Creating Your Controlled Language

User Guide

Rochelle Fisher

Creating Your Controlled Language: User Guide

Rochelle Fisher

Publication date 31 July 2025

Abstract

This version of the documentation is for technical writers or developers who have Python installed, can run the scripts given in this document, and can continue with the linguistic analysis to create a proprietary controlled language.

Sample Sources:

- Rubicon Communications LLC. (Apr 25, 2025) netgate© Security Gateway Manual: Amazon AWS. © Copyright 2025 Rubicon Communications LLC. https://docs.netgate.com/manuals/pfsense/en/latest/aws-vpn-appliance-security-gateway-manual.pdf
- OpenCTI. (2025) OpenCTI User Guide: Manual Creations. © 2025 Filigran. All rights reserved. https://docs.opencti.io/latest/usage/manual-creation/
- MISP. (2024) MISP User Guide: A Threat Sharing Platform. GPL and CC-BY-SA 4.0 international.

Table of Contents

Introduction to Controlled Languages

A Controlled Language (CL) is set of rules for grammar, sentence length, and vocabulary. The rules define words and grammar as allowed or not allowed. This enforces consistency for all your documentation and conforms to or builds an industry standard for your products.

Intended Audience

This document is for a developer or technical writer with coding experience. Python installation is a requirement. If you are a linguist or responsible for the content of your organization, but do not have Python, you can share this document with a developer or IT. They can run the scripts, and you can continue with the CL itself when they are done.

Scope of Project

This is a personal project of Rochelle Fisher, July 2025. The goal is to make a sample of technical writing and a sample of a proprietary CL for cybersecurity.

If you use part, parts, or all of this project for personal use, for profit, or for an organization (non-profit or for-profit), please include a reference to this project or its documentation.

Reference Example:

`Rochelle Fisher, Sample Cyber Security CL, version 1.0, July 2025`.

Workflow

These are the procedures in this project:

1. Collect samples.

   You will create a CL from your documentation. This project creates a CL for cybersecurity from online, GPL, or unlicensed content.

2. Aggregate the samples to one text file.

3. Run the Python script on the text file, to get a CSV file of each word and its number of instances used.

4. From the Python output, create the dictionary file.

5. For each word, starting with the least used, decide its part of speech, allowed or not allowed, audience, and examples.

6. Create grammar and content rules.

   This project will start with ASD-STE 100 rules and then customize them.

7. Out of scope: Select an AI tool, or create a Python script, to test content for CL conformance (we like writer.com).

Collecting Samples

This is the cloudiest part for a sample project because we do not want to infringe on copyright laws. We will not use the content as-is, available for consumer use. We will use it only to get the vocabulary for our CL. When you create your CL, use your organization's documentation.

Sources:

- Rubicon Communications LLC. (Apr 25, 2025) netgate© Security Gateway Manual: Amazon AWS. © Copyright 2025 Rubicon Communications LLC. https://docs.netgate.com/manuals/pfsense/en/latest/aws-vpn-appliance-security-gateway-manual.pdf
- OpenCTI. (2025) OpenCTI User Guide: Manual Creations. © 2025 Filigran. All rights reserved. https://docs.opencti.io/latest/usage/manual-creation/
- MISP. (2024) MISP User Guide: A Threat Sharing Platform. GPL and CC-BY-SA 4.0 international.

You can collect your samples or complete content as PDF, DOC, DOCX, HTML, or text. Aggregate all the samples to one file. Convert the file to plain text. Keep this file. You will use it as input the Python script and as the main source for word analysis.

Creating the Word List

Create and Run the Word Counter Python Script

To create the dictionary, we start with the unique words used and how many times each word is used in our sample. This Python script will give us a file formatted for easy import to a spreadsheet.

Procedure 1. To create and run the python script word_counter.py:

1. Save your aggregated sample as `sampleAgg.txt`.

2. Copy this script in a text editor.

3. In the same folder as `sampleAgg.txt`, save the file as `word_counter.py`.

4. Run the python script:

   python word_counter.py > dictionary.csv

```
import re
import sys
from collections import Counter

def analyze_document_frequency(filepath: str = "sampleAgg.txt"):
    """
    Analyzes a text document to count word frequency.

    This function reads a text file, converts its content to lowercase,
    extracts words between 4 and 15 characters long,
    and counts the occurrences of each extracted word. It prints
    the words sorted by frequency in descending order, then
    alphabetically for ties.

    Args:
        filepath (str): The path to the input text file.
        Defaults to "sampleAgg.txt".

    Returns:
        None: Prints the word frequencies to standard output.
        Errors are printed to stderr.
    """
    try:
        # Open and read the document, converting content to lowercase.
        # 'utf-8' encoding is specified for broad compatibility.
        with open(filepath, 'r', encoding='utf-8') as document_file:
            text_string = document_file.read().lower()
    except FileNotFoundError:
        # Handle the case where the specified input file does not exist.
        print(f"Error: '{filepath}' not found.", file=sys.stderr)
        return
    except Exception as e:
        # Catch other general exceptions during file reading.
        print(f"Error while reading: {e}", file=sys.stderr)
        return
    # Use regular expression to find all words.
    # '\b' ensures whole words. '[a-z]{4,15}' matches lowercase
    # letters between 4 and 15 characters.
    match_pattern = re.findall(r'\b[a-z]{4,15}\b', text_string)

    # Count word occurrences using collections.
    # Counter for efficiency.
    word_counts = Counter(match_pattern)

    # Sort words: primary key is count (descending),
    # secondary key is word (ascending).
    # Lambda function creates a tuple for sorting:
    # (-count) for descending, then word for ascending.
    sorted_words = sorted(word_counts.items(), \
    key=lambda item: (-item[1], item[0]))

    print("Word Frequencies:")
    print("-----------------")
    # Iterate through the sorted words and
    # print each word and its count, separated by a pipe.
    for word, count in sorted_words:
        print(f"{word} | {count}")

if __name__ == "__main__":
    # Make sure the function runs only when
    # the script is executed directly,
    # not when imported as a module.
    analyze_document_frequency()
```

Note on Code Source: This code started with the python script in *Alchemy of Tomes* [Fisher (2020)], which does not work in the latest Python versions. We used Gemini (Google AI) to update this script.

Create the Dictionary File

With three small samples of cybersecurity documentation from different organizations, the Python script gave us 4755 unique words. Your sample will be much larger. Your goal is to create a cybersecurity dictionary of approximately 1,000 words. This does not include product names or company-specific words.

Prerequisites: Run the Python script.

Effort: Use the output file from the Python script to create the dictionary file. This will require approximately ten minutes.

Procedure 2. To create the dictionary file:

1. Import the output file to a spreadsheet.

   We will use Google Sheets. You can use Microsoft Excel or similar.

   Our Python script uses a pipe (|) as a delimiter between the word and its frequency of use. When you import the file, set the pipe as the delimiter.

2. Set the column headers.

   If you know that you will use a specific AI-driven tool, such as writer.com or jasper.ai, change the dictionary headers and values to work with the acquired tool.

If you do not have a checker tool yet, change row 1 to be these headers: `Word, Count, Part of Speech, Definition, Good Example, Bad Example, Allowed?, Audience, Alt1, Alt2`.

3. If there are rows between the header row and the first word, delete them.
4. In a separate tab, enter your audience personas in one column. Make sure `all` is in this list.
5. In the main tab, Audience column, set data validation to select the persona from the list.

The script sorted the output by highest frequency to lowest. Start with the words most used. These will be the easiest.

Getting Started with the Dictionary

Work the First Words

The first words that you set up in your dictionary will be the easiest and will have the most impact.

Prerequisites: Have the dictionary file in a spreadsheet.

Effort: This will require less than an hour.

Procedure 3. To work through the most used words in your dictionary:

1. In the word with the highest count, set the Part of Speech (PoS).

   A typical result for the most used words are the names of your organization or product. ASD-STE calls these *technical names*. Set the PoS of the technial names in the top results as `Name`.

   This lets you filter for proper nouns which change more often than regular nouns.

   For example, the company name will change if your organization delivers a white label product.

2. Enter the PoS for the other most common words, such as *the that this from*. When you get to a word that is may be used in multiple parts of speech and is not a common word for all English content, skip it.

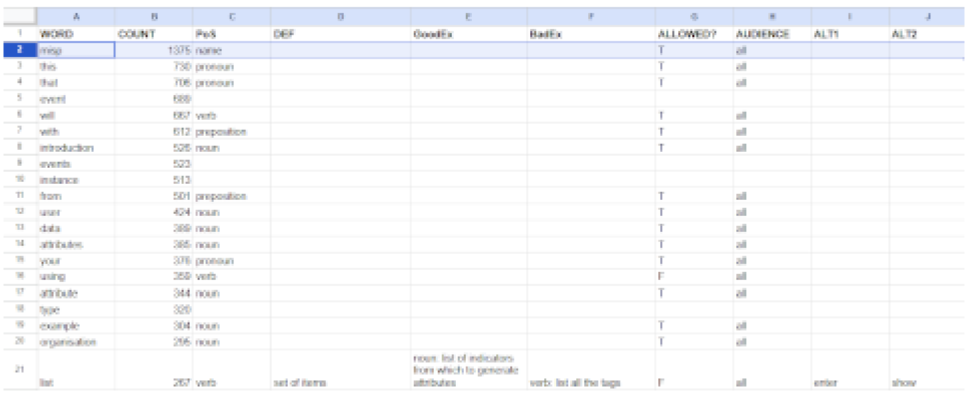
3. In the Allowed column, enter `T` (for true) or `F` (for false).

   Most of these first words will be allowed.

4. In the Audience column, enter `all` or select a persona from the list, if you are sure this word will be allowed only for this persona.

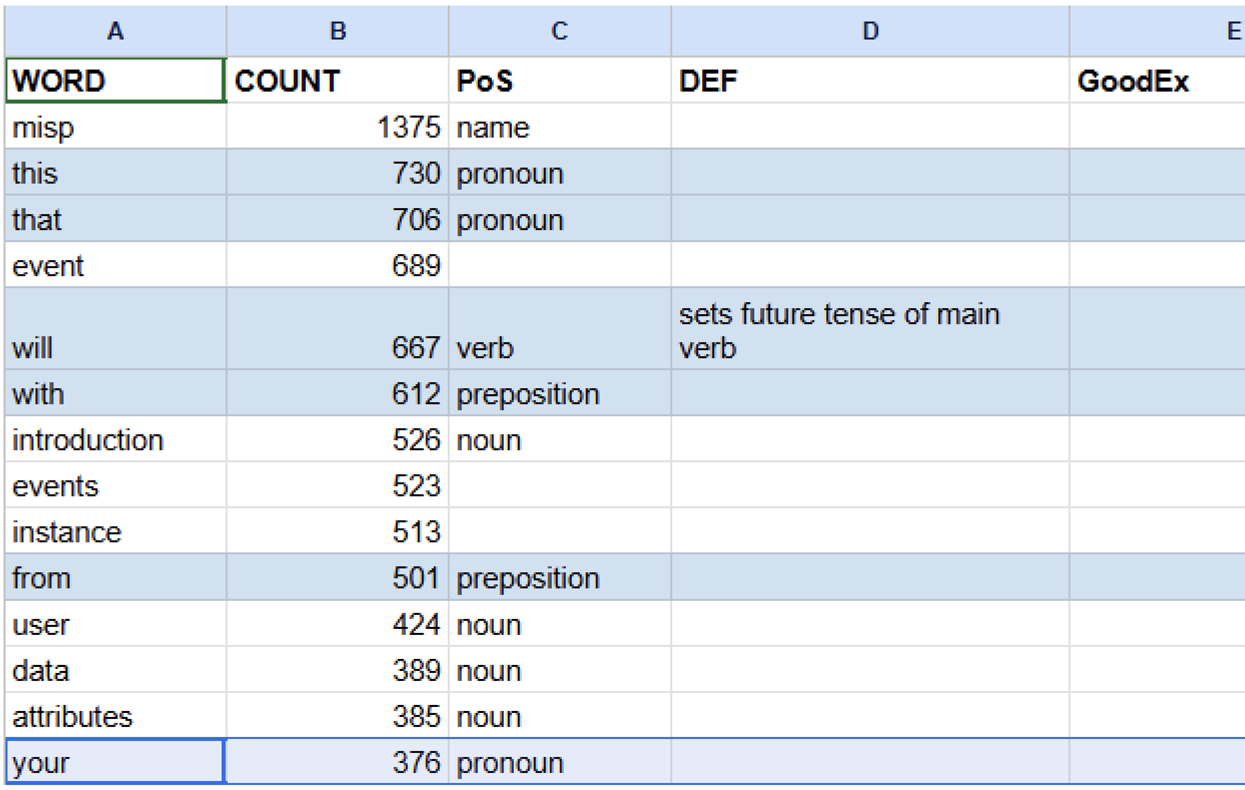
Examples of the First Words

- In our example, one of the product or company names is the word with the highest frequency. We set its PoS as `Name` and set Allowed to `T`. We set Audience to `all`. We do not set a value for the other columns yet.



	A	B	C	D	E	F	G	H	I	J
1	WORD	COUNT	PoS	DEF	GoodEx	BadEx	ALLOWED?	AUDIENCE	ALT1	ALT2
2	mqp	1375	name				T	all		
3	this	730	pronoun				T	all		
4	that	706	pronoun				T	all		
5	event	689								
6	will	667	verb				T	all		
7	with	612	preposition				T	all		
8	introduction	526	noun				T	all		
9	events	523								
10	instance	513								
11	from	501	preposition				T	all		
12	user	424	noun				T	all		
13	data	389	noun				T	all		
14	attributes	385	noun				T	all		
15	your	376	pronoun				T	all		
16	using	359	verb				F	all		
17	attribute	344	noun				T	all		
18	type	320								
19	example	304	noun				T	all		
20	organisation	296	noun				T	all		
21	list	267	verb	set of items	noun: list of indicators from which to generate attributes	verb: list all the tags	F	all	enter	show

- In the top words, we have many that are clearly to be allowed: *this that will with from your*. These are pronouns, prepositions, and modular verbs. In a technical writing dictionary for native speakers, it does not give a lot of information to define the PoS. Is *that* a pronoun, adverb, conjunction, or determiner? We set a rule that we do not use minimalist rules. If *that* helps make a sentence easier to understand, we use it.



	A	B	C	D	E
	WORD	COUNT	PoS	DEF	GoodEx
	misp	1375	name		
	this	730	pronoun		
	that	706	pronoun		
	event	689			
	will	667	verb	sets future tense of main verb	
	with	612	preposition		
	introduction	526	noun		
	events	523			
	instance	513			
	from	501	preposition		
	user	424	noun		
	data	389	noun		
	attributes	385	noun		
	your	376	pronoun		

- We have words that are industry standard for software technical writing: *introduction user data attributes attribute example organisation*. Later, we will enter a specific deinition for these words. For now, we set the PoS, set Allowed to `T`, and set Audience to `all`.

  attribute and *attributes* are two items.

  We restrict the values of the Word and ALT columns to one word or phrase, to prepare for a checker tool that might be a simple Python script.

  If you know that you will use a specific AI-driven tool, such as writer.com or jasper.ai, change the dictionary headers and values to work with the aquired tool.

- See *organisation*. We know we need this word, but this spelling is British. We know our rules will tell us to use American spelling. In this case, we will make a decision for our dictionary without analysis, or to put it more accurately, despite analysis. The spelling of *organisation* is the most commonly used form, but we will not use it. We will use the American spelling.

1. Sort the complete range of dictionary alphabetically, by *Word*.

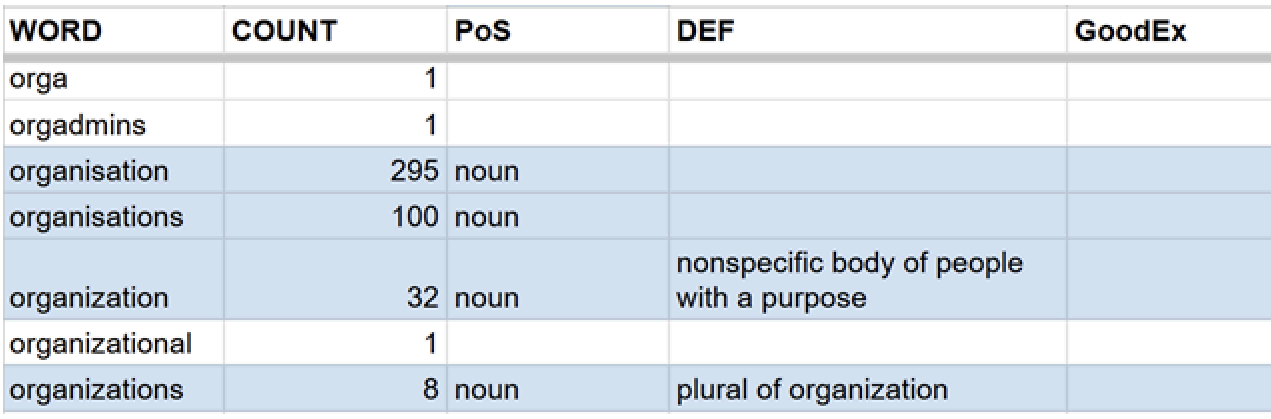
2. Set *organization* and *organizations* as allowed (`T` in Allowed).

3. Set the British *organisation* and *organisations* as not allowed (`F` in Allowed).

4. In the allowed words, set the definition: `nonspecific body of people with a purpose`.

   We will use *organisation* for a company, nonprofit org, military base, and all similar bodies.

5. In ALT1 for *organisation* and *organisations*, enter *organization* and *organizations*.



WORD	COUNT	PoS	DEF	GoodEx
orga	1			
orgadmins	1			
organisation	295	noun		
organisations	100	noun		
organization	32	noun	nonspecific body of people with a purpose	
organizational	1			
organizations	8	noun	plural of organization	

Why not add *organizational* on the fly? Answer: It is used only one time. We will analyze the text and find a more common rewrite in that one sentence. Or maybe that one instance is for *Organizational Unit (OU)* in Active Directory. If so, we will make that phrase a "word".

Sort the range by Count. We are done with the easiest words. Next: analyze words in their syntax and by content to make decisions for the dictionary.

Analyzing Words

Work the Next Words

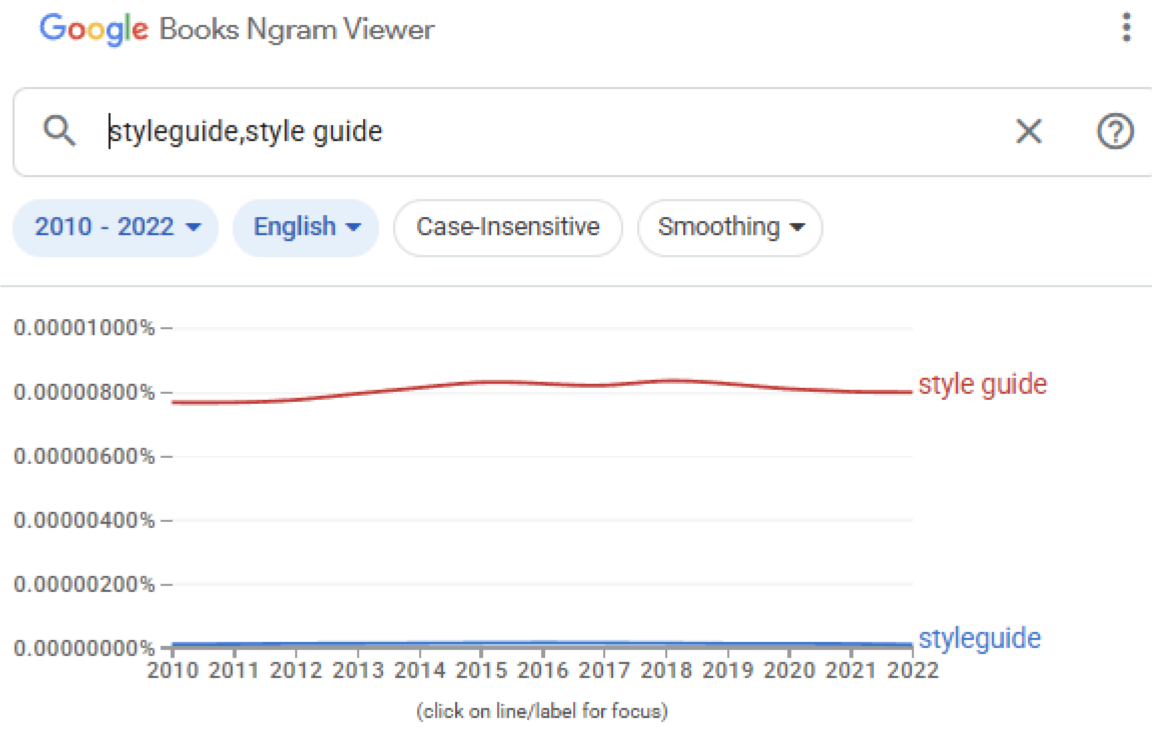
You will analyze how your dictionary words are used in your sample. Begin with the top words.

Procedure 4. To analyze words:

1. Open your aggregated sample document that you used as input for the Python script.
2. Search for the word to analyze. We like to use Notepad++, to get all the results in a list, each result with context.
3. From the context, see if the word is used in more than one part of speech. If it is, and it is not obvious which is the most common, count the instances of each PoS.

   I usually paste or import the results in a spreadsheet and then use COUNTIF, filter, or pivot table.
4. See if the word, in the allowed PoS, is used with more than one defintion. If it is, note the defitions.

   Use counts, industry standards, and internet searches to get the best defintion. Get the ASD-STE 100 document and see what their dictionary says about the word. NGRAM is also a good tool.



5. See if the draft definition works in a large random sample, or in all uses. If you decide to make the word not allowed, see if the chosen alternate word or syntax works.
6. Update the rules, definition, PoS. Enter good and bad examples.
7. If not allowed, make sure the words you set as alternatives are allowed and configured in the dictionary for PoS and definition, with examples.

Examples of Word Analysis

Let's start with *organizational*.

Procedure 5. To set *organizational* as a word in our CL:

1. Open the file with all your textual content. We named this file `sampleAgg.txt`.
2. Search for *organizational*.

   We found this sentence (owned by one of the organizations from which we used samples):

   > Produce intelligence that will be embedded into organizational workflows and would serve decision-makers.

3. Analyzing the use of *organizational*, we see ambiguity. Does the author mean that the workflows are organized? That they are for the organization? That there are different workflows for different groups in the hierarchy?

   If we remove the word, it does not change the meaning, as far as we can see. We decide that this word is not allowed. If we had access to the SME, we would discuss their meaning and find alternatives.

   Suggestion: You have access to your SMEs. Set your CL words as best as you can. Then, discuss multiple words with similar issues. Edit your CL for alternative words and other decisions.

The next task is to work through the top words that are not obviously allowed or not allowed. Sort the dictionary by Count.

Our first word to work through is *event*.

Procedure 6. Analyzing *event*:

1. In Notepad++, search for *event* in the sample content.

   We get 1417 results, for *event* and *events*.

2. Read the results. Make the defintion draft. For example:

   a security incident detected on the secured network

3. Fix the defintion as you read more result lines.

   For example: The fifth result is for a procedure to create an event in the security application. We learn that an event always includes threat intelligence data and usage. We update the definition to:

   > object that contains a cybersecurity incident, report, or finding, with attributes, identifiers, and other data for analysis, prevention, and mitigation

   We see that *event* is used with other definitions.

   - A system or user action

     This is a technical name for a specfic product. For our general cybersecurity dictionary, it does not fit. We can communicate actions in errors and logs with their names, without the use of *event*.

   - A phrase: "in the event this happens"

     This is a verbose phrase to mean *if*.

   - CLI or API commands and pathnames. Our checker tool must ignore code, filenames, and pathnames.

     This is an advantage of an XML technical writing tool. We can use elements that set content by type: <code>, <codeblock>, <filename>, <pathname>, and similar. We can then set the checker tool to ignore text in these elements. We add a rule to our style guide to use these elements.

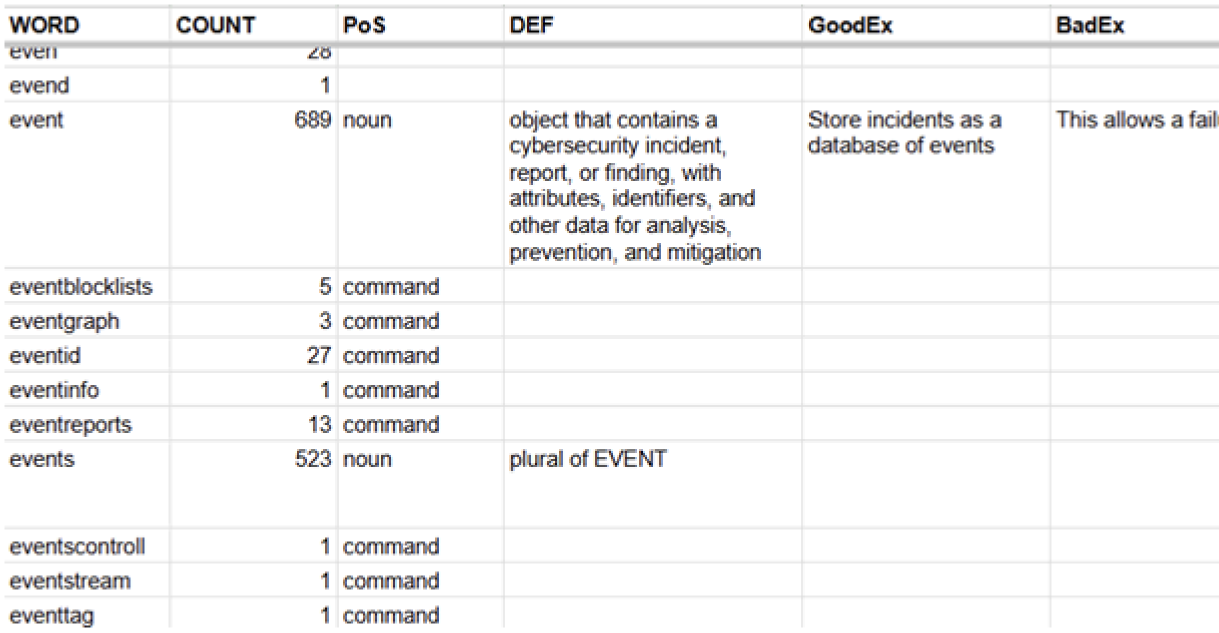
4. Sort the dictionary by Word, to configure all similar words (plural with singular, commands, and so on).

5. In *event*, we enter the definition, good example, and bad example. In *events*, enter: `plural of event`.

6. A non-AI checker tool (and even some AI tools) cannot see the difference of meanings in the use of *event* with our specific definition or the use of the non-allowed defintion. In Rule, enter: `Do not use to mean action of a user or server`.

   Our checker tool will show rules to help writers be consistent.

7. For the words that start with *event* and are obviously commands or pathnames, set the PoS to `command` and Allowed to `T`.



WORD	COUNT	PoS	DEF	GoodEx	BadEx
even	28				
evend	1				
event	689	noun	object that contains a cybersecurity incident, report, or finding, with attributes, identifiers, and other data for analysis, prevention, and mitigation	Store incidents as a database of events	This allows a fail
eventblocklists	5	command			
eventgraph	3	command			
eventid	27	command			
eventinfo	1	command			
eventreports	13	command			
events	523	noun	plural of EVENT		
eventscontroll	1	command			
eventstream	1	command			
eventtag	1	command			

8. In the dictionary, add a row: `in the event`.

   Word = in the event

   Count = added

   PoS = phrase

   Good Example = If X happens

   Bad Example = In the event this happens

   Allowed = F

   Audience = all

   Alt1 = IF

9. Make sure that *if* is set to Allowed = *T*. Search for *if* in the dictionary and set its values. If not in the dictionary, add it now.

   Word = if

   Count = added

   PoS = conjuction

Definition = introduces condition

Allowed = T

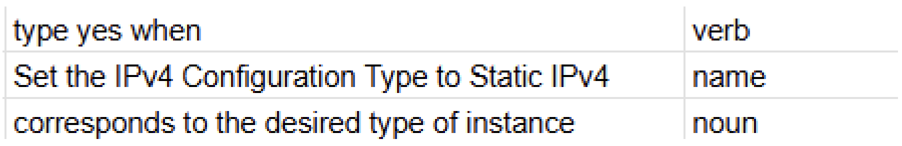
Audience = all

Procedure 7. Analyzing *type*:

This word is an excellent example. It is used in different parts of speech with different definitions in writing and native speaking. To control our written language, we must restrict this word to one PoS and one definition. Or we can decide to set it to not allowed, to be replaced with specific words.

1. Search for *type* in the sample.

2. Skim the hits with context. If it is not clear which PoS is mostly used, enter the PoS of each row. We found that it was most often used as a noun or technical name, but there were sentences with it used as a verb.



type yes when	verb
Set the IPv4 Configuration Type to Static IPv4	name
corresponds to the desired type of instance	noun

3. Add a row for *type* as a verb and set it to not allowed.

   Word = type

   Count = added

   PoS = verb

   Good example = enter yes

   Bad example = type yes

   Allowed = F

   Audience = all

   ALT1 = ENTER

   Rule = do not use as a verb

4. Make sure *enter* is allowed.

   Word = enter

   PoS = verb

   Definition = input values

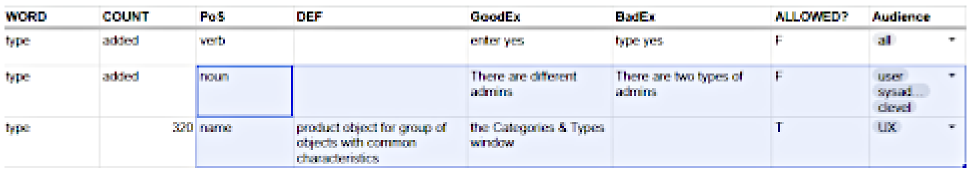
   Good example = enter yes

   Bad example = type yes

   Allowed = T

   Audience = all

5. We see that there many uses of *type* in the GUI and CLI. We could try to make it a technical name for user interface (UX) creators. The word *type* would be allowed in microcopy and coding but not in technical writing.



WORD	COUNT	PoS	DEF	GoodEx	BadEx	ALLOWED?	Audience
type	added	verb		enter yes	type yes	F	all
type	added	noun		There are different admins	There are two types of admins	F	user sysad... clevel
type	320	name	product object for group of objects with common characteristics	the Categories & Types window		T	UX

But we see in the results that *type* is used in text that cannot easily be rewritten. We must allow it for everyone, but only with the required definition, as an object in the product.

Word = type

PoS = name

Definition = product group of objects with common characteristics

Allowed = T

Audience = All

Rule = only as product object - rewrite if meant general

6. Make sure the style guide rule that all text on the interface (GUI, API, or CLI) must be wrapped in an element, such as <code>, <codeblock>, <guilable>. We can then set the checker tool to ignore text in these elements. If your checker tool shows the rules, it will not show this rule for interface labels, where it would cause user fatigue and be ignored when it is necessary.

7. We look through the uses of *type* to mean a general group of people or things having common characteristics. We can rewrite those.

   - Given the text:

     > supports various relationship types, and their usage depends on the entity types being linked

     This one sentence uses *type* with two definitions. The first can be removed. The second fits the allowed defition.

     Controlled version: `supports various relationships, and their usage requires linked entity types`

   - Given the text:

     > there are two types of admins: Org Admins and Site Admins

     The use of *type* is not required. If the SME does not like *level*, we can change it to a different word (*set*, *permissions*). Also note that we remove *two*. It is always best to not enumerate features, to make sure you do not create a conflict in the text when a new feature is added.

     Controlled version: `there are different admin levels: Org Admins and Site Admins`

   - Given the text:

     > the type of storage used by Product can have an impact

     The full text discussed SSD devices and feed caching technology. We guess that "type" meant hardware and configuration.

Controlled version: `The storage you use can have an impact.` OR `your storage hardware and algorithm can impact Product` OR `storage hardware can impact Product`

What Now?

You have a dictionary in progress. You must go through all the words, update the definitions and rules as you go.

When you are done with the top twenty or thirty words, sort your dictionary to see at the words with only one count. These words will be easy to set as not allowed, commands, or misspellings.

- If the word is not allowed, set its values. Make sure the alternative words are allowed.
- If the word is a command, in PoS, enter `command`. Set it to Allowed = `T`. Make sure your style guide says to wrap commands in relevant elements.
- If the word is a mistake, in PoS, enter `misspelling`. When your content is fixed for this mistake, you can remove it from the dictionary.
- If the word is correct and allowed, investigate. If it is a word for a specific audience, set it to Allowed for that audience. If a product owner or sponsor wants it for everyone, discuss why.

  For example, *absent* is used one time, in the phrase *absent a route*. This is a specific network configuration action. It is correct. We allow it for the user, sysadmin, and internal audiences. But we do not want it in the UX microcopy or C-level marketing.

You will need a checker tool. You can acquire the tool before you complete the dictionary. You can ask a Python developer to make a script that returns an email or a webpage with results (unallowed words used, rules as reminders on allowed words, and unknown words). Or you can acquire an AI tool that integrates with your source CCMS. The important thing is that your dictionary is used. It is a dynamic tool for all content creators in your organization.