under PROJECT_DIR_REL)
echo "-> Collecting schemas…"
TMP_COLLECT="$(mktemp -d -t openapi-collect.XXXXXX)"
# Look for any */target/openapi under the project path
while IFS= read -r -d '' dir; do
  # Module dir is the parent of target/, or the project dir itself for single-module
  module_dir="$(dirname "$(dirname "$dir")")"
  # Base name of the Quarkus project root (e.g., "templates" from "backend/templates")
  base_name="$(basename "$PROJECT_DIR_REL")"

  dest="$TMP_COLLECT"
  mkdir -p "$dest"

  cp "$dir"/openapi.json "$dest"/$base_name.json
  cp "$dir"/openapi.yaml "$dest"/$base_name.yaml
done < <(find "$WT/$PROJECT_DIR_REL" -type d -path "*/$OPENAPI_DIRNAME" -print0)

# If nothing was found, fail clearly
if [[ ! -d "$TMP_COLLECT" || -z "$(find "$TMP_COLLECT" -type f -maxdepth 2
2>/dev/null)" ]]; then
  echo "□ No OpenAPI output found under '$PROJECT_DIR_REL/**/$OPENAPI_DIRNAME'. Did
the Quarkus build produce schemas?" >&2
  exit 2
fi

# Copy into docs branch destination
echo "-> Copying schemas into '$OUT_DIR' on branch '$CURRENT_BRANCH'…"
mkdir -p "$OUT_DIR"
cp -R "$TMP_COLLECT"/. "$OUT_DIR"/


# Show what changed and commit (optional)
echo "-> Preview of changes:"
git status --porcelain "$OUT_DIR" || true
echo

# MAIN_SHA="$(git -C "$WT" rev-parse --short HEAD)"
# if [[ "${AUTO_COMMIT:-1}" == "1" ]]; then
#   git add "$OUT_DIR"
#   if ! git diff --cached --quiet; then
#     git commit -m "docs: update OpenAPI schemas from ${MAIN_BRANCH}@${MAIN_SHA}"
#     echo "✓ Committed schema updates to '$OUT_DIR'."
#   else
#     echo "✓ No changes to commit."
#   fi
# else
#   echo "✓ Files staged in '$OUT_DIR' (AUTO_COMMIT=0). Review and commit as
desired."
# fi
```

openapi generator

Use openapi json or yaml file to generate codebase or docs. Here we show how to generate asciidoc based on the openapi.json.

basic command

```
openapi-generator-cli generate -i "openapi.yaml" -g asciidoc -o output.adoc ①

openapi-generator-cli generate --openapitools "openapitools.json" ②
```

① specify input, output and generator

② use `openapitools.json`, example below

openapitools file

It specifies generators and so on.

example `openapitools.json`

```
include::example$openapi-generator-adoc--openapitools.json[]
```

example `openapitools.json`

```
{
  "$schema": "./node_modules/@openapitools/openapi-generator-cli/config.schema.json",
  "spaces": 2,
  "generator-cli": {
    "version": "7.14.0",
    "generators": {
      "backend-apidoc": {
        "generatorName": "asciidoc",
        "glob": "./generated_openapi/*.json", ①
        "output": "../mydocmodule/modules/backend/partials/#{name}", ②
        "additionalProperties": {
          "headerAttributes": false,
          "useTableTitles": true,
          "useMethodAndPath": true,
          "useIntroduction": true,
          "skipExamples": true
        },
        "templateDir": "apidocs_templates", ③
        "config": "config.yaml" ④
      }
    }
  }
}
```

① json File of matching glob will be used to generate the asciidoc

② The `#{name}` is the name of the file.

③ to use custom templates specify the `templateDir`

④ specify the config file to be used. Can be `json` or `yaml`

Config file

Example config file:

example `config.yaml` to be specified under "config"

```
files: ①
  index.mustache:
    templateType: SupportingFiles
    destinationFilename: index.adoc
  api-partial.mustache:
    templateType: SupportingFiles
    destinationFilename: api-partial.adoc
```

① The files will be searched from the `templateDir` or use the default file, i guess.

custom templates

for more info.

The `templateType` can be:

- API
- APIDocs
- APITests
- Model
- ModelDocs
- ModelTests
- SupportingFiles

> Excluding SupportingFiles, each of the above options may result in multiple files. API related types create a file per API. Model related types create a file for each model.

The model or API name will be will be prefixed to the destinationFilename if the templateType is other than `SupportingFiles`

> Note that user-defined templates will merge with built-in template definitions. If a supporting file with the sample template file path exists, it will be replaced with the user-defined template, otherwise the user-defined template will be added to the list of template files to compile. If the

> generator’s built-in template is model_docs.mustache and you define model-docs.mustache, this will result in duplicated model docs (if destinationFilename differs) or undefined behavior as whichever template compiles last will overwrite the previous model docs (if destinationFilename matches the extension or suffix in the generator’s code).

Make pdf

Formalhauthorized — version dev, 2025-10-13

To build pdf using antora and asciidoctor-pdf

References:

1. docs.antora.org/assembler/latest/
2. docs.asciidoctor.org/pdf-converter/latest/

install

You need to install asciidoctor, asciidoctor-pdf and antora's pdf extension. Note that asciidoctor will install ruby so, you can use gem.

Here I assume debian.

Setup

```
sudo apt update
sudo apt install asciidoctor
gem install asciidoctor-pdf
// For syntax highlight
gem install rouge
// install antora pdf-extension either locally or globally depending your settings
npm i -g @antora/pdf-extension
```

asciidoctor-pdf

To use asccidoctor-pdf, you can run

Setup

```
asciidoctor-pdf basic-example.adoc
```

antora

To use antora to build pdf, you need to register the pdf-extension in your antora playbook. It is also good to add the site url for the links to work

antora-playbook.yml

```yaml
site:
  url: https://docs.example.org
antora:
  extensions:
```

```
  - '@antora/pdf-extension'
# ...
```

You can have the configurations for pdf generation in `antora-assembler.yml` or add your own config file under `config_file` key.