Principles of good technical documentation

> 🌟 Before you start: The principles here overlap with plain language best practices and guidelines. Be familiar with the guidelines on plainlanguage.gov and the training and resources at ☰ MNIT MPCA Tech Docs Style Guide | Plain language 🌟 .

Developing Quality Technical Information: A Handbook for Writers and Editors *by the IBM Information Development team is a well-known guide to creating excellent technical documentation. The guide covers nine fundamental characteristics of high-quality technical information: accuracy, clarity, completeness, concreteness, organization, retrievability, style, task orientation, and visual effectiveness.*

Quality technical information is:

Easy to use
    Audience and task oriented
    Accurate
    Complete
Easy to understand
    Clear
    Concrete
    Follows style guidelines
Easy to find
    Organized and topic-based
    Retrievable
    Visually effective

Easy to use

Audience and task oriented

Write in terms of how users use the tool, product, or service to do their job. Understand the topics you're writing about from the audience perspective(s). It is OK to have more than one audience, but if audiences do not have much overlap, consider creating different topics or deliverables. For example, an Identity and Access Management administrator and a CBTO likely need different levels of detail and how-to instructions. If you have multiple audiences, it is OK to prioritize one audience and write for them.

Tasks (also called step-by-step or how-to instructions) are the most important topics for users because they help users get work done. Focus on real tasks, not product functions: "Create a filter" over "The Filter menu."

Not all technical documentation topics are tasks, some are concepts or reference or information. In all cases, you write for the intended audience, present information from the user's point of view, and indicate a practical reason for information.

See "Organized" below. Read more about how a framework like Diátaxis can help you identify the form of documentation to use based on user needs.

Accurate

Make sure your information:

- Is verified by testing or Subject-Matter Expert (SME) review.
- Is up-to-date, correct, and true.
- Is free from errors or mistakes, including grammar errors and incorrect cross-references or broken links.

Complete

Provide all relevant information, with sufficient detail, and include all the information you promised or indicated.

However, less is more. Save time and effort and create more user-friendly documentation by providing only the essential information the audience needs to complete a task or understand a concept. Avoid unnecessary details and focus on a clear, concise writing style. Do not overwhelm users with too much or non-essential information. This approach is called *minimalism*.

Avoid repeating information. Instead, point users to existing information when possible. Repeat information only when the user benefits. For example, to avoid danger or costly errors.

Easy to understand

Clear

Write information users can understand the first time. They should not need to reread it. You make information more clear mainly at the level of words and sentences. This often means rewriting once you have something written.

> Tools, including AI-based tools, can help you simplify and clarify, and improve readability and grammar. The usual precautions and security concerns apply. Review MNIT policies and TIAGA Standards.

Focus on meaning; use concise sentences, precise nouns and verbs, and consistent terminology. Avoid jargon and meaningless words. Use modifiers appropriately, and remove unnecessary modifiers. Write positively. More than one negative word in a sentence makes it more difficult to understand. For example, instead of "not many" use "few."

Use lists appropriately. Use ordered list when the sequence or priority is important. Use unordered lists (bullets) for items of similar importance, such as a set of alternatives. Keep list items parallel (see this five minute video from Kahn Academy on parallel structure).

Keep lists short, especially steps in tasks. Do you remember the research that says people can store seven items plus or minus two in their short-term memory? That was published in 1956. More recent research indicates our short-term memory is less; four to six items (1975, 1987) or even as few as three items (1999). You can group items to help keep lists short. Longer lists are acceptable for reference information.

Concrete

When you need to explain abstract concepts, use examples, scenarios, and analogies or similes to help make the subject more concrete.

Make examples easy to find. They should be appropriate for the audience, realistic, and accurate. Make code examples easy to copy and paste and adapt.

Follows style guidelines

A consistent and appropriate style helps your audience understand information. See https://mnitmpca.atlassian.net/wiki/spaces/TECHDOCS/pages/1849655303.

Easy to find

Organized and topic-based

Author content in short, independent topics that can be easily reused and rearranged. Each topic covers a specific subject or task and can be combined to build larger pieces of information as needed. Most technical information falls into one of three types, or

one of four forms, depending on the standards you reference.

- *Concept*: Background information users need to know to do their work. Also referred to as explanation, it answers the question "Can you tell me about…?" Do not embed contextual information into the other topic types. If there is contextual information in tasks or reference topics, pull it out an into its own topic.
- *Task*: Procedural details, such as step-by-step instructions. Include perquisites and examples. Also referred to as *how-to* topics, these are directions that guide the reader through a problem or towards a result. How-to instructions are goal-oriented. A similar document type is a tutorial. A *tutorial* is an experience that takes place under the guidance of a tutor. A tutorial is always learning-oriented.
- *Reference*: Provides quick access to facts your audience needs to look up, such as syntax rules or message explanations. For example, API docs are reference guides. Your audience isn't likely to read reference material front to back to learn something new; they consult it in order to do work or solve a problem.

Organize tasks by order of use. For example, install prerequisites, install the product, then configure the product.

Organize concepts and references into your structure in a way that makes sense to the user. This could be inline with tasks to explain or support the how-to information, or separate from tasks, arranged alphabetically, chronologically, by component or category, or by job responsibility. Work with your primary audience to determine the best organization.

Similar to our digital products, use the *progressive disclosure* design strategy. Disclose the most crucial information first, then reveal additional details only when needed. Break down complex concepts or processes into manageable chunks to avoid overwhelming the reader with too much information at once. This is also a key concept in plain language. For example, write an overview of a complex process and reference step-by-step details in tasks below. See an example from a Splunk administration guide.

Organize information consistently so users can learn to predict what information they will find and where they will find it. The sequence of topics should be consistent. Standard deliverables are always organized in the same way. For example, getting started information always follows the same sequence of topics with the same headings for all products in the same family.

Apply the same structure, templates, or rules to similar topics. Structured content is consistent, more findable, easier for audiences to use for orientation and skimming and easier for you to maintain (i.e., if you know the structure, you know where updates need to be made, reduces duplication).

See https://en.wikipedia.org/wiki/Topic-based_authoring and Diátaxis and https://medium.com/@davidagash/a-painless-introduction-to-information-typing-d06041013fd5.

More on topic-based and structured authoring (advanced technical writing)

Topic-based authoring is essential when you have content teams responsible for multiple deliverables for multiple audiences and supporting multiple release versions. For example, user guides and admin guides can share some topics, or version 2.0 and version 3.0 can share 90% of their topics, there are just additional topics for new or updated features and functionality. It is most efficient and accurate to write a topic once and reuse everywhere than to maintain duplicate content across many deliverables.

Structured authoring means applying strict and standard formatting across all topics. Structured authoring is frequently done in semantic markup, like XML. Darwin Information Typing Architecture (DITA) is a popular open source XML architecture for topic-based technical content.

Authoring in semantic markup enables teams to:

- Separate content from presentation, so the writer focuses on the information rather than how it looks on the page.
- Break down content into topics or components that can be independently edited and reused in different contexts (and translated more easily and affordably).
- Single source content output. The same source can be output as online help, PDF manuals, so on or apply different stylesheets for different channels.

Topic-based and structured authoring can be too complex and expensive for small teams or teams where content is contributed by people with roles other than full-time technical writer. The good news is you don't have to implement XML authoring and component content management tools to get the usability and accessibility benefits of consistent structure and separating the

source content from the presentation layer. Many authoring tools provide templates, keyword or tagging functionality, rules checkers, and style checkers.

Retrievable

Make it as easy as possible for users to find information.

Facilitate navigation and search. Use a table of contents. Write headings consistently and clearly identify the topic. Do not duplicate titles or headings. For example two topics titled "Overview" is not useful, instead "Overview of GitHub Actions" and "Overview of GitHub Runners."

Make links helpful. The wording of links and cross-references describe the target.

Visually effective

Use graphics that are meaningful and appropriate. Balance the number and the placement of visual elements.

Always use existing templates and stylesheets.