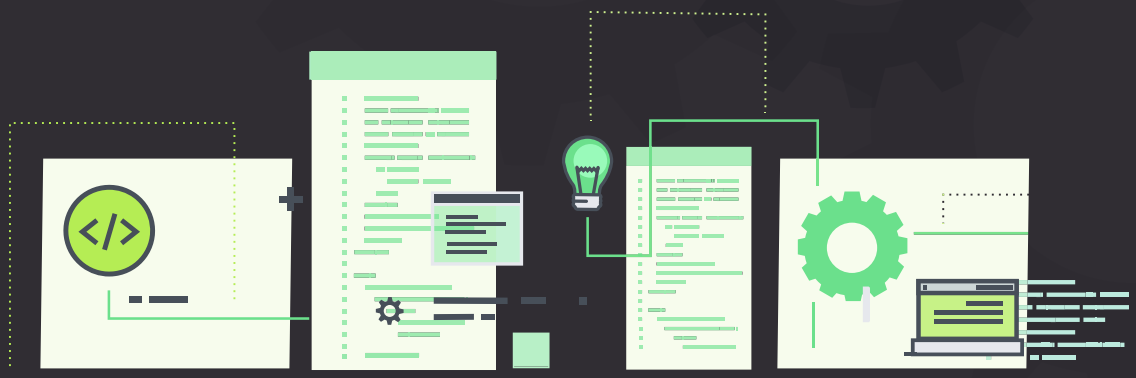


# API DOCUMENTATION

## THE SECRET TO A GREAT API DEVELOPER EXPERIENCE

{ ... } SWAGGERhub

# The Secret to a Great API Developer Experience

Today, organizations across all industries are recognizing the business and strategic opportunities of investing in an API program. APIs enable digital transformation, and open the doors to a host of engineering and business possibilities.

But the growth of the API economy also requires a new way of thinking for teams that are building APIs. Even the best API programs will fail if end consumers don't understand the value of working with the API, and don't have the necessary resources available to start using it. It is not enough to simply have an API; you also need to have a great API Developer Experience.

## What is API developer experience?

API Developer experience is an extension of general User Experience (UX). It is the aggregate of all the experiences a developer has when interacting with your API. A good API developer experience goes beyond technical writing. It is about providing all the right resources to help your end consumers understand, and successfully work with your API.

## Why does API developer experience matter?

In today's modern software world, the average technical consumer has an enormous amount of buying influence when it comes to the tools and platforms that organizations decide to implement. According to Jeffrey Hammond, principal analyst at Forrester, adoption patterns within software teams are shifting towards developers, giving them the power to hinder or aid the adoption of solutions. This means that technical adopters of software products are now the decision makers for adopting and buying products. Making it easy for your products to be consumed by the technical adopter is thus crucial in today's hyperconnected and competitive ecosystems, especially in the API economy.

## The secret to a great API developer experience

When it comes to providing a great developer experience, there is no substitute for a high performing, easy-to-use API. Developer experience will always start with providing a reliable API that teams want to work with, and can trust to securely integrate with.

A critical component to providing a great developer experience is providing accurate and up-to-date API documentation. API documentation is the information that is required to successfully consume and integrate with an API. This could be in the form of technical writing, code samples, and examples for better understanding how to consume an API.

Today, some of the most well-known and widely adopted APIs are investing in rich, human friendly documentation for their APIs. Companies like Facebook, YouTube, Microsoft, PayPal, and DropBox — which use internal and public APIs to drive technical orchestration and strategic growth — are putting documentation at the center of their API developer experience.

### How can you start your API documentation journey?

Providing a great developer experience, with documentation at the center of it, has never been more accessible to API teams. While in the past, teams had to rely on static forms of documentation — like PDFs or manually-updated webpages — there are now solutions to automate your documentation workflow, and build out interactive API docs that make consumption of APIs a smooth and easy process.

There have been significant changes in the ways organizations document their APIs. Nowhere are these changes more evident than in the widespread adoption of API description formats like the OpenAPI/Swagger Specification, which provide the building blocks for generating beautiful interactive API documentation that end consumers can interact with without having implementing it into their code base. This auto-generated documentation is a central resource that your development team can customize, and build on to create a more comprehensive user manual for working with your API.

In this eBook we will look at the factors that go into providing a great developer experience, and how documentation fits in. We will introduce best practices for API documentation, and will look at how teams can start documenting their APIs with Swagger and improve their existing documentation workflow in SwaggerHub.

Let's get started!

# Contents

# Planning Your API Developer Experience

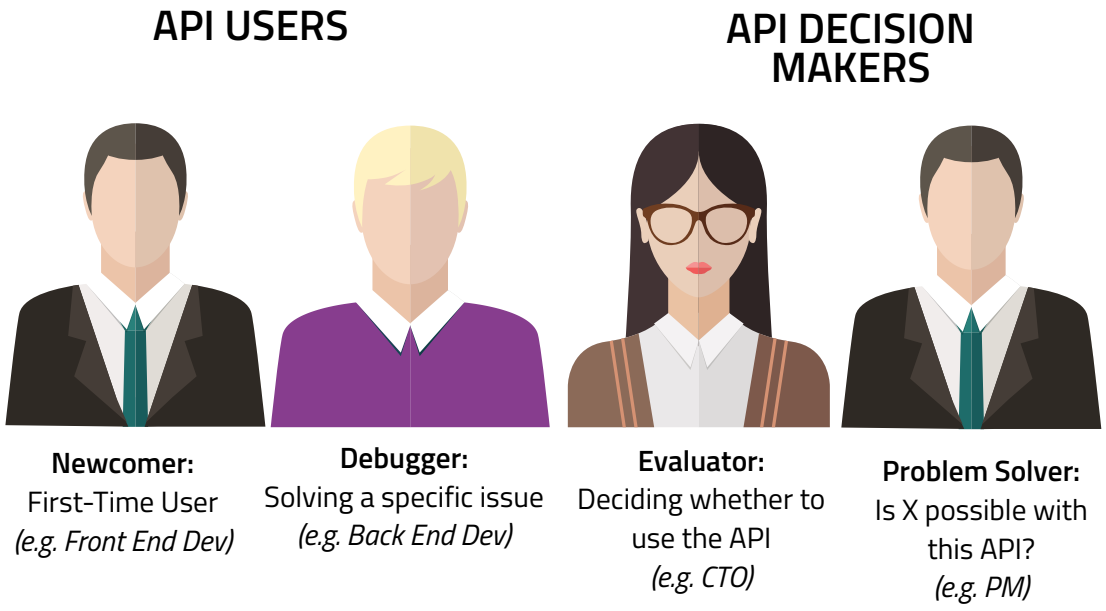
Empathy is the guiding force for good developer experience. A successful API needs to be treated with the same care and attention as any first-class product in the company's portfolio. Applying product management principles when dealing with APIs is the first step to treating your APIs as first class products. In order to build an API that is optimized for a positive experience for your end consumers, we need to have a good strategy. This starts with understanding a few crucial details about how people discover, and work with your API:

1. Understanding the API's target audience
2. Understanding the API adoption journey
3. Mapping the journey to the right audience type

## Understanding the API's target audience

As we stressed before, documentation is the usage manual for your API. As with any product — and yes APIs are products — you need to start by understanding who needs to consume your API's documentation.

In the graphic below, you will see an overview of the potential audiences for your API's documentation:




The most common consumers for your API's content and documentation are the people who need to directly integrate and work with your API, as well as the people who want to evaluate and see if this API will fit into their development and business strategy.

These can be categorized into two groups:

- **API users:** The developers who want to integrate with your services and are looking to resolve a specific issue using your API.
- **API decision makers:** The people who will be evaluating your services and seeing if it solves a strategic need.

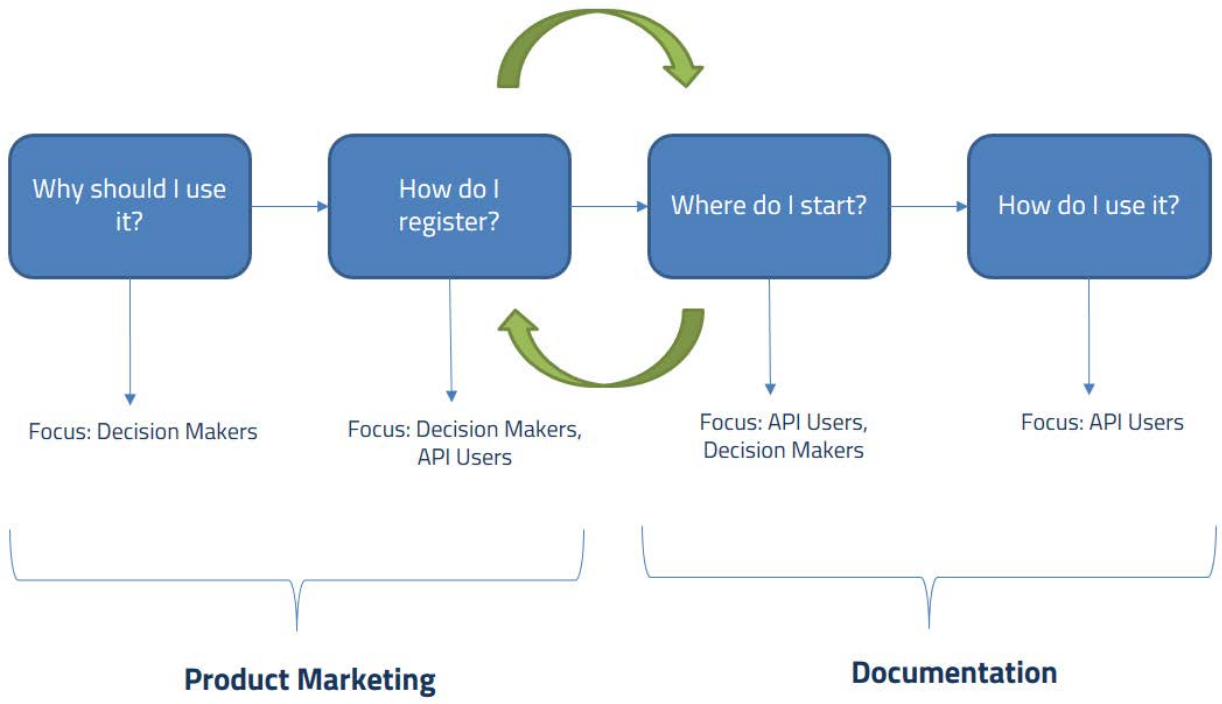
When you start to break down the audiences for your API, it's easier to understand what type of resources you need to provide and how these resources should be presented.

## Understanding the API adoption journey

Your audience will now go through different stages of discovering and evaluating your API before fully committing to your services. This user flow takes your target audience from awareness to finally becoming familiar and comfortable enough to use your API services. A well-designed API will cater to the optimal developer experience across every stage of this journey.

The graphic below provides an overview of the key stages users will go through before adopting your API. These stages can be simplified into four main questions:

1. Why should I use it?
2. How do I register to use it?
3. Where do I start?
4. How do I use it?



Keep in mind that audiences can overlap, but breaking down the user journey allows you to establish a starting point to understand how you need to communicate with the different people involved in adopting your API.

These phases also tap into different roles within your organization. For example, the first two stages are more of a product marketing exercise, concentrating on communicating the value of your services and how they alleviate existing prospect challenges. Of course, the next two stages are purely a technical writing exercise wherein the technical writers need to be producing content to help your end consumers to understand and integrate with your services.

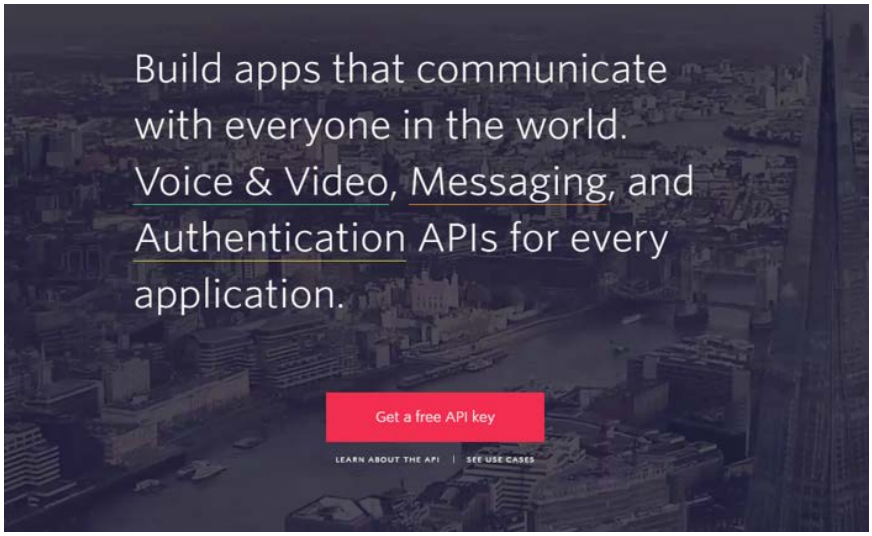
Let's take a deep dive at understanding these different stages:

## "Why should I use your API?"

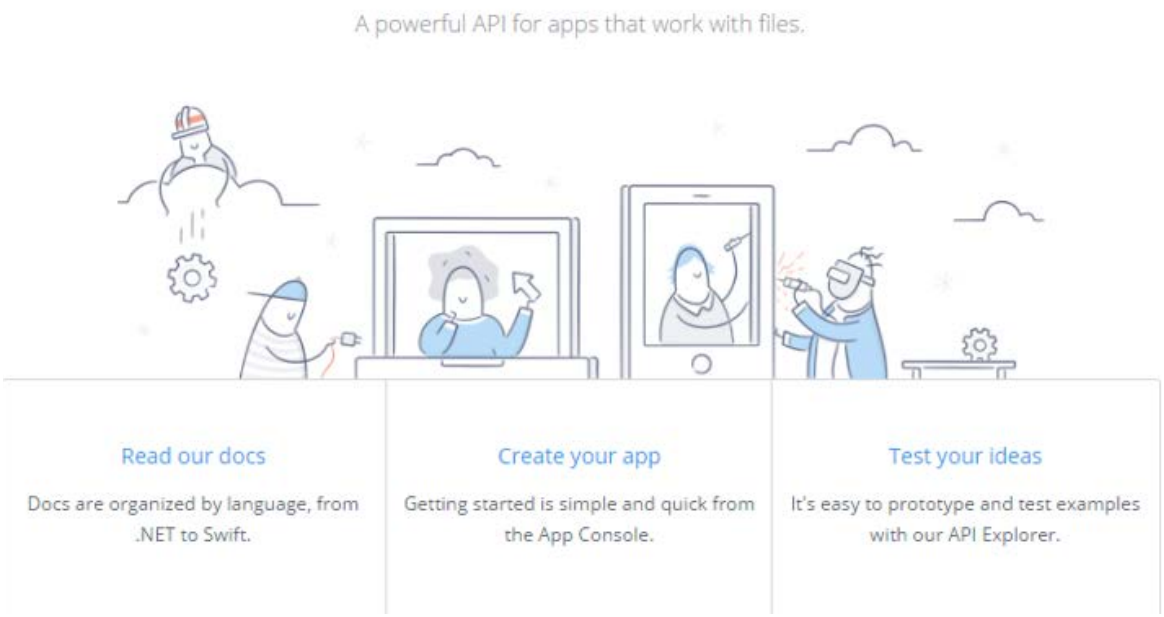
This phase aims to provide your audience an overview of your services in a way that immediately resonates with the task they wish to accomplish. API users and evaluators should quickly understand the value your API is offering and how they can start using it. This is where the target audience interacts with the information presented for the first time. In order to present this information in a simple and effective manner, you need to understand why these prospects are coming to your API, how they discover your services, and what are the existing problems that your services can help solve.

Two examples of companies that do this well are:

**Twilio:** Twilio allows users to build apps that communicate with voice, video, and messaging through their APIs. The Twillio overview page tells you exactly what they're supposed to do, and then provides a more detailed overview as you move down the page.



**DropBox.** Dropbox is a cloud storage and sharing solution of individuals and organizations. Dropbox's API page makes it easy to understand how developers could use their API to build apps, and provides easy access to resources to get started.




## "How do I register?

Once you've convinced your API users and decision makers that your services could be of value to them, they will move forward in their journey. The registration process is the first step to getting into your API services. This process should be as straightforward as possible, but you should also be able to obtain the information required for you to effectively manage these users. One strategy is to allow developers to sign up using popular third party services, like Twitter or GitHub. With these services, it's easy to quickly capture contact information without making them jump through too many hurdles.

If you do have to include extra steps for people to register to access your API, be sure to provide comprehensive documentation and help tips to accelerate the process. One example of a bad developer experience is to force users to fill out a lengthy form, or requiring unnecessary details before providing an API key.

## "Where do I start?" and "How do I use it?"

Now that the target audience has registered to access your API, the next step is to get them started. These are the last stages of your developer journey towards consuming your API services. Now they want to get their hands dirty with your API code, trying it out and eventually integrating it with their own apps.

How do you guide them through this journey? The secret is great API documentation.

# Documentation is the Center of Good Developer Experience

Concise and clear documentation — which allows your API consumers to adopt it into their application quickly — is no longer optional for organizations that want to drive adoption of their APIs.

Consider the following:

- According to SmartBear's 2016 State of API Report, 75% of organizations that develop APIs now have a formal documentation process. 46% say it is a high priority for their organization.
- A survey by ProgrammableWeb found that API consumers considered complete and accurate documentation as the biggest factor impacting their API decision making, even outweighing price and API performance.

Good documentation accelerates development and consumption, and reduces the money and time that would otherwise be spent answering support calls. It's important to remember that documentation matters for internal API users as well. Don't assume that your internal stakeholders will have intimate knowledge of how to work with the API.

What should go into your API documentation? Here are a few tips that may help you when documenting your API.

## Tip #1: List the Fundamentals

There are certain sections which every good API documentation must have. Without these sections, consumers will have a hard time understanding how your API can be used.

**Authentication.** This is the information about authentication schemes your users need to start consuming your API. If you're using OAuth for example, don't forget to explain how to set up an OAuth application and securely obtain an API key.

**Error messages.** Error messages are important because you want to get feedback when you're integrating with your API services in an incorrect way. Explain your error standards, and also provide solutions on how to overcome them when an end consumer gets an error.

**End points and information on how to consume them.** Pay attention to describing your request and response cycles. These are the main components of your API where users will be interacting with your services, so pay close attention to this.

**Terms of use.** This is the legal agreement between the consumer and your organization, defining how the consumer should ideally use your services. This is important because you want to see to it that developers and consumers comply with your organization's recommended practices, and not do anything that goes against your business values. Include API limits under best practices, with terms and conditions. Constraints also need to be clearly stated so that consumers understand what APIs usage and practices are permitted.

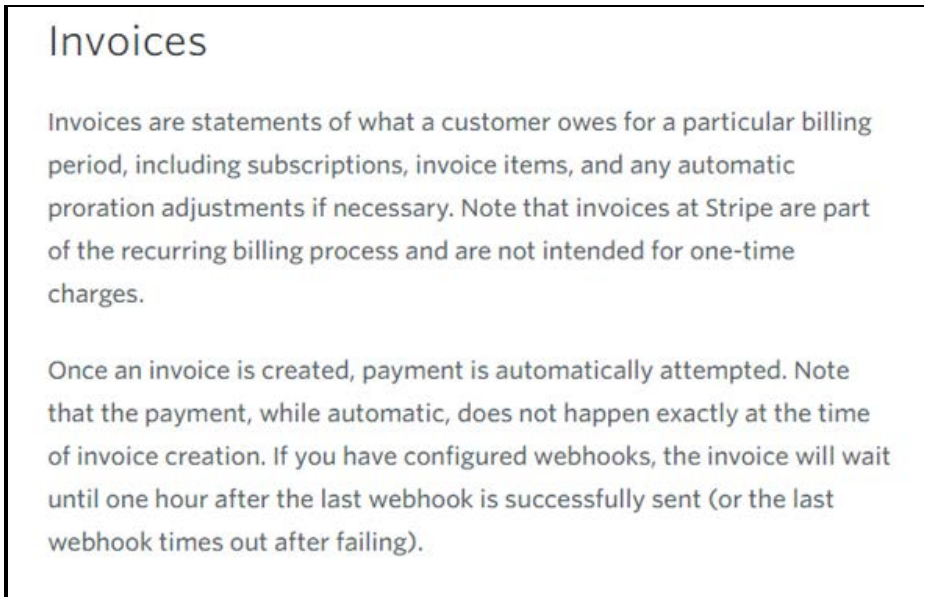
**Change log.** Detail updates and versions of your APIs and how that might affect API consumers. This will help consumers know the stability of your API and see if any changes need to be made for an effective API call. Once you've listed these sections, it's easy to go about documenting them, because now your API is already off to a great start.

## Tip #2: Write for Humans

Most API documentation writers assume that the audience for their documentation is 100% developers, or people that will have full technical understand of how to work with the API. But keep in mind that many people working with the API may not have intimate knowledge of the domain or jargon you may be using. Documentation should cater to the API users, typically developers, and the relatively less technical API evaluators, typically Product Managers and CTOs.

Start your documentation by writing English domain explanations for each call. Avoid using a lot of technical jargon, and write in a way that can be easily understood by people who are new to the API. If you do have technical or domain specific jargon, link that specific item to further documentation explaining these terms. These tactics will ensure clarity and good structure across your API and why certain calls exist, before you ever get lost in the details of the parameters, headers, and responses.

Consider this example from Stripe's API documentation. They do an excellent job of supporting any technical jargon they have with additional pieces of content that explain them.



Invoices

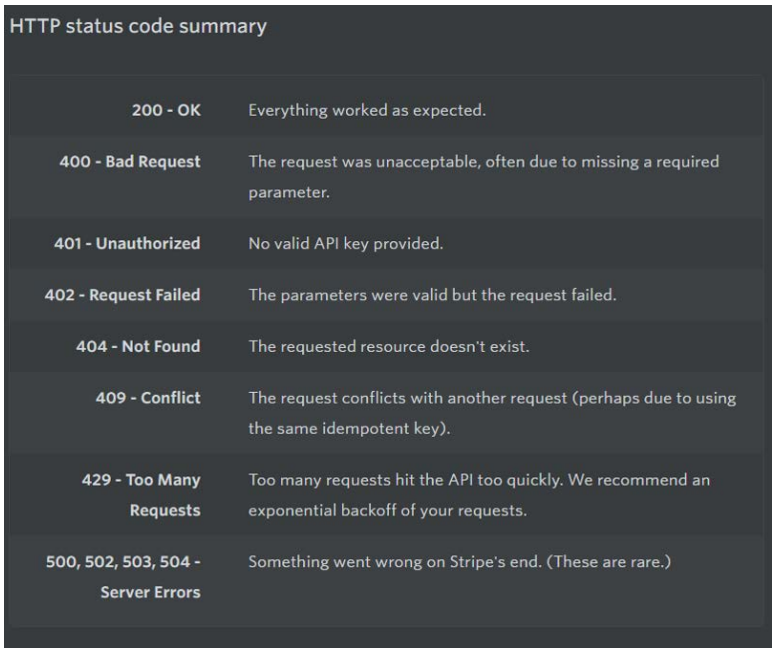
Invoices are statements of what a customer owes for a particular billing period, including subscriptions, invoice items, and any automatic proration adjustments if necessary. Note that invoices at Stripe are part of the recurring billing process and are not intended for one-time charges.

Once an invoice is created, payment is automatically attempted. Note that the payment, while automatic, does not happen exactly at the time of invoice creation. If you have configured webhooks, the invoice will wait until one hour after the last webhook is successfully sent (or the last webhook times out after failing).

## Tip #3: Explain Your Request-Response Cycles

Your API users should know exactly what to expect from a successful API call. Describe the full sample response body in every supported format. Think of not only the format, like XML or JSON, but also of HTTP headers, error codes, and messages. Having too much information available for the end consumer won't be an issue, especially when they're trying to integrate your services into their applications.

Provide examples in every single object which your API is supposed to return, as well as examples of parameters that users can add for a successful API call. Here's another example of Stripe. End users can easily understand exactly what each error code means in the start of the documentation.



HTTP status code summary

| | |
|---|---|
| **200 - OK** | Everything worked as expected. |
| **400 - Bad Request** | The request was unacceptable, often due to missing a required parameter. |
| **401 - Unauthorized** | No valid API key provided. |
| **402 - Request Failed** | The parameters were valid but the request failed. |
| **404 - Not Found** | The requested resource doesn't exist. |
| **409 - Conflict** | The request conflicts with another request (perhaps due to using the same idempotent key). |
| **429 - Too Many Requests** | Too many requests hit the API too quickly. We recommend an exponential backoff of your requests. |
| **500, 502, 503, 504 - Server Errors** | Something went wrong on Stripe's end. (These are rare.) |

The YouTube API documentation also describes responses and parameters well, with helpful examples for each of them.

| Parameters | |
|---|---|
| `autoplay` | This parameter specifies whether the initial video will automatically start to play when the player loads. Supported values are `0` or `1`. The default value is `0`. |
| `cc_load_policy` | Setting the parameter's value to `1` causes closed captions to be shown by default, even if the user has turned captions off. The default behavior is based on user preference. |
| `color` | This parameter specifies the color that will be used in the player's video progress bar to highlight the amount of the video that the viewer has already seen. Valid parameter values are `red` and `white`, and, by default, the player uses the color red in the video progress bar. See the YouTube API blog for more information about color options.<br><br>**Note:** Setting the `color` parameter to `white` will disable the `modestbranding` option. |

## Tip #4: Experimentation is Power

Beyond basic documentation, there are additional resources that teams should provide to build on the experience they are offering to end consumers.

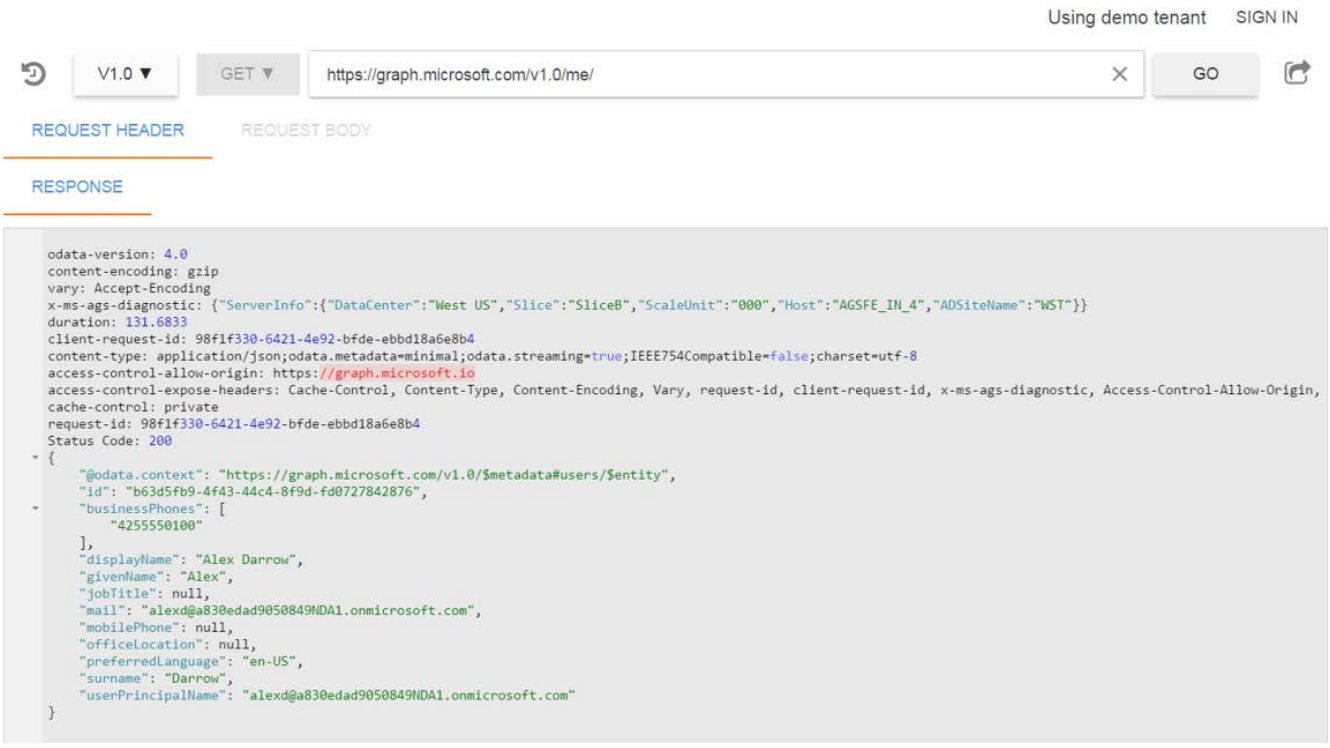
- **Getting started guide:** A "getting started" guide can be a helpful next step resource for developers that have decided to implement your API. This resource should provide a more detailed walkthrough on quickly consuming the API.

- **Interactive docs and console:** Your audience will want to try your API before directly integrating it into their application. Provide a sandbox, or a console, to allow developers to easily deploy and reset responses for different end points, without the need to tamper with source code.

- **SDKs:** Once your API is out, it's a good idea to invest in building client libraries that allow end users to effectively consume your API. If a lot of developers are finding value in your services, they may even build SDKs for you. Offering a good developer experience is a great way to nurture your developer community.

- **Tutorials:** Provide sample snippets, sample SDKs and good use cases for helping users understand your services.

Microsoft Graph API offers an API explorer, which allows users to try various end points and see what the response packet is.



# API Documentation with OpenAPI (Swagger Specification)

Today, thousands of API teams around the world use the OpenAPI Specification and Swagger tooling to generate documentation for their internal and public-facing APIs.

## OpenAPI/Swagger Specification Overview

The OpenAPI/Swagger specification is a machine and human readable description format that defines the API's contact. This contract defines what data is exposed by the resources and services, and how the client should call these resources. While this idea seems simple, it has powerful implications in the multi-platform API economy where services are built in many languages, and consumed by clients across different devices.

The specification was renamed as the OpenAPI Specification to standardize the way REST APIs are designed and documented. We will be using Swagger and OpenAPI interchangeably in this ebook. There are many practical uses for implementing the OAS into your API workflow, the most common application of OAS for API teams is to generate beautiful interactive documentation that can easily be shared with the API's end user.

The OAS addresses two of the biggest challenges team face when handing API documentation:

### Maintenance of API Docs

Maintaining and updating this documentation for your development team and end consumers, so they work with the API efficiently, becomes a difficult process. This is especially true if you're using static documents, like a .pdf, to provide documentation to your end consumers. OAS lets you define a contract for your API, and auto-generate an interactive UI for that API using a tool like Swagger UI or SwaggerHub.

That interactive UI can act as the canvas for your team to build out your API documentation with the different examples, descriptions, error messages, and other details that go into great API documentation.

### Multiple Service Interaction

APIs needed a common interface for consumption and interaction between different services. Traditional API interfaces, be it text documentation, or others like Javadocs, do not allow for interactions between services in different languages.

Web services defined with OAS can communicate with each other irrespective of the language they're built in, since it is language agnostic and machine readable.

## Documenting Your API with the OAS

Documenting your API with the OAS starts with generating an initial OAS contract, or "spec," which forms the foundation for your API's design and documentation. Below is an example of what that contract looks like:

```
swagger: '2.0'
info:
  version: '1.0.0'
  title: 'The Meetup API'
  description: |
    The Meetup API provides developers a safe and secure way to access data from
https://meetups.com, with the ability to consume and manipulate the data presented
to them. The Meetup API allows third party devs  to use this information in an easy
and effective way.
    Once you've [registered your client](https://secure.meetup.com/meetup_api/key/),
it's easy to start requesting data from the database.
    All end points are accessible via `https` and are located in `api.meetup.com`
    For instance, you can grab all the profiles of Meetup users in boston by access-
ing the following URL with your client ID.
    ```
    https://api.meetup.com/2/profiles?city=boston&client_id=CLIENT-ID
    ```
you can experiment with the API [in our console](https://secure.meetup.com/meetup_
api/console/) at any time.
  termsOfService: https://www.meetup.com/help/article/1744538/
host: api.meetup.com
basePath: /2
schemes:
  - https
produces:
  - application/JSON
consumes:
  - application/JSON

paths:
  /meetups:
    get:
      description: The `GET` operation gives information about [Meetups](https://
www.meetup.com) in a specific city
      parameters:
        - name: city
          in: query
          type: string
          description: the city in which you want the list of the Meetups in
        - name: title
          in: query
          type: string
          required: false
          description: The title of the Meetup you want information about
      responses:
        200:
          description: Successful response
```

There are two approaches to generating Swagger (OAS) for your API:

## DESIGN FIRST

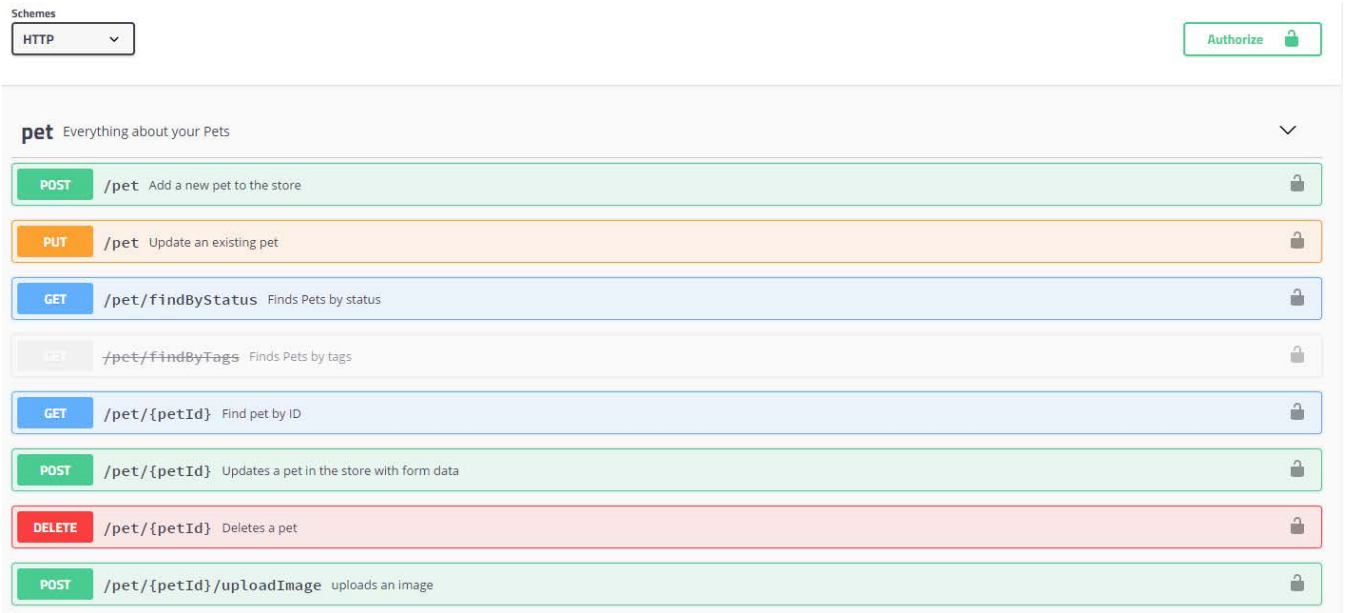
Building the API with the API's contract in YAML or JSON is the Design First approach. This design is then used to generate code and drive the development of the API.

## CODE FIRST

Generating the contract from an existing API's codebase is the Code First approach. This contract is then used to build on the API's documentation.

If you don't have a Swagger definition for your API already, we recommend this tutorial for documenting an existing API with Swagger. Regardless of what approach your team takes to Swagger, there will be work you'll need to do to build out the documentation for your API. In most cases, organizations will rely on technical writers to fill out the documentation. This involves adding meaningful, understandable information that your end consumers can use to achieve API success.

From this contract, an interactive version of the documentation can be generated using the Swagger UI. The Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API's resources without having any of the implementation logic in place.



# Enhanced API Documentation with SwaggerHub

API documentation, as we detailed above, is no easy task. Organizations not only need to work on technical writing, but also make sure the documentation is secure and easy to work with. There are operational considerations like hosting and maintenance, all of which cost organizations additional overhead in terms of money and time.

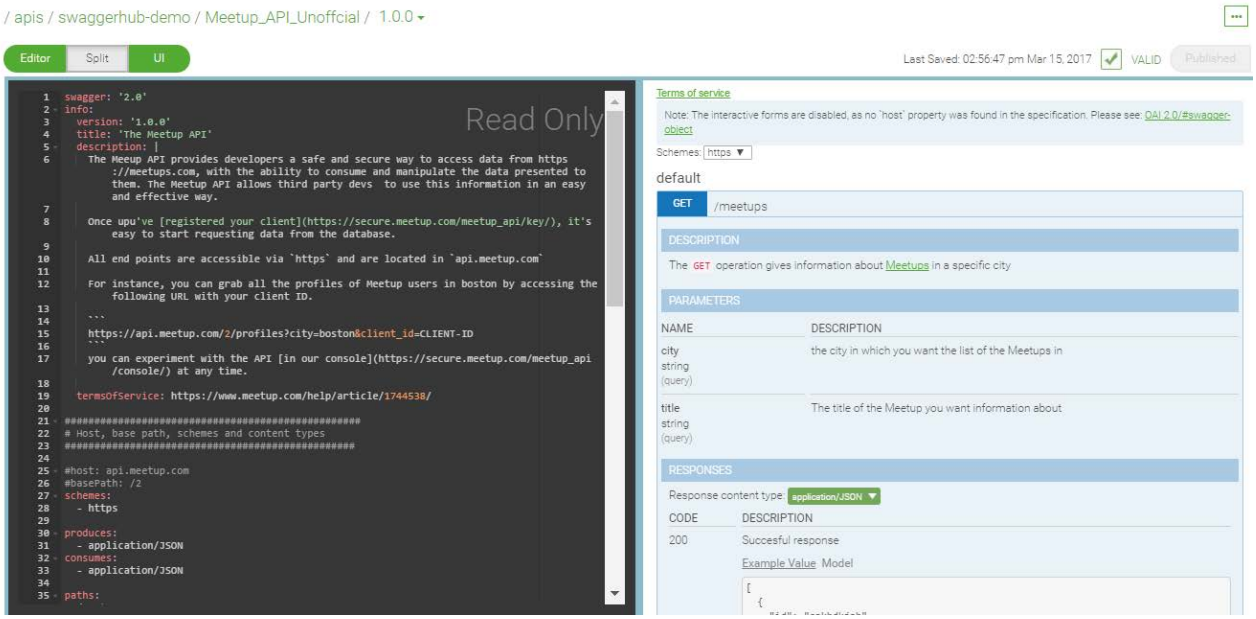
This is why a dedicated organizational platform like SwaggerHub can help. SwaggerHub is an integrated API design and documentation platform, built for teams to drive consistency and discipline across the API development workflow.

SwaggerHub's interactive documentation renders your OAS/Swagger definition visually for live interaction and workability, so your end consumers know exactly how your API will read and behave, before integrating it into their code base. It's generated straight  from the API contract, which can be created from scratch using the SwaggerHub Editor, or imported from your filesystem and source control hosts.

SwaggerHub provides key capabilities for accelerating an organization's API documentation workflow, while saving money and effort in implementation. These features include:

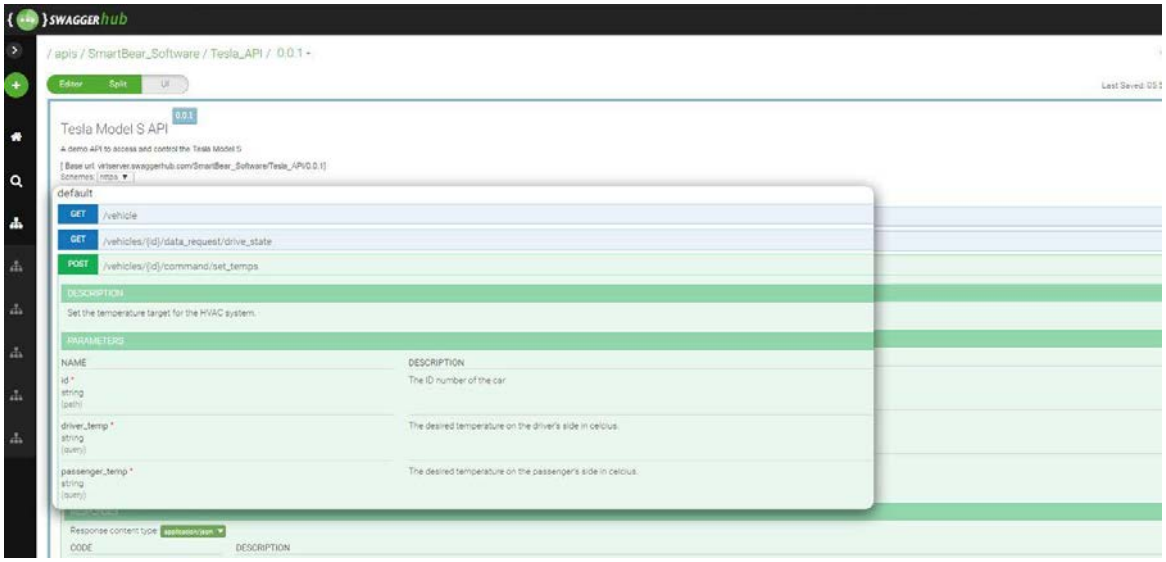
## Real-Time Visual Rendering

SwaggerHub provides a powerful visual editor for defining every endpoint of your API with OAS/Swagger, allowing API architects and developers to understand how consumers would interact with your API before any code is written. SwaggerHub puts documentation as the center of your API workflow, letting you auto-generate the interactive UI for your API that updates with every change to make to the API's contract.



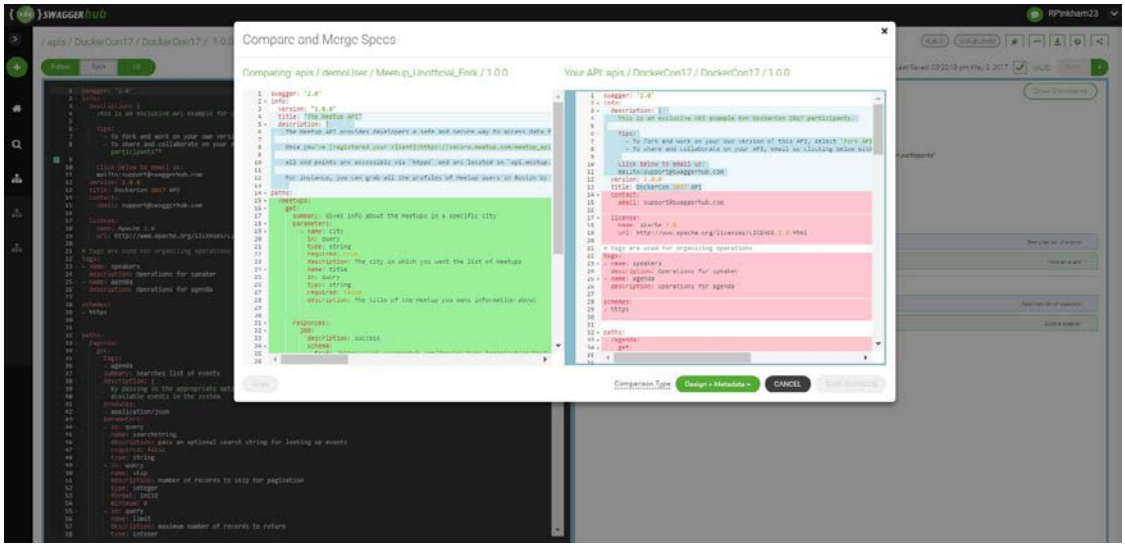
## Secure, Automatic Hosting

SwaggerHub eliminates the need to setup a backend server to host your API's documentation. SwaggerHub lets you host your docs in one centralize platform, and easily give controlled access to your API's development team or end consumers. You can make the documentation publicly available from within SwaggerHub, or set them to private and invite internal team members or partners for secure sharing.



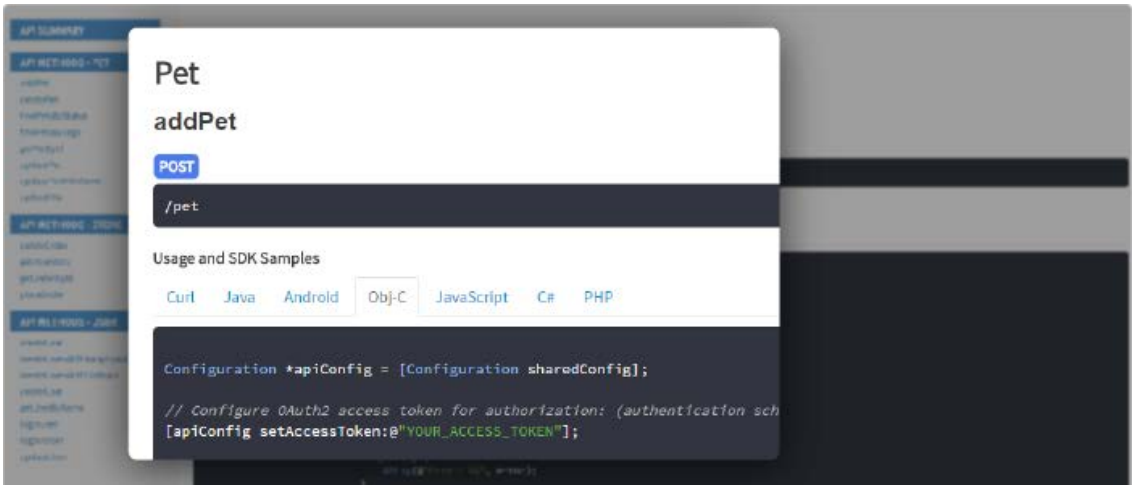

## Optimized, Confusion-Free Workflow

Your API's design and documentation has multiple stakeholders, and good collaboration is key to a successful API release. Work better together with tight control on what updates get added to your main API workflow. Collaborators can make copies of the main API by forking it, add changes, and merge changes back into the main API workflow through the visual compare and merge feature. As your API changes, your teams are instantly notified with email notifications delivered to their inbox.



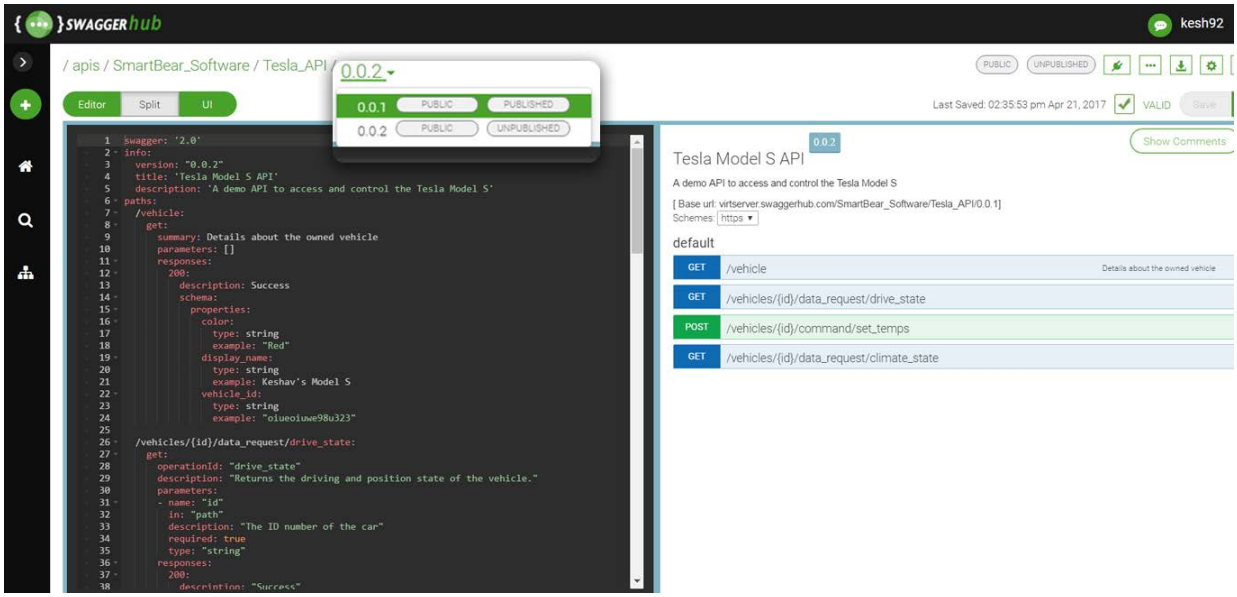
## Bootstrapped API Dev Portal

Your API's developer portal is a natural extension of great developer experience, containing documentation and code snippets to help end consumers understand and work with your API. Get your developer portal's base client side code generated for you to build upon with SwaggerHub's in-built HTML-portal generator. With one click, SwaggerHub generates the HTML template, with your API's documentation and 6 client SDK usage examples for your developer portal, making it super easy for your development team to customize, interact and work with.




## Maintain and Iterate

Organizations can centrally maintain all their API documentations and associated versions.Incrementally build on top of existing API documentation, or maintain documentation for multiple versions of your API in production with SwaggerHub's versioning system. Publish documentation for a stable, working API by building on an existing version, and gracefully deprecate outdated versions.



# Improving Developer Experience with Great API Documentation

Remember that documentation is the usage manual of your API, and is one of the biggest drivers to achieving your API's business goals. The Swagger framework alleviates documentation concerns, creating interactive, human and machine readable documentation, that's auto generated and needs minimal maintenance.

Creating API documentation your consumers will love takes effort, but the investment will have a significant payoff in the form of a great developer experience, easier implementation, and improved adoption of your API.

SwaggerHub takes care of all the infrastructural considerations and security implementation out of the picture, allowing organizations to seamlessly collaborate and create great API documentation that consumers and development teams will love.

More than 40,000 API developers and organizations trust SwaggerHub to help improve their API documentation workflow.

**Find out how SwaggerHub can help your team.** Start your free 14-day trial of SwaggerHub today, or reach out to the team directly to learn more about our enterprise solutions at info@swaggerhub.com.

*"Crowdflower has been using Swagger to define our APIs for some time, and that process has become significantly easier thanks to SwaggerHub. Having great tools like Swagger and SwaggerHub that promote collaboration when designing new services — and makes documenting and integration testing those services much easier — is a huge help to our team."*

***Cameron Befus VP Engineering, CrowdFlower Inc.***

CrowdFlower