Annex B Documentation Conventions

B.1 Exceptions Allowed

Unless otherwise stated in the project-specific CONTRIBUTIONS.md file, each project based on this specification **shall** develop documentation as defined by this annex.

B.2 Development Environment

B.2.1 Overview

In addition to the development tools needed to manage and submit any contribution within the Git environment (e.g., Git, GitHub), developing project documentation requires the following tools:

- **A text editor,** which is used to create and edit markdwon and yaml files,
- **Python,** which is required to run MkDocs,
- **MkDocs,** which is an open-source tool for translating a set of markdown files into a static website, and
- **Materials for MkDocs,** which is an open-source tool that extends the markdown language to support additional features that are useful for developing the look and feel of the project's documentation.

This combination of tools has been selected because it:

- is designed to be easy to install and use,
- requires minimal setup,
- works well with Git and GitHub,
- supports search functionality,
- can produce a static website,
- when coupled with add-ons, can produce PDFs
- has an active development community

It is recommended to establish this development environment prior to making any edits. Generating the documentation website locally from a known baseline allows the contributor to verify that the development environment is working correctly prior to introducing edits to the files. Contributors are required to generate the documentation locally to verify that their proposed changes do not introduce any errors to the project. The MkDocs

development environemnt allows users to see their changes in real time so that any errors can be addressed quickly.

B.2.2 Text Editor

Any text editor can be used to produce markdown and yaml files. These files are to have the extensions of ".md" and ".yml", respectively.

Contributors are encouraged to use Visual Studio Code, which is an open-source editor, with the following extensions enabled as it provides a reasonably close rendering of the final display format:

- Markdown Preview Enhanced by Yiyi Wang, this extension provides a markdown previewer with support for diagrams, math (LaTeX), mermaid, charts, and more;
- markdownlint by David Anson, this extension assists in ensuring markdown files follow consistent formatting rules; and
- YAML by Red Hat, this extension provides syntax highlighting, validation, and autocomplete for YAML files.

While any text editor can be used, this suite of tools offers a free solution that is designed to render the markdown in real-time while assisting the user in producing high quality code. However, users should be aware that the toolset still does not attempt to render some of the more advanced features of Materials for MkDocs. The final look and feel can be obtained using the MkDocs server.

B.2.3 Python

MkDocs requires Python 3.8 or higher. You can check to see if Python is already installed and its version with the following command:

```
python --version
```

The most recent version of Python can be installed from official Python website.

Once installed, you should verify by running both the `python --version` and `pip --version` commands. PIP should be installed as a part of the Python package.

B.2.4 MkDocs

Running the MkDocs server locally allows the contributor to see proposed changes in real-time and test them thoroughly prior to submitting pull requests. To install MkDocs, run

```
pip install mkdocs
```

Once installed, verify its installation with:

`mkdocs --version`

Once you have verified the installation, start the MkDocs server by changing to the directory containing your cloned copy of the project repository and running

`mkdocs serve`

Once the server is running, you can direct a web browser to localhost port 8000 to see the development version of the website. This site will be updated in realtime as you update files in the repository. If you want to create a static site, However, to render all elements within the project correctly, you will need to install Materials for MkDocs.

B.2.5 Materials for MkDocs

To install Materials for MkDocs and the commonly used extensions for ITS projects, run the following command:

```
pip install mkdocs-material pymdown-extensions
```

B.3 Working with the Content

The content of ITS open-source documentation is generally written in Markdown, a lightweight and easy-to-use markup language that allows you to format text in a readable and visually appealing way.

Please read the "Frequently Used Markdown" section for details about how to use it in this project.

B.3.1 Default Document Structure

ITS open-source projects can cover a range of projects that have wildly different documentation needs. Each project is allowed to define its own structure, but unless otherwise specified **shall** use the structure defined in this document, which is intended for projects that result in a product that can be conceptualized as a single traditional document (e.g., a traditional standard).

Each major portion of the document **shall** be defined in a separate markdown file. Major portions are defined as:

- the title page, which shall be `index.md`;
- each top-level section of the front matter (e.g., Foreword, Introduction);
- each section in the body of the docuemnt; and
- each annex.

The document structure **shall** be reflected in the project's `mkdocs.yml` file under the `nav` section with all front matter located under a `Front Matter` heading.

```
    - Front Matter:
      - Title Page: index.md
      - Notices: notices.md
      - Acknowledgements: acknowledgements.md
      - Foreword: foreword.md
      - Introduction: introduction.md
    - 1 General: general.md
    - 2 Overview: overview.md
    - 3 Commenter Responsibilities: commenter-responsibilities.md
    - 4 Contributor Responsibilities: contributor-responsibilities.md
    - 5 Maintainer Responsibilities: maintainer-responsibilities.md
    - 6 WG Responsibilities: wg-responsibilities.md
    - A Code of Conduct: code-of-conduct.md
    - B Documentation Conventions: documentation-conventions.md
    - C Coding Conventions: code-quality.md
```

Note

When using the default configuration for ITS projects, this results in a left-hand left-hand navigation bar that shows the major portions of the document while the right-hand navigation shows the content of the currently opened section.

Note

Be sure to follow naming conventions. Notice that file names are not capitalized, and there are hyphens in place of spaces between words.

B.3.2 Structure of the Title Page File

The `index.md` file **shall** represent the title page of the document and **shall**:

- Start with a line containing a hashtag and nothing else
- Include code that suppresses unwanted markdownlint warnings
- Identify the status of the document
- Define the Document Identifier (e.g., NTCIP X8008)
- Define the Document Title (e.g., ITS Open-Source Process)
- Any other information required by the Standards Development Organization (SDO)

Example of a Title Page File

```
#
<!-- markdownlint-disable MD033 -->
<div style="text-align: center; font-style: italic; font-weight: bold;">
  A proposal to the NTCIP Joint Committee</div>
<div style="text-align: center; font-size: 1.5em; font-weight: bold;">
  NTCIP X8008
</div>
---
<div style="text-align: center; font-size: 1.5em; font-weight: bold;">
  National Transportation Communications ITS Protocol
</div>
<div style="text-align: center; font-size: 2em; font-weight: bold;">
  ITS Open-Source Process
</div>
<!-- markdownlint-enable MD033 -->
```

B.3.3 Structure of All Other Front Matter Files

Each file representing a major portion of the front matter, other than the title page, **shall** include a single level 1 heading that has the same title as defined in the `nav` section of the `mkdocs.yml` file and is the first line of the document

Example of Start of a Front Matter File

```
# Foreword
```

B.3.4 Structure of a Section File

Each file representing a section of the main body of the document **shall**:

- Start with code that sets the section counter for the body to the section number while suppressing unwanted markdownlint warnings
- Include a single level 1 heading that has the same title as defined in the `nav` section of the `mkdocs.yml` file and occurs immediately after the code defining the section number
- End each heading with `{.body}`

Example of Start of a Section File

```
<!-- markdownlint-disable MD033 -->
<!-- markdownlint-disable MD041 -->
<style>
  body { counter-set: section 3; }
</style>
<!-- markdownlint-enable MD033 -->
# Documentation Conventions {.body}
```

Note

Rule MD033 of markdownlint does issues a warning about the use of HTML within markdown, but it is is necessary in this case to allow automated numbering to work properly.

Rule MD041 of markdownlint indicates that the first line in a file should be a top-level heading, but our convention requires defining the section number first.

B.3.5 Structure of an Annex File

Each file representing an annex of the document **shall**:

- Start with code that sets the section counter for the annex to the numberical order of the annex (the script will transofmr this into an alphabetic letter)
- Include a single level 1 heading that has the same title as defined in the `nav` section of the `mkdocs.yml` file and occurs immediately after the code defining the section number
- End each heading with `{.annex}`

Example of Start of an Annex File

```
<!-- markdownlint-disable MD033 -->
<!-- markdownlint-disable MD041 -->
<style>
  body { counter-set: section 1; }
</style>
<!-- markdownlint-enable MD033 -->
# Example Annex {.annex}
```

B.3.6 Adding Definitions to the Glossary

If you add definitions to the project's glossary, ensure the definitions are added **alphabetically**.

B.3.7 Frequently Used Markdown

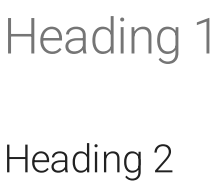
B.3.7.1 Headings

The hash (`#`) symbol at the start of a line denotes a heading (e.g., section, clause, subclause). There are six levels of headings, and the number of hash symbols indicates the heading level. The title of the heading should appear after the hash symbols and a space.

Example

```
# Heading 1

## Heading 2
```



B.4.7.2 Text Formatting

- Make text bold by enclosing it with double asterisks (`**`).
- Make text italic by enclosing it with single underscores (`_`).
- Create inline code by wrapping text with backticks (`` ` ``).

Example

```
**This is a bold text.**

_This is an italic text._

This is an`inline code`.
```

This is a bold text.

This is an italic text.

This is an `inline code`.

B.4.7.3 Lists

- Create ordered lists using numbers followed by a period (`1.` , `2.` , etc.).
- Create unordered lists using hyphens (`-`).
- The line before a list must be blank and a list cannot be immediately precedded by a different list
- The style for the list is defined by the first list item

Example

```
1.  Item 1
2.  Item 2

New List

- Unordered Item 1
- Unordered Item 2
```

1. Item 1
2. Item 2

New List

- Unordered Item 1
- Unordered Item 2

Note

The numbering of numbered lists is automatic within markdown (i.e., when rendered, the list items are numbered sequentially from 1 regardless of what numbers are contained within the markdown file); however, it is good coding practice to maintain the correct numbering within the markdown file to prevent any confusion among contributors.

B.4.7.4 Links

Create links using square brackets (`[]`) for the link text and parentheses (`()`) for the URL.

Example

```
[NTCIP](https://ntcip.org)
```

NTCIP

B.4.7.5 Images

Embed images using an exclamation mark (`!`), followed by square brackets (`[]`) for the alt text, and parentheses (`()`) for the image URL. AN optional attribute field can be added to the end to specify the size.

Example

```
![NTCIP](_assets/images/NTCIP.jpg){ width=200px }
```

B.4.7.6 Blockquotes

Create blockquotes using the greater-than symbol (`>`) or through Materials for MkDocs' admonition quote (`!!! quote`).

Example

```
> This is a blockquote.

!!! quote
    This is a Materials for MkDocs admonition quote.
```

> This is a blockquote.

Quote

This is a Materials for MkDocs admonition quote.

B.4.7.7 Code Blocks

Create code blocks using triple backticks (```) for fenced code blocks and specify a language next to the backticks before the fenced code block to highlight the syntax.

Example

````
```
bash git pull
```
````

B.4.7.8 Admonitions

Create callout out blocks for different purposes using the Materials for MkDocs admonistions feature by including three explanation points and the admonition type with the contained text indented by four spaces (`!!! note`)

Example

```
!!! note
    This is a note.

!!! tip
    This is a tip.
```

Note

This is a note.

Tip

This is a tip.

Materials for MkDocs supports the following standard admonitions:

- abstract
- bug
- danger
- example
- failure
- info
- note
- question
- quote
- success
- tip
- warning

B.4.8 Markdown Tips

- Preview your Markdown locally to ensure proper formatting before submitting your contribution.
- Keep your Markdown content organized, and use headings to structure your sections.

- There should be exactly one `heading 1` within each file.
- Use code blocks to highlight code snippets or configuration examples.
- See the official Markdown Guide for more information about Markdown.
- See the Materials for MkDocs Guide for more information about Materials for MkDocs.

October 30, 2024