
>>> network .toCode()
The Joy of Writing Documentation

Agenda

What is Documentation

The Documentation Framework

Writing Documentation

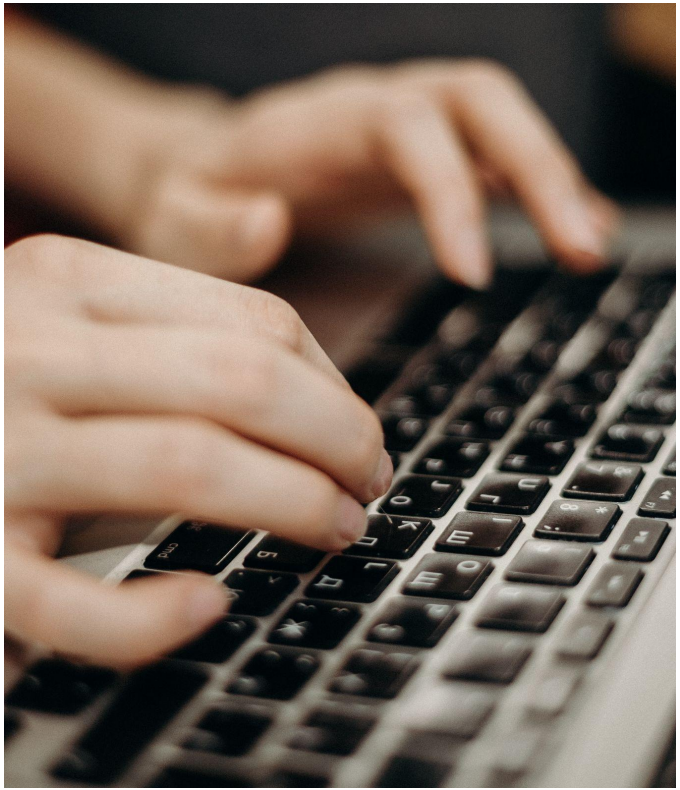
Writing Tips

Common Mistakes

Questions

What is Documentation?

>>> What is Documentation?



Documentation is any communicable material that is used to describe, explain or instruct regarding some attributes of an object, system or procedure, such as its parts, assembly, installation, maintenance and use.

Wikipedia

>>> Documentation Goals & Components

Primary goal is to break down complex information into understandable explanations.

To achieve this primary goal we identify five components:

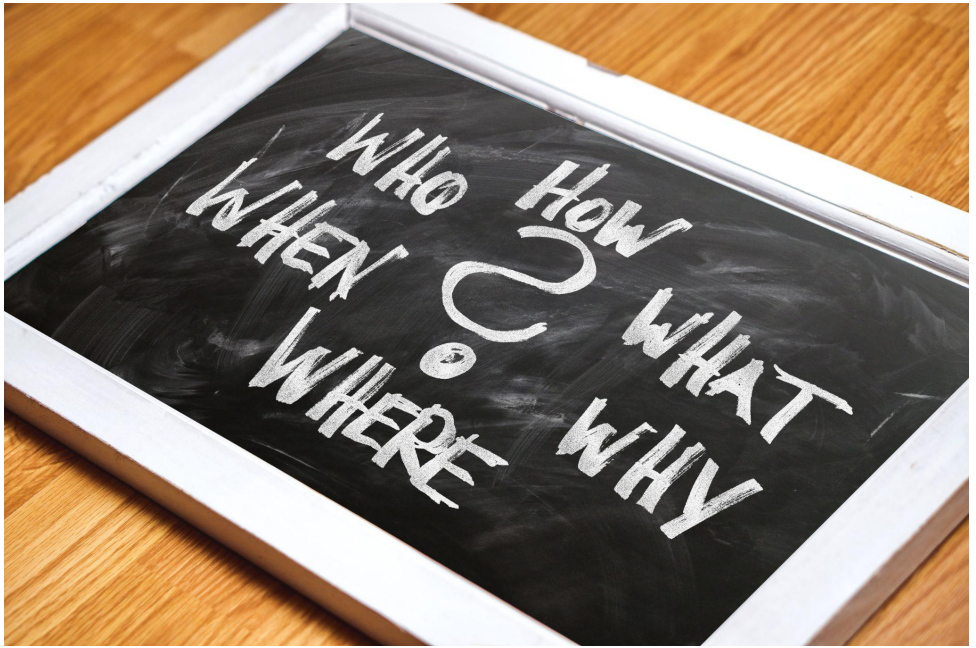
- Purpose/intent
- Target audience
- Content
- Organization
- Style

>>> Documentation Goals & Components

Primary goal is to break down complex information into understandable explanations.

To achieve this primary goal we identify five components:

- Purpose/intent
- Target audience
- Content
- Organization
- Style



>>> So Why Write it?

Many developers don't like to write documentation, but there are numerous advantages:

- Provides a record of information in relation to the software developed
- Improves knowledge sharing
- Makes code easier to maintain and update
- Improved collaboration between developers
- Helps new hires get up to speed faster
- Allows developers to look back on past code to understand why they did what they did

The Documentation Framework

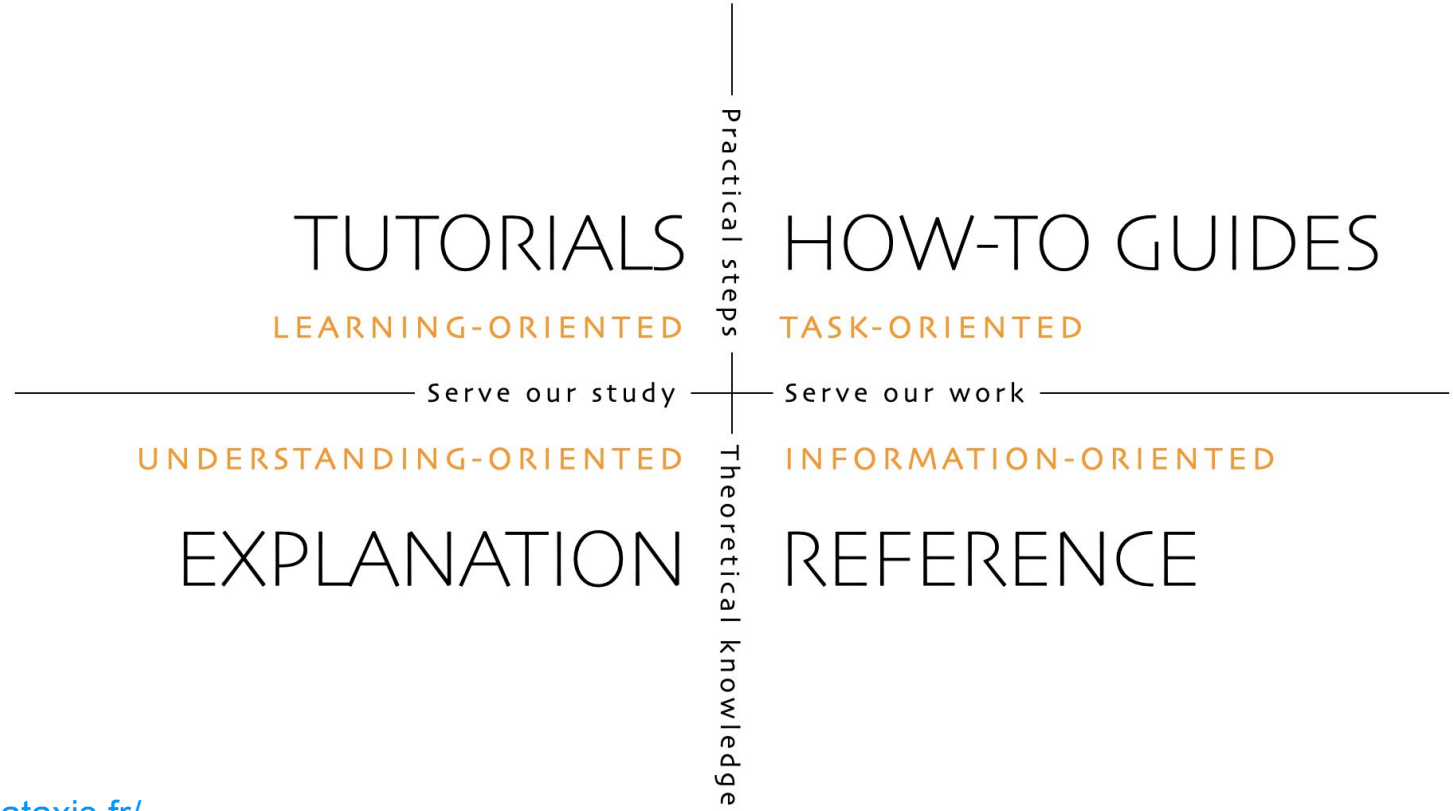
>>> The Documentation Framework

The first step to technical writing is having a plan.

The Diataxis Framework

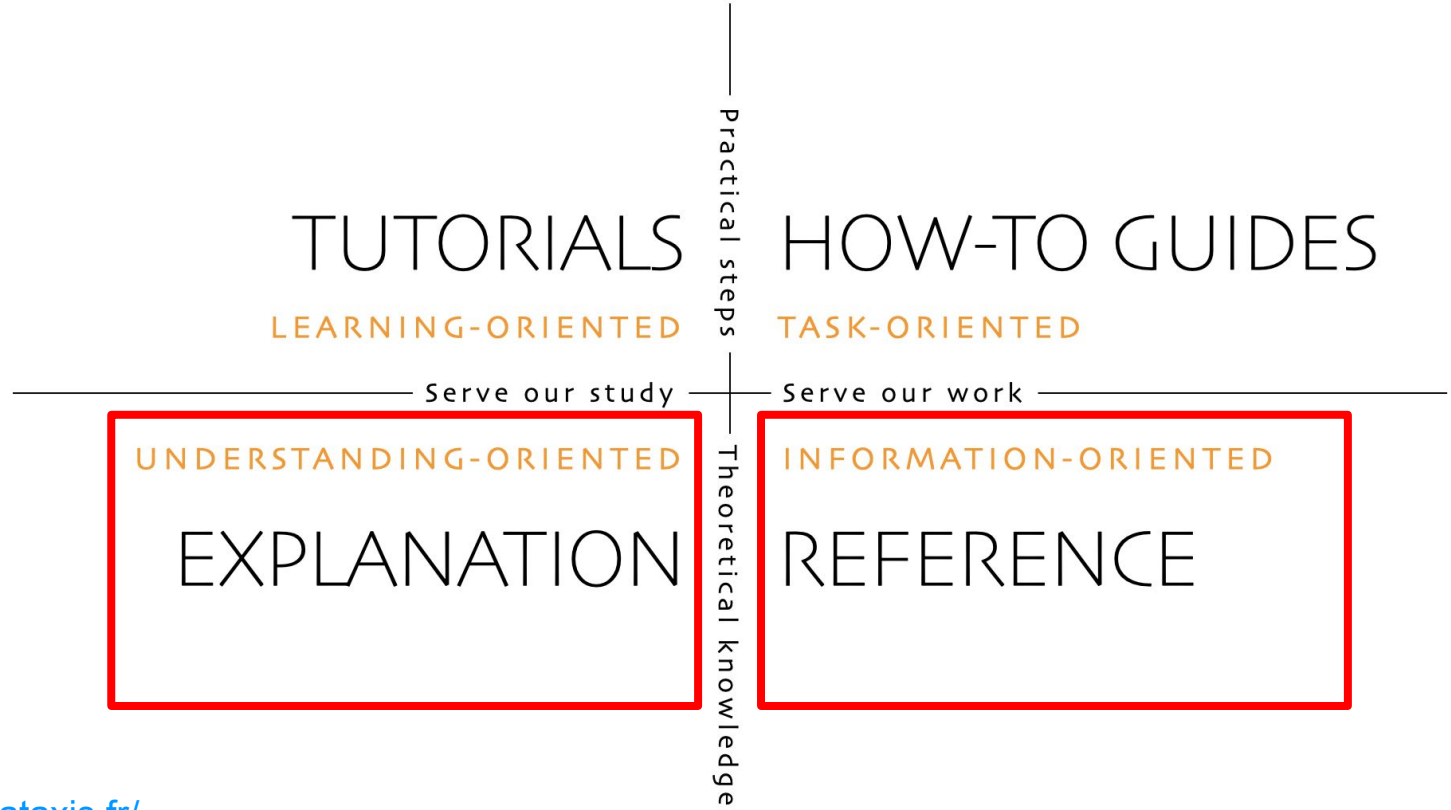
- Identifies four modes of documentation
- Each mode:
  - Corresponds to a different user need
  - Fulfills a different purpose
  - Requires a different approach to its creation

>>> The Diátaxis Approach to Technical Documentation



https://diataxis.fr/

>>> The Diátaxis Approach to Technical Documentation



https://diataxis.fr/

Writing Documentation

>>> Writing Documentation

A *Technical Description* of a system or part of system and how to operate it.

- Docstrings
- Code Comments
- API Documentation

https://diataxis.fr/reference/

>>> Reference - Docstrings

A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. - *Python PEP 257*

```
75  def split_interface(interface: str) -> t.Tuple[str, str]:
76      """Split an interface name based on first digit, slash, or space match.
77
78      Args:
79          interface: The interface you are attempting to split.
80
81      Returns:
82          The split between the name of the interface the value.
83
84      Examples:
85          >>> from netutils.interface import split_interface
86          >>> split_interface("GigabitEthernet1/0/1")
87          ('GigabitEthernet', '1/0/1')
88          >>> split_interface("Eth1")
89          ('Eth', '1')
90          >>>
91      """
```

>>> Reference - Docstrings

Docstrings are described in PEP 257.

- They should provide users with a brief overview of the object.
- Used with functions, classes, and files
- Always use `"""triple double quotes"""`
- No blank lines before or after Docstring
- First line always ends in a period
- Can be:
  - One-line
    - Simple use cases
    - Single sentence on a single line
  - Multi-Line
    - Provides additional info beyond basic description (inputs, returns, raises, etc.)
    - First line begins right after triple double quotes `"""My Docstring...`
    - Closing triple double quotes must be on its own line

>>> Reference - Code Comments

```
20          - "{{ 'acl' if ('SUDO_USER' in ansible_env and ansible_env.SUDO_USER != 'root') else [] }}"
21          # TODO need to make this uniform in the distros
22          # - "{{ nautobot_ldap_packages if nautobot_ldap_enabled else [] }}"
23          # - "{{ 'git' if nautobot_git else [] }}"
```

```
1     ---
2     # Don't edit this file, copy the desired configuration variables to group_vars
3
4     # update firewalld rules?
5     manage_firewall: true
6
7     # Do we have sudo privileges on the server ('/bin/sh -c' required)?
8     root_privileges: true
9
10    # Install EPEL package on Redhat?
11    nautobot_install_epel: false
```

>>> Reference - Code Comments

Writing Comments

- Write comments as you write the code
  - Include comments which may not be needed, you can remove them when cleaning up/refactoring
  - When in doubt, add a comment
- Make comments meaningful
  - Should not duplicate code
  - Resolve confusion, don't add to it
- Use codetags to identify todos, fix mes, etc.
  - Described in PEP 350, though rejected as an official standard
  - Some code editors support highlighting codetags
  - Examples:

```
1   # NOTE: Note Comment...
2   # TODO: To do comment...
3   # FIXME: Fix me comment...
4   # BUG: Bug comment...
5   # HACK: Hack comment...
```

>>> Reference - Other Types of Code Commenting

Write your code as if comments didn't exist (but still use comments)

- Smart Object Naming
  - Give objects (variables, functions, classes, etc) names that are meaningful and descriptive
    - This helps inform others what that particular object is supposed to do or what it represents
  - Use `initial_interface_config` and `post_interface_config` instead of `interface_config1` and `interface_config2` to describe the initial and post config states of an interface when making changes
  - Use `parse_iosxe_config()` instead of `parse_ios_config()` to describe a function that parses a Cisco IOS-XE configuration into common format (just ios can cause conflict between IOS-XE and IOS-XR)
- Type Hinting
  - Helpful way to communicate what type a variable is expected to be (string, integer, class type, etc.)
  - Some code editors, such as VS Code, are type-aware and use type hinting to make code suggestions

```python
import typing as t

def split_interface(interface: str) -> t.Tuple[str, str]:
    """Docstring."""
    head = interface.rstrip(r"/\0123456789. ")
    tail = interface[len(head) :].lstrip()  # noqa: E203
    return (head, tail)
```

>>> Writing Documentation

A *Discussion* which clarifies or illuminates a topic leading to better understanding of that topic.

- README.md Files
- SADAs
- High Level Designs
- Customer Facing Documents

https://diataxis.fr/explanation/

>>> Explanation - Overview

Can be thought of as a conversation

- Use complete sentences and paragraphs
- Talks about the subject in detail
- Builds reader's understanding on subject
- Is not instructional or a technical reference
  - This is saved for the other modes of the diataxis
  - Example: Suggesting alternative software to customer
    - Included: Why the recommendation is made and how it will impact the business
    - Not Included: how that new software would be installed or how to use it

>>> Explanation - SADA Example

4.1 Source of Truth

The Source of Truth (SoT) for a network contains all systems and databases that act as the authority for data about one or more record types in configuration management. For instance, IP address management (IPAM), data center infrastructure management (DCIM), CMDB, DNS, VM orchestration, and other such databases together serve as the authoritative Source of Truth for data about a network. In practice, it is rare for one tool to serve as the entire Source of Truth for a network. The tool responsible for each discrete record type is often called the System of Record (SoR).

It is crucial to never have more than one System of Record for a given record type. For instance, it is not possible to use both a spreadsheet and a stand-alone application as the Source of Truth for all IP addressing within a network because this arrangement does not provide a remedy when the two sources of data disagree. While it is acceptable to publish the information in different places, only one SoR may be considered authoritative at any time. For example, an IPAM application and a DNS zone file cannot both have ownership of DNS; one or the other must be designated as *authoritative*.

Below is an example of common data domains and systems suitable to act as a System of Record for those domains:

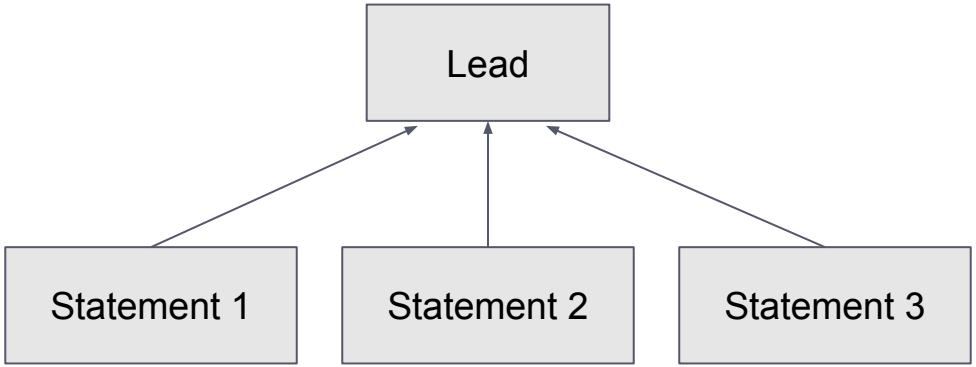
Writing Tips

>>> Identify the focus of the writing..

..And Stick to it

Write the Lead - A short sentence defining the key aspects of the content

- Defines the **content** component of technical writing
- Writing this first ensures you identify the fundamental content before writing
- Every statement must be related to the lead and offer additional details and/or context
- Helps keep your writing focused on the subject while preventing and identifying drift
- If it's not related, remove it or rewrite so it is related



>>> Lead Example

4.1 Source of Truth

The Source of Truth (SoT) for a network contains all systems and databases that act as the authority for data about one or more record types in configuration management. For instance, IP address management (IPAM), data center infrastructure management (DCIM), CMDB, DNS, VM orchestration, and other such databases together serve as the authoritative Source of Truth for data about a network. In practice, it is rare for one tool to serve as the entire Source of Truth for a network. The tool responsible for each discrete record type is often called the System of Record (SoR).

It is crucial to never have more than one System of Record for a given record type. For instance, it is not possible to use both a spreadsheet and a stand-alone application as the Source of Truth for all IP addressing within a network because this arrangement does not provide a remedy when the two sources of data disagree. While it is acceptable to publish the information in different places, only one SoR may be considered authoritative at any time. For example, an IPAM application and a DNS zone file cannot both have ownership of DNS; one or the other must be designated as *authoritative*.

Below is an example of common data domains and systems suitable to act as a System of Record for those domains:

>>> Outline the Key Points Before Writing

Defines the final component: **Organization**

- Outlines help structure the content based on key points
- Each key point references a paragraph or two, key points can have subpoints
- Every key point should add additional information or context to the **Lead**
- Every subpoint should add additional information or context to the parent **Key Point**

[LEAD STATEMENT]

- [KEY POINT 1]
  - [SUBPOINT 1]
  - [SUBPOINT 2]
- [KEY POINT 2]
  - [SUBPOINT 1]
  - [SUBPOINT 2]
- [KEY POINT 3]

>>> Outline the Key Points Before Writing

[RECOMMENDATION - UTILIZE NAUTOBOT GOLDEN CONFIG APP FOR CONFIG RENDERING]

- [KP1 - CLIENT HAS SOME CONFIG RENDERING ALREADY COMPLETED]
  - [CLIENT HAS INTERNAL SYSTEM TO RENDER CONFIGS]
  - [ABOUT 20% ARE JINJA2, 80% ARE BASELINE CONFIGS IN TEXT FILES]

>>> Utilize the Flesch Reading Score

The Flesch Reading Ease Test:

- Analyzes text and provides a score between 1 and 100
- Helps writers gauge how understandable their writing is.

https://charactercalculator.com/flesch-reading-ease/

Two primary metrics provided:

- **Reading Score** - Overall score from formula - Larger numbers are better
- **Reading Level** - Indication of grade level required to read - Smaller numbers are better

Formula:

```
0.39 × (total words ÷ total sentences) + 11.8 × (total syllables ÷ total words) - 15.59
```

>>> Utilize the Flesch Reading Score

Score	Grade	Summary
90 - 100	5th grade	Very easy to read
80 - 90	6th grade	Easy to read
70 - 80	7th grade	Fairly easy to read
60 - 70	8th & 9th grade	Plain English
50 - 60	10th to 12th grade	Fairly difficult to read.
30 - 50	College	Difficult to read.
10 - 30	College graduate	Very difficult to read
0 - 10	Professional	Extremely difficult to read

>>> Great Writing is Good Rewriting

First drafts are not the final drafts:

- Meant to get ideas on paper
- Focus quality of content, not quality of writing

Once the content is finished and nothing more to add:

- Analyze the writing to identify what can be improved
- Rewrite and reorganize in ways to improve the quality of the writing
- The Flesch reading score is an extremely helpful tool to gauge improvement

>>> Great Writing is Good Rewriting

There are no current needs to create a complex event-driven network automation system, leveraging platforms such as Ansible AWX, and most of these integrations can be achieved externally via API calls to run Nautobot Jobs via API calls, or internally to Nautobot, via webhooks or Jobhooks that will trigger actions for changes in the Source of Truth elements.

Reading Score: 2.10

Reading Level: 27.37

There is no immediate need to replace the existing systems with a complex event-driven network automation system at this time. Most integrations can be achieved by running Nautobot Jobs using API calls or Jobhooks to make changes to the Source of Truth.

Reading Score: 45.63

Reading Level: 11.99

Aim to improve, not perfect

Common Mistakes

>>> Overuse of compound & Complex Sentences

Independent Clause - a group of words completing a complete thought forming a simple sentence

- Subject + Verb + Object

Dependent Clause - a group of words that does not form a complete sentence and cannot stand by itself

Compound Sentence - a sentence that connects two independent clauses.

"Be yourself; everyone else is already taken." —Oscar Wilde

"You will face many defeats in life, but never let yourself be defeated." —Maya Angelou

Complex Sentence - a sentence with one independent clause and at least one dependent clause.

"It doesn't matter how slowly you go as long as you don't stop." —Confucius

"Because things are the way they are, things will not stay the way they are." —Bertolt Brecht

- Can make documentation difficult and confusing to read
- Does not mean they should be avoided all together, but used where they make sense
- Ask yourself, "can this sentence be improved by separating?"

>>> Overuse of Passive Voice

A sentence or clause where the subject of the sentence is being acted upon by the verb or action.

- Passive voice produces a sentence where the subject **receives** the action
- Active voice is where the subject **performs** the action

Passive	Active
A great deal of meaning **is conveyed** by a few well-chosen words.	A few well-chosen words **convey** a great deal of meaning.
Our planet **is wrapped** in a mass of gases.	A mass of gases **wrap** around our planet.
Waste materials **are disposed** of in a variety of ways.	The city **disposes** of waste materials in a variety of ways.

Problems with Passive Voice:

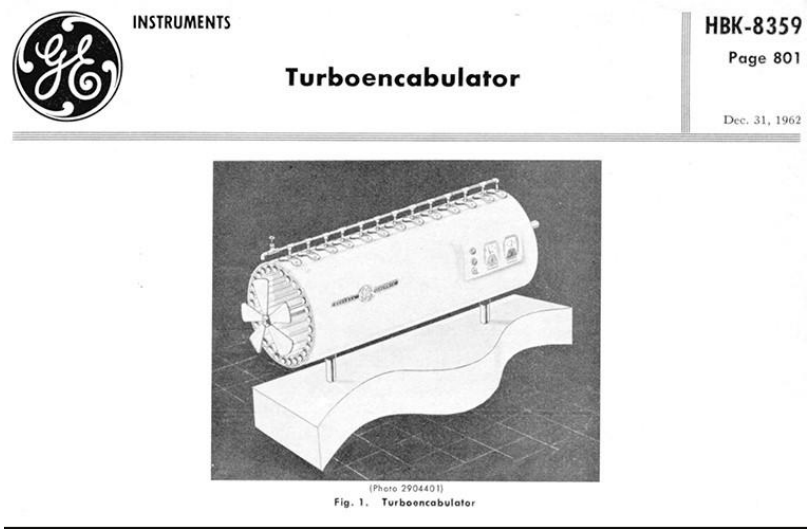
- Creates Unclear, less direct, and wordy sentences
- Can become tedious to read, leading to disengaged readers
- Can be difficult to identify for inexperienced writers

Passive voice does have its place in writing, though it should be used intentionally and sparingly.

>>> Obfuscation through complexity and technical jargon

Popular Example: The Turbo Encabulator

“For a number of years now, work has been proceeding in order to bring perfection to the crudely conceived idea of a transmission that would not only supply inverse reactive current for use in unilateral phase detractors, but would also be capable of automatically synchronizing cardinal grammeters. Such an instrument is the turbo encabulator.”


INSTRUMENTS

HBK-8359

Page 801

Turboencabulator

Dec. 31, 1962

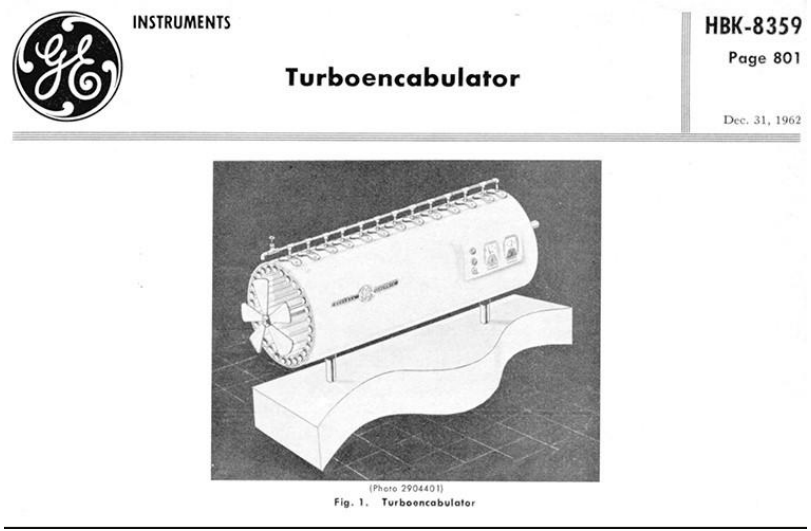
(Photo 2904401)

Fig. 1. Turboencabulator

>>> Obfuscation through complexity and technical jargon

Popular Example: The Turbo Encabulator

"For a number of years now, work has been proceeding in order to bring perfection to the crudely conceived idea of a transmission that would not only supply inverse reactive current for use in unilateral phase detractors, but would also be capable of automatically synchronizing cardinal grammeters. Such an instrument is the turbo encabulator."


INSTRUMENTS

HBK-8359

Page 801

Dec. 31, 1962

Turboencabulator

(Photo 2904401)

Fig. 1. Turboencabulator

- Comprehension is lost when using too much technical jargon
- What is understandable by you can be lost on your readers
- Know your audience and write at their level


>>> network .toCode()
Questions?