MARCH 18, 2022

SAMPLE API DOCUMENTATON PLAN

ANNA PARKER
SENIOR TECHNICAL WRITER
annaparkerv2@gmail.com

Anna Parker, Senior Technical Writer
Friday, March 18, 2022

API Documentation Plan

Recently, during some instructional design professional development, I discovered that I am a *computational thinker*. What that means is that I solve problems through decomposition, pattern recognition, and algorithm design. A friend of mine calls me *Lady MacGyver* because I can make things happen with what is available to me, and I thrive in intense problem-solving situations.

As a computational thinker, my overall approach is tinkering, creating, debugging, persevering, and collaborating with others. I have not documented APIs before; however, I am an outstanding technical writer, instructional designer, and people love working with me.

Unique value proposition: I believe that when I transfer my cocktail of expert-level competencies to API documentation and training—specifically in the retail space—I will come up with insights others will not have considered and introduce innovative ways to improve the customer experience.

I hope you enjoy my work!

– Anna Parker

Contents

API documentation planning process

For this section, I have opted to follow Information Mapping formatting standards. This design is ideal for reflecting the planning process requirements and presenting them in a format the hiring committee can skim quickly.

Analysis

Due diligence before my first day.

Define who, what, where, when, why

You will find answers to the 5W questions in the Widget API documentation plan.

Who: Who's the audience for my documentation?
Who performs the various actions described by the writing? Who do users go to for extra customer support?

What: What are the users attempting to do?
What problem do they need to solve? What steps should they take to meet this goal? (How to meet user needs.)

Where: Where do users find the tools and information they expect?
Where do users go for additional info? Specific Commands? Best practices? Samples?

When: What is the sequence of tasks the users need to do?
What is the order of operations? Are there time constraints? When/in what situations will users run into problems?

Why: Why does the user do each task and sub-task?
Why perform the function in the first place? Why didn't we include certain information? Why did we design it this way in the first place? (Rationale)

Research current practices

1. What are company's REST API documentation standards and practices?
   Many companies publicly post their API documentation—link to company URL
2. What are the industry best practices for REST API documentation?
   https://idratherbewriting.com/learnapidoc/
3. What are the standards for API documentation?
   https://developers.google.com/style
4. What are current examples of the best API documentation out there and why? (This blog is from 2019, but it aligns with Tom Johnson's more current lessons on good examples of APIs). Twilio | GitHub
5. What is the checklist of must-haves in API documentation?

   –Resource description
   –Endpoints and methods
   –Parameters
   –Request sample
   –Response example and schema

6. What are the ideal deliverables?

   –Authentication guide
   –Quickstart guide
   –Endpoint definitions
   –Code snippets
   –Example responses

Approach

Learn the lay of the land

When I initially come on board, these are the questions I will have. Most of them are covered by standard first-day orientation.

Who

Who am I reporting to? Who may I shadow for a day or two? Who do I go to for support for different things? Who do I work with on different teams? Which Subject Matter Expert (SME) knows everything but is impossible to lock down and has very precious time? Who can I go to for different types of information?

What

What problem am I solving and what steps do I take? How do you document your specifications and which ones do I follow? In what way do you follow the Agile project management model? What is your product management model, how do you log tickets, and how do you prioritize the backlog? What do sprints look like?

Where

Where do I find the technology, tools, and resources I need and what are they? I know you work in a docs-as-code environment, but what is your source control system? Do you use a tool like Javadoc to collect comments? Do you use GitHub and branches? What is your publishing/deployment environment? What is my testing environment?

When

What are the company's Standard Operating Procedures (SOPs) for working with docs as code? What sequence of events must I follow? What's my workflow? What are my milestones? What's my schedule/deadline? How much time can I have to learn new tools and fine-tune my coding skills?

Why

Why am I doing things the way I am? What's my rationale? What matters most—quality, efficiency, or cost-effectiveness? What are the mission, vision, and values of company, and how closely do you follow them in your decision-making? What is the corporate culture?

Build rapport through meet and greet

Emotional intelligence (EI) and understanding peer values are key when it comes to collaborating with different teams and understanding stakeholder expectations and desired results.

When it comes to working with engineers, complex concepts, tight deadlines, and shifting priorities—as is common in a startup company environment—EI begets a competitive edge, enjoyable collaborations, and a desire to set peers up for success.

For more information, see Working with engineers/.

Plan

Widget API documentation plan

The engineering team is working on a new REST API that partners can use to add data to a custom dashboard.

The partners will be creating components to display the data using whatever technology they use.

Use case

The partner's want to be able to answer questions about the data.

Audience

Primary: *Software engineers* who will be designing and coding data.

Secondary: *Planners and strategists*—typically management—who need to understand what the API does to make executive decisions.

Deliverables

Company's documentation team already divides their deliverables into *x* guides as you can see at *company_url.*

Using *Existing API* as my model, I discern that the *x Guide* focuses on conceptual information that is ideal for planners and strategists, while the *y Guide* features functional use cases and technical details that software engineers require to design and code their software solutions.

The Widget API deliverables will be:

- X Guide
- Y Guide

Upon further inspection of the model API, I see that the writers cover: resource description, endpoints and methods, parameters, request sample, response example and schema, which is the standard according to API documentation expert Tom Johnson, Senior Technical Writer at Google and author of Documenting APIs: A guide for technical writers and engineers.

X Guide

The X Guide has the following sections:

Overview: Provides a high-level overview and visual of what the API offers

Changelog: Standard readme file that documents updates

Concepts: Business description for each of the Dashboard API endpoints

- Endpoint Alpha
- Endpoint Beta
- Endpoint Gamma
- Endpoint Delta
- Endpoint Epsilon

Plan, *continued*

(X Guide, continued)

Tutorials: Data retrieval walkthrough with sample code that illustrates the API capabilities

Overview [draft]

If your team decides to build and host your own *xxx* Dashboard, use the Widget API endpoints to get the *xxx* data to display real-time *xxx* information about *xxx* and give partners the power to manually make changes.

Use this simple dashboard interface to visualize how the status data can be displayed.

<!--- Insert screen capture of simple dashboard interface that displays status information --->

You can display:

- A
- B
- C

Y Guide

The Y Guide has the following sections:

Endpoints overview: Provides an overview of the endpoints with a description in table format.

Content for table [draft]

The API includes the following endpoints:

`GET /alpha`
Gets all specific alpha data.

`GET /beta`
Gets all specific beta data.

`GET /gamma`
Gets all specific gamma data.

`GET /delta`
Gets all specific delta data.

`POST /epsilon`
Gets all specific epsilon data.

Plan, *continued*

(Y Guide, continued)

Endpoint technical details

The remaining endpoint sections feature the same kind of technical information you see in the *model* I'm following:

- Endpoint and definition
- Valid types
- Workflow states
- Security
- Parameters
- Request examples
- Responses table
- Response examples
- Errors

Alpha context: Detailed technical information for the `GET /alpha` endpoint.

Beta context: Detailed technical information for the `GET /beta` endpoint.

Gamma context: Detailed technical information for the `GET /gamma` endpoint.

Delta context: Detailed technical information for the `GET /delta` endpoint.

Epsilon context: Detailed technical information for the `POST /epsilon` endpoint.

Common API reference content

The following technical content is common across all the *company* APIs.

- APIs
- Localization
- Error and status codes
- Authentication

Deliverable format

I understand that the *company* team works in Markdown with Z. I would follow whatever setup you have right now.

Scheduling

I assume you have a project manager who tracks all the milestones and ensures that I meet my deadlines. I would want to meet with the project manager early on to find out how long I am expected to take to complete my tasks. I would also learn the culture and expectations around booking time with SMEs.

Design

I'm not interested in reinventing the wheel. This is already done. Over time I look forward to contributing to improvements as we get feedback from our users.

Develop

Writing!

The name of the game is **single sourcing**. If I can avoid writing anything from scratch, I will.

Start with a boilerplate and go from there

1. Duplicate the existing model documentation and start modifying as much of it as I can based the information I have.
2. Document a simple framework/table that lays out all the deliverables, sections, and subsections and identifies my information gaps and where I think I can get the information. See Appendix A – Widget API deliverables at a glance.
3. Book time with a fellow tech writer to find out how they gather their missing information from their developers and learn what works well and what their obstacles are.
4. With that knowledge in my head, I would book my first call with the engineer SME that is only a meet and greet. It's an important one though. Objective is to introduce ourselves, build rapport, and discover preferred method of communication. What is their favourite part of documenting APIs and their most dreaded parts? How do they want the process to flow? Any opportunities for single sourcing? Any information they have already provided elsewhere? Perhaps overview content is already written in marketing materials? It's an opportunity to develop a standard approach and improve on what is the current process. What would be most efficient and enjoyable for all of us? Based on the information I gather in that first brief meeting, I will devise my approach for the information gathering session that comes next.
5. Next step is to create the meeting structure, agenda, and list of questions for information gathering session with SME. I would send it and ask for feedback and preferences, so they can prepare for the meeting in their preferred way. It is important that I have that meeting online so that I can record it. I love using the transcription functionality specifically for conceptual content. Oftentimes I can take what the SME says verbatim and edit that content.

Working with engineers/SMEs

Here are some specifics around my first meetings with engineers/SMEs.

First Meeting

- 30 minutes or 15-minute standing meeting, it depends on the culture
- Introduce myself and background to break the ice and establish credibility
- Have them introduce themselves in the same way
- Questions:
  –How long have you been with *company*?
  –Have you been involved with documenting any of the other APIs?
  –What was your favorite part?
  –What did you learn?
  –What is your preferred method for communication? (chat, email, video call)
  –What is your schedule?
  –Do you have any advice for me?

Develop, *continued*

(Working with engineers/SMEs, continued)

Second Meeting

Send email to SME with the following information:

- Go to source control system and identify which existing API documentation is most similar to ours.
- Do a quick review of the content there.
- Be prepared to walk me through an informal demo of a tutorial that will be recorded--use one of the existing tutorials as a model.

During the meeting:

- Clarify the parameters of what I'm looking for in the tutorial walk-through and hand the meeting over to the SME.

Following the meeting:

- The recorded tutorial walk-through will provide me with enough material to draft my first pass on the X and Y Guides.
- I will work on the content as far as I can go on my own.
- I will expand the API deliverables at a glance table and book a meeting with a tech writer peer to determine how to address my gaps without bothering the SME, and then work through those updates.

Third Meeting/Communication

Send an email to the SME with a link to the spreadsheet and identify what I still need.

- Depending on their preferred method of communication, I may book a meeting to go over our next steps. If they don't like meetings, then I won't.
- Collaborate with SME to gather remaining content until my final draft is ready for review.
- Book a review meeting with a tech writer peer and go through the content. Make changes.
- Send final draft to SME in advance and set a review meeting time that serves as their deadline.

Review

Editing techniques, tools, and guides

I use the following techniques, tools, and guides when I do formal editing work.

Techniques

I do multiple passes in MS Word and make changes to the document with the *Track Changes* feature turned on:

1. Substantive edit to address organization, structure, flow of content, and content design
2. Copy-edit without tools to address voice, word choice, and clarity
3. Copy-edit with tools to check for spelling, punctuation, typos, consistency, and accuracy
4. Final proofread line-by-line out loud for final polish

Review, *continued*

(Editing techniques, tools, and guides, continued)

Tools

For regular writing in MS Word, I have the free *Grammarly* plugin that catches common spelling and grammar mistakes.

I also use the *PerfectIt 5* MS Word plugin to check consistency and enforce style rules that I can customize to align with corporate standards.

Guides

For any professional writing that I do, my first step is to procure the corporate style guide.

If there isn't one, or if I'm looking for specifics that aren't documented, I will go to the Microsoft Style Guide (https://docs.microsoft.com/en-us/style-guide/welcome/).

For API documentation, I follow the best practices outlined in the Google developer documentation style guide (https://developers.google.com/style).

Publish

Follow the technical writing team's standard publishing and deployment procedures.

Appendix A – Widget API deliverables at a glance

This table provides an overview of the sections of content, status, owner, and comments. It is intended to serve as the framework for the API documentation, a collaboration tracking sheet for the writer and SME, and a living status report for project management purposes.

As the project evolves, more rows are added to the Sections column as I encounter the chunks of content.

Guide	*Section*	*Status*	*Comments*
X Guide	Overview	Started	High level content provided in spec
	Changelog	Done	This is the initial release—using boilerplate content
	Concepts	TBD	
	Tutorials	TBD	
Y Guide	Endpoints Overview	Started	Specs provided baseline definitions, but writer will enrich near completion
	GET /alpha	TBD	
	GET /beta	TBD	
	GET /gamma	TBD	
	GET /delta	TBD	
	POST /epsilon	TBD	