HOW TO OPTIMIZE YOUR SWAGGER WORKFLOW FOR TODAY'S API ECONOMY

SWAGGERhub

Recognizing APIs as a driver of business and strategic initiatives is a fairly recent development, and organizations have now started to invest heavily in their API strategy. Organizations that are making significant investments in developing API strategies go beyond traditional technology leaders. Industries — from financial services to healthcare, education to manufacturing, and many more — have seen a significant growth in API adoption.

This has been accompanied by an explosion in adoption in API description formats, like Swagger (OpenAPI), which provide a contract for your API that is language agnostic and human readable, allowing both machines and humans to parse and understand what the API is supposed to do.

Today, thousands of organizations — both large and small — use Swagger to streamline development and drive adoption for their APIs. Implementing Swagger across your organization's internal and external APIs comes with a unique set of challenges, especially when attempting to scale multiple API projects across a number of different teams. This is where having a workflow the facilitate the implementation of Swagger can come in handy.

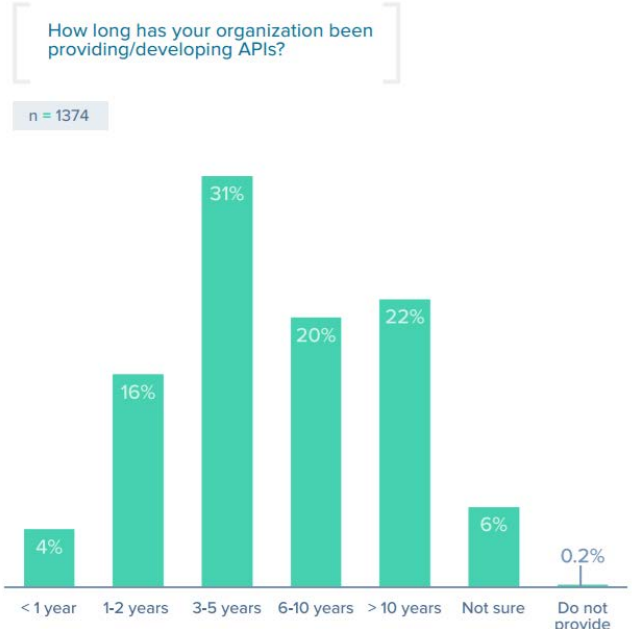
The goal of this whitepaper is to introduce an easy-to-follow workflow for working with Swagger within your organization.

Contents

The Current State of the API Economy

Application Programming Interfaces, or APIs, have been around long before the invention of the PC and were always viewed as technical assets. However, in the past decade, the disruption in information — caused by the Internet, social media, e-commerce, and mobile devices — has led more organizations to understand the importance of APIs as channels of distribution for data and services.

Today, it is estimated that the number of public APIs is around 50,000. This does not include the thousands of privately managed internal APIs, which continue to gain adoption across major industries around the globe. According to SmartBear's 2016 State of API Report, 1 in 5 API providers only began developing APIs within the last five years alone.



API adoption is only set to increase with the exponential rise in the number of connected devices. The IoT revolution is also upon us, with home devices, bots, and wearable devices all contributing to our growing desire to stay connected. By 2020, it is estimated that every person in the world will have seven devices, on average.

This is further proof that organizations should start thinking about their API strategy, as APIs are the best way to penetrate across all of these different platforms and devices.

Today, there are two primary models organizations take to fuel strategic growth with APIs — the closed, partner-only API model, where only partners of the API's owner can access and consume the API, or the open, public model, where anyone in the general public can access and consume the API, usually for a fixed number of transactions.

Closed Partner-Only Model

A partner-only model relies heavily on handcrafted relationships with your business partners to use the information provided with your API. It allows for a much more controlled exchange of data, which makes for an easy-to-monitor API program. In some cases, partners may also need to share the app ideas and the reasons for integration, and companies can decide if it fits their privacy policy and strategic guidelines, before providing them with the partner status.

It's expected that API partner programs will have multiple tiers of access, which allows for upsell and enables you to derive greater value from your organization's information store. Amazon Web Services Partner Network is a great example of this type of program, which offers various tiers of service to partners, based on their status. The success of this program is dependent on identifying the best engineering and strategic partners within their API ecosystem for the business to leverage.

Benefits of a partner API model:

- **Monetize information:** A partner model is a quick route to monetizing the information your organization has available, by providing access to partners that you already have an established relationship with.
- **Better control and maintenance:** It's much easier to provide a more tailored experience to partner companies, especially paid partners, because there's a limited set of partnerships.
- **Helps existing partners:** It also benefits existing partners, allowing them to access better information, providing more incentive for companies to get into a business relationship with your organization.
- **Opportunity for structured growth:** Partner programs provide an incentive for companies to promote your products.

Open "Public" Model

The open, public API model allows anyone in the general public to access and consume the API. Usually, there is a limit to the number transactions per unit time. The most common example of this model can be seen in social media APIs.

Public APIs are regulated by the number of calls that can be made per unit time. For example, the YouTube API has a limit of one million calls-per-day. Anytime you exceed these calls, the organization or individual integrating with the API will have to pay a fee. Since a public API is open for anyone, it requires support from a good API management solution, either built in-house or from a reliable API Gateway vendor like AWS, IBM, or Microsoft. Issues like rate limiting and monitoring are important to ensure that your API, and the information you provide, is not misused or exploited. It's also important to ensure that your public API is in a position to scale and handle high volumes of traffic flexibly.

Benefits of a public API model:

- **Opportunities for innovation:** An open API also enables third-party developers to create a rich and broad range of value-added products, that improve on the original functionality of your services. Twitter, for example, has clients, bots, and analytics tools built on its API.
- **Market expansion:** Public APIs can open business opportunities that are otherwise not accessible via traditional business strategies. Providing services via a public API enables many startups to compete against more established players by being available 24/7 (in a global market), with a ready-to-use service.
- **Increased platform awareness:** A public API program helps in spreading awareness about your product, and allows more products to rely on your services. Facebook, for example, has become the number one identity management solution, with mobile and web apps enabling users to sign up with their Facebook account, promoting the usage of Facebook in the process.

The Role of Swagger in the API Economy

With more and more APIs emerging, each of them being built on different programming languages and technology stacks, the need to unify them became clear. This was the problem which Swagger hoped to solve when it was created in 2010.

Swagger is an open source, human and machine readable API framework to design, build, and document APIs. The framework helps define a language agnostic, human readable format for APIs, that eases implementation, drive adoption, and stabilizes development.

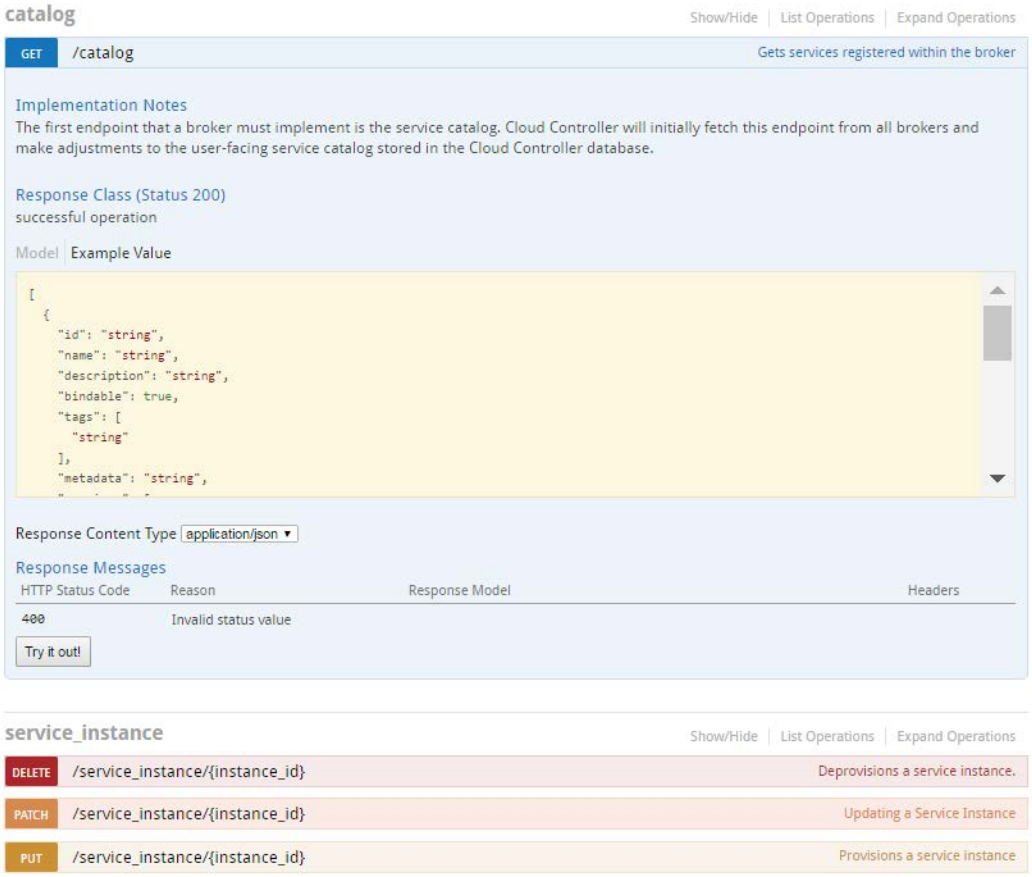
Swagger is now the world's most popular framework for API development, with over 3 million monthly downloads and has evolved into the industry standard for designing and documenting APIs.

At the core of Swagger is the OpenAPI Specification. The specification is a language agnostic human-readable format for describing the operations of your API. The specification helps define the API's contract for your API development teams and its end consumers, which sets clear expectations for the experience of integrating with the API.

The contract below is an example of the specification:

```
swagger: '2.0'
info:
  version: 1.0.0
  title: Echo
  description: |
    #### Echos back every URL, method, parameter and
header
    Feel free to make a path or an operation and use
**Try
Operation** to test it. The echo server will
    render back everything.
schemes:
  - http
host: mazimi-prod.apigee.net
basePath: /echo
paths:
  /echo:
    get:
      responses:
        200:
          description: Echo GET
    post:
      responses:
        200:
          description: Echo POST
      parameters:
        - name: name
          in: formData
          description: name
          type: string
        - name: year
          in: formData
          description: year
          type: string
```

One of the biggest reasons why the Swagger framework has been so widely adopted is the ability to generate interactive documentation straight from the API contract. The interactive documentation has evolved into its own specific tooling called the Swagger UI. The Swagger UI allows both development teams and end consumers to visualize and interact with the API's resources, without having to manually integrate and implement the API into their own applications.



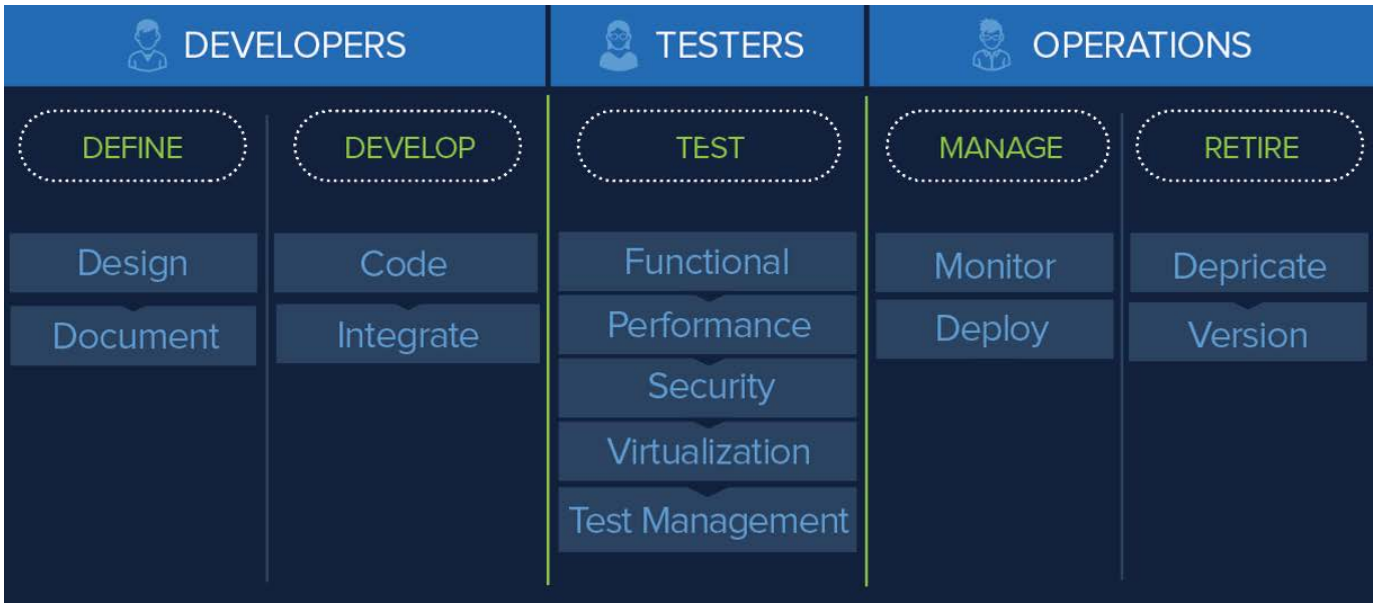
The benefits of adding Swagger to your API:

- **The proper canvas for API design and collaboration:** Swagger provides a single point of focus for your team to collaborate on the design of the API before you begin coding.
- **A clear, concise, and formal mechanism for reuse with integrity:** There are mechanisms inside the Swagger definition that are clearly set for reuse and reference across common objects. They don't require the person who wrote the definition to explain what it means.
- **The proper tooling for automation:** Swagger provides the right structure and tooling for server template generation — either traditional or serverless. Apart from server templates, it also supports the generation of client SDKs in almost every popular client language.
- **The necessary information for testing automation and monitoring enhancement:** Your QA team can import the Swagger contract into their testing solutions, reducing the guesswork that can be involved in understanding the APIs operations.

The Role of Swagger in the API Lifecycle

Taking APIs from concept to deprecation encompasses the entire API lifecycle, which has become a crucial engineering concern to build concrete API solutions. In order to succeed in today's competitive, interconnected business landscape, organizations must give their API programs a first-class treatment that gives developers the flexibility to integrate with their ever evolving internal and external environments, in the workflow of their choice. This is a big reason why choosing the right tool for the API lifecycle management becomes important.

For most teams, the API lifecycle will look something like this:



As discussed, Swagger plays a role in all stages of the API lifecycle, but has the most influence on the design, documentation, and development of the API.

Approaches to Building APIs with the Swagger Framework

With the popularity of API frameworks like Swagger, serious thought has been given on the best approach to building APIs based on historical usage and best practices. Two important schools of thought have emerged: The "design first" and the "code first" approach to API development.

The design first approach, also called the "contract first" approach advocates for designing the API's contract first before writing any code. This is a relatively new approach, but it's fast catching on with the growing popularity of API description languages.

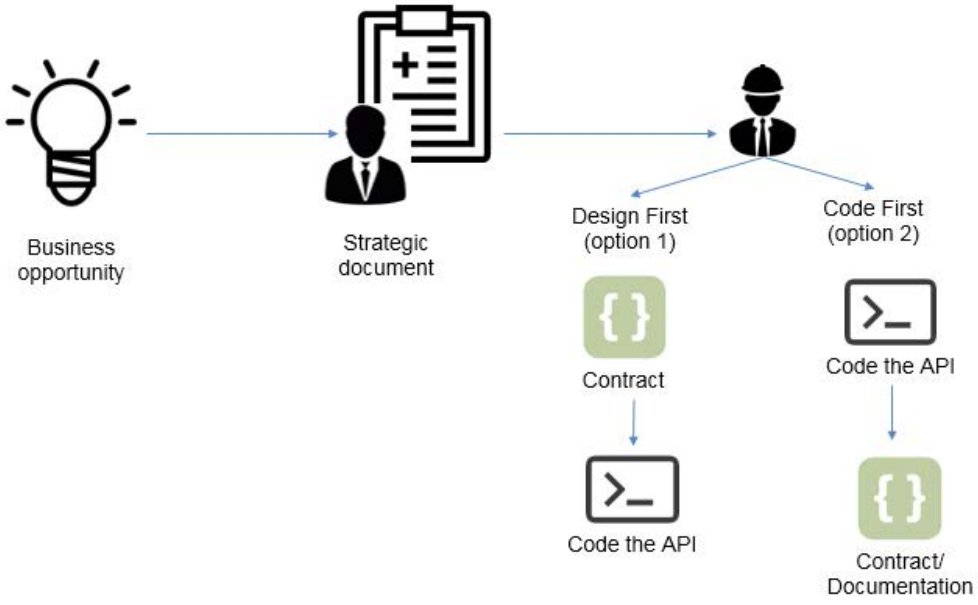
The code first approach is a more traditional form of building APIs with development of code happening after the business requirements are laid out, eventually generating the documentation from code.

To understand them better, let's look at the general process followed during the API lifecycle. Like any product, the concept of the API starts with the business team identifying an opportunity. The opportunity is analyzed and a plan to capitalize on it is created in a text document by strategic analysts, marketers, and other business people. This document is then passed along to the development team, which is where the plan takes some form.

There are two possibilities from here on:

- **Design First:** The plan is converted to a human and machine readable document, such as a Swagger document, from which the code is built.
- **Code First:** Based on the plan, some code is written which can then be converted to a human or machine readable document, such as a Swagger document.

Finally, after the API is functioning, it is sufficiently tested and then deployed to a suitable host.



Given how the API economy is shaping out to have such large scale financial, technical, and strategic opportunities, it's important to actually put thought into how your APIs are built. There are advantages and disadvantages associated with both approaches, and at the end of the day, choosing the right approach boils down to your immediate technological and strategic needs that you wish to solve with your APIs. In this next section, we will dive into the design first approach to look at how it can benefit your API development.

The Benefits of the Design First Approach

If organizations seriously want to make an impact in the API economy, then the design first approach is a good way forward. This is because the design first principle advocates for designing a great user experience first, before any line of code is written.

Designing APIs prior to implementing code also makes the API more RESTful and robust. Here are a few benefits of the design first approach:

Ensures Good Developer Experience

A well-designed API can do wonders for the adoption and consumption of your APIs, and good design can be better achieved with the design first approach. If your API strategy involves high adoption of your API and retention of users integrating with your API, then good Developer Experience (DX) matters.

An effective API design helps your end consumers quickly understand your API's resources and value propositions, reducing the time taken for them to integrate with your API. An API with consistent design decreases the learning curve when integrating with your API, making it more likely to have higher reuse value and engagement.

Delivers Mission Critical APIs

The biggest reason to go for the design first approach is when your API's target audience are external customers or partners. In such a case, your API is a key distribution channel that your end customers can use to consume the services you provide, and good design plays a key role in determining customer satisfaction. Such APIs play a critical role in representing your organization's services, especially in an omni-channel ecosystem, where consistency in information and hassle-free consumption is an important indicator of business success.

Allows for Better Communication

The API contract can act as the central draft that keeps all your team members aligned on what your API's objectives are, and how your API's resources are exposed. Identifying bugs and issues in the API's architecture with your team becomes easier from inspecting a human-readable design. Spotting issues in the design, before writing any code is a much more efficient and streamlined approach, than doing so after the implementation is already in place.

The approach you take to developing your APIs will play a vital role in determining how they're consumed and maintained. Who are your API's end consumers? What needs do they have? What are you trying to solve with your API program? These are the types of questions that should guide your decision making when it comes to choosing the right methodology to your API development.

Optimize Your API Workflow with SwaggerHub

One of the biggest challenges teams face when implementing an API strategy is ensuring that different team members, with varying skillsets and responsibilities, get involved at the right stage of the API lifecycle.

Other challenges include:

- **Team management and collaboration:** How do you work across teams to update Swagger definitions, and write detailed documentation for your API?

- **Version management:** As organizations continue to iterate on the design and documentation of an API, the complexity of managing multiple versions of their APIs is introduced. How does an organization effectively manage multiple versions of the same API?

- **Secure hosting and sharing of Swagger files:** Maintaining a central repository for teams to access Swagger definitions becoming increasingly important as you scale your API strategy with Swagger. How do teams think through securely orchestrating various aspects of their API lifecycle?

- **Sync with existing toolset:** How do you keep your internal and external environments updated with the different versions of the API, especially when having your software and API development taking place across various external systems like Source Control Hosts, API Management platforms and testing suites?

Addressing these challenges requires organizations to adopt an optimal API development workflow, and identify the necessary tools to optimize that workflow across all stages of the API lifecycle — from design and document to test and deployment.

Taking the needs of the hyperconnected API economy into account, SwaggerHub caters to every aspect of the API lifecycle, for a centralized yet flexible solution for an API program. SwaggerHub acts as your centralized source of truth for your Swagger needs, orchestrating them across every aspect of your API lifecycle, with the flexibility to integrate with the tool set of your choice, without any vendor lockdown. It is optimized for the design first approach, with a host of collaborative and development features to support this, though it also caters to organizations who want to document existing APIs.

The SwaggerHub Workflow

Design, development, and documentation are integral parts of releasing great APIs. In software teams, roles overlap with a constant exchange of information between different teams to build the perfect API. In this final section, we will walk through the SwaggerHub's collaborative API workflow. We will be walking you through creating and managing organizations while highlighting how to iterate and deliberate in teams on the API development process using SwaggerHub.

Defining Organizations and Teams

SwaggerHub's collaborative features were built keeping in mind the needs of the modern team to allow for iterative and cross-functional development of APIs. This is possible by defining organizations and teams within SwaggerHub. An organization's power can be fully realized through teams, a feature which lets organization owners group members together to collectively work on APIs.

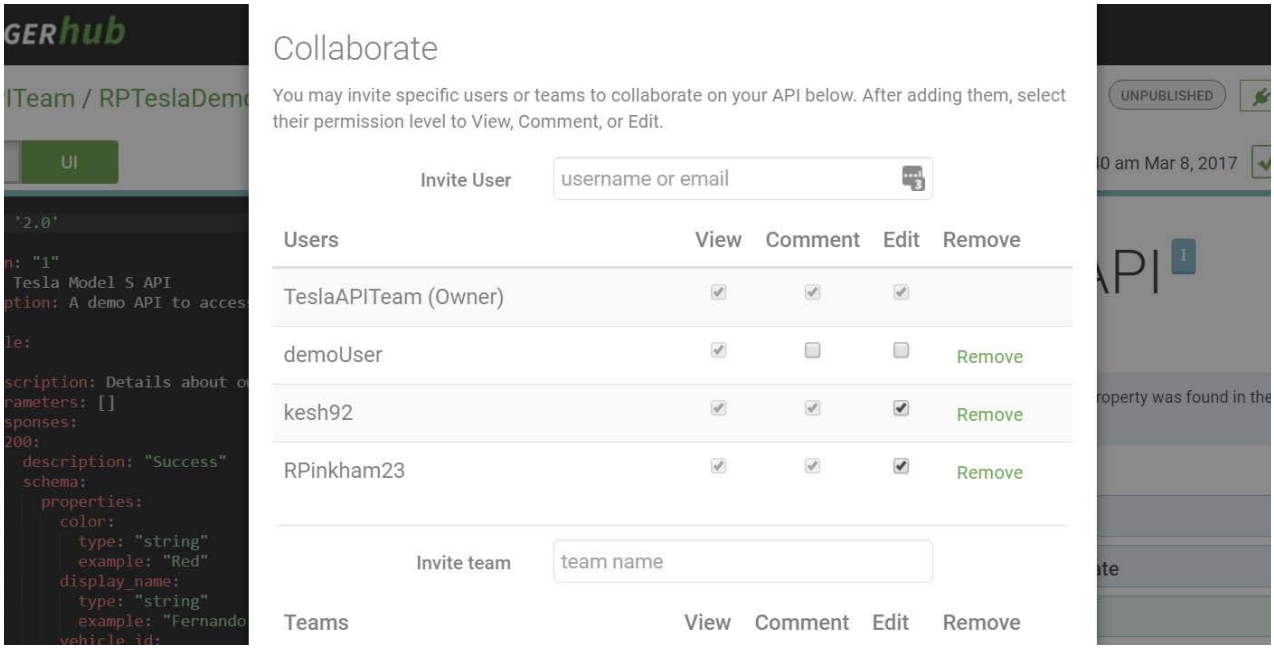
Examples of teams you can setup within your organization, include:

- **API Architects** - responsible for the design of the API.
- **API Documentation** - responsible for documenting the API.
- **API Development** - responsible for developing the server and client side code for the API.

Creating APIs Under Your Organizations

Various stakeholders manage different pieces of the development process of an API. Every API starts with understanding the requirements. Is this a private API to be used internally by the company to build better software, or is it a public facing API which could be used to drive the market forward and raise awareness of the company's products and services? Depending on the requirements set forth by the product owners and technical leads, the API team would identify the need for APIs, be it public or private. SwaggerHub lets owners define the visibility of an API directly in the SwaggerHub editor.

Private APIs are usually spearheaded by technical leads and development managers that recognize the need for an internal API to ease the development process of other applications within the organization. Product owners, marketing managers, and technical leads usually lead the discussion for public APIs which are open to the masses for consumption. It is only when the vision, purpose, and technology for the API are fully explored and set, can the API design and development team help make it a reality.



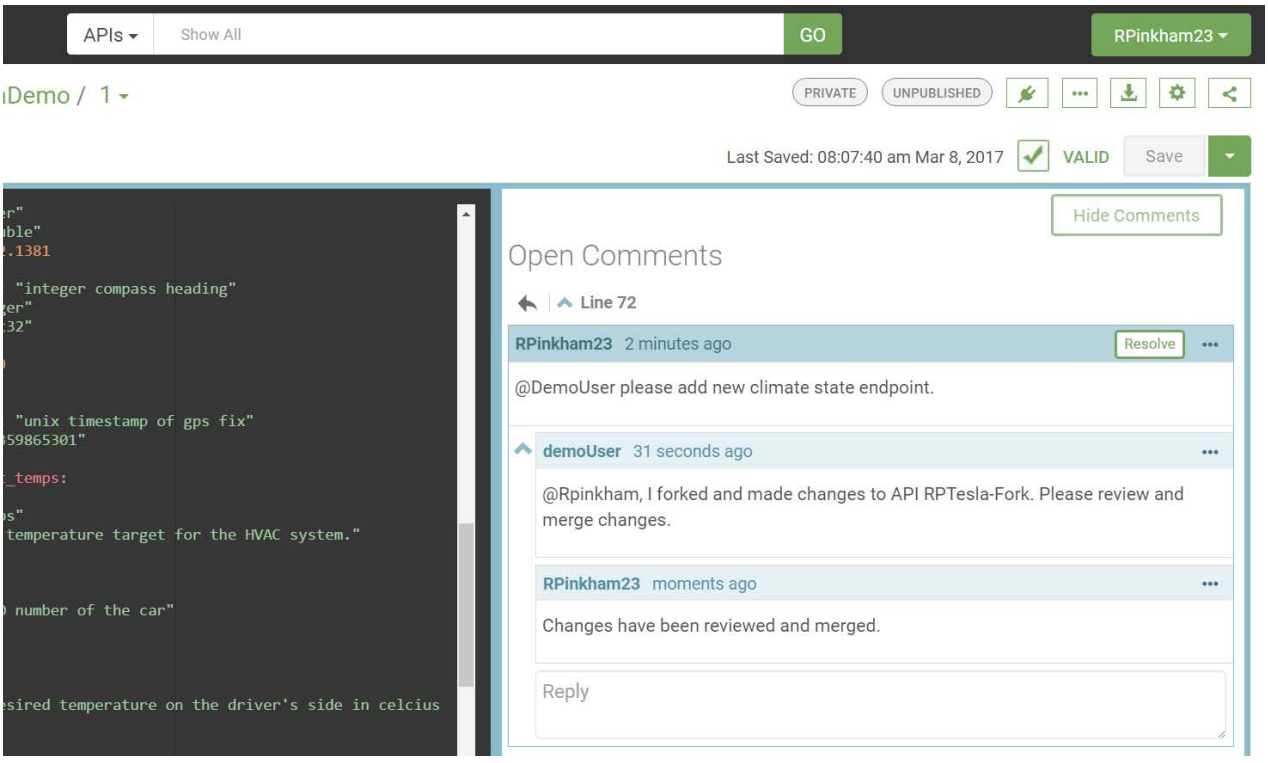
Adding Collaborators

APIs on SwaggerHub, be it public or private, can have multiple collaborators associated with it. Once the API is created, from the SwaggerHub editor, organization owners can add teams as collaborators to the API with the privileges of their choosing. Collaborators can edit the API and communicate with other collaborators via comments. Collaborators can be given the ability to edit the API (Editor), comment on individual lines of the Swagger specification (Commentator) or simply have the ability to view the API (Viewer). Use comments to discuss ideas or point to issues in specific lines of your API definition in the SwaggerHub editor.

Collaborative API Design

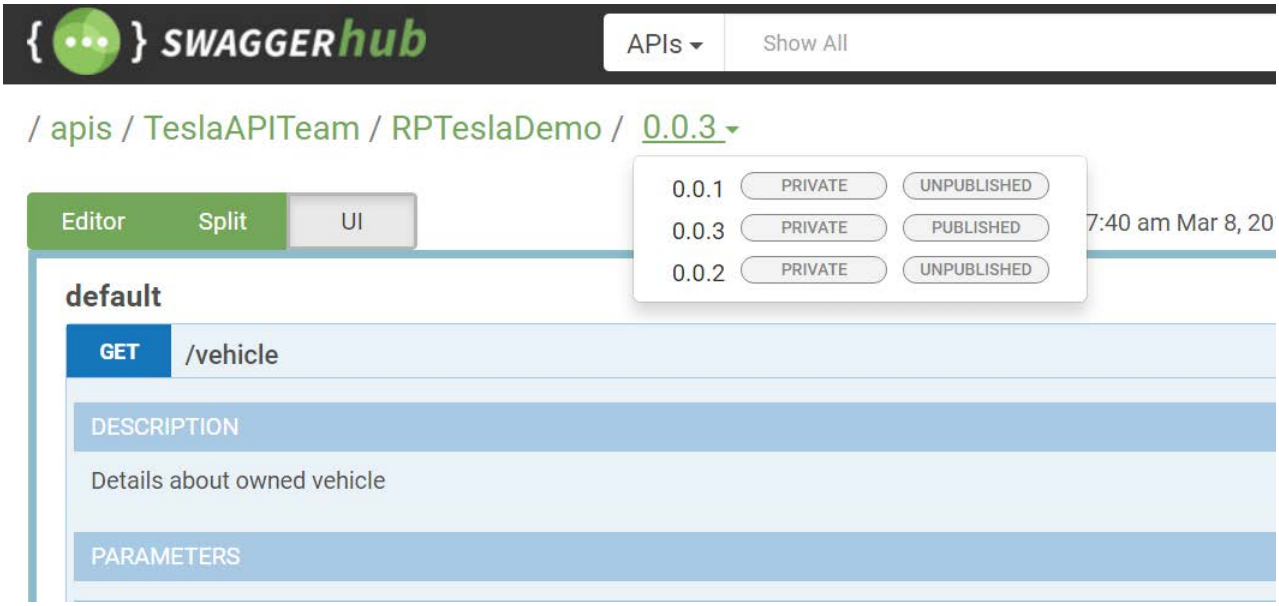
Team members can deliberate on new ideas, highlight issues on various lines of the specification and resolve them together using SwaggerHub Comments. Allow your collaborators to fork APIs, add changes, and merge them back, giving organizations an enhanced and controlled workflow that lets your entire team move forward together.

Whether you're updating a Swagger definition that's already in use, or adding Swagger to an existing API, it's helpful to compare the changes you're making before you update the API. You can compare your API specification on SwaggerHub with another API spec on SwaggerHub or your local file system, and merge these changes directly back into your main API. Unlike a traditional diff & merge solution, the SwaggerHub Compare & Merge feature is built specifically for reviewing and implementing changes to the Swagger definition. It provides the right amount of feedback and will help validate that your API is accessible, based on the changes you're making to the definition.



Iteration Through Versioning
Software teams release new and updated versions of their applications. These versions build on top of previous versions adding new changes or bug fixes, without rewriting existing code or syntax. SwaggerHub's versioning system can help with this. Collaborators can define multiple versions of each API, meaning new revisions and edits can take place in multiple versions without overwriting the source.
All collaborators will be given instant access to new versions, making it super simple to just continue working on multiple updates of the API. Versioning is important not just to keep track of revisions, but to also to keep the latest API concept and design updated.

Once the teams are satisfied with the API design, and all the back-end services and documentation are ready, the API can be published. By publishing the latest version, all the users with access to the API know that the version is stable and can be safely referenced and consumed. Publishing an API makes it read-only. You can edit it only if you unpublish the API again.



Beyond Design
After you finalize the design and publish the API, all the collaborators get notifications and can start working on client-side SDKs and back-end services.

SwaggerHub supports generating server-side templates and client SDKs, which takes care of all the boilerplate code letting developers focus purely on the business logic. With the design, back-end services and client SDK ready, all that is left is to deploy and manage the API, possibly, with SwaggerHub's API Management Integrations like Azure or AWS Gateway.

Using the power of organizations, teams, member roles, version control and commenting, SwaggerHub makes for a great experience for collaborative API development.

Optimize Your Workflow with SwaggerHub

SwaggerHub offers flexible plans for teams of all sizes. Store all your Swagger assets in one central location, setup and manage teams, and collaborate across the lifecycle of your API. Available in the cloud or installed within your firewall with our Enterprise solution.

Plan and work within your organization

SwaggerHub is built for teams — allowing you to add multiple team members to collaborate on the design and build of your API. SwaggerHub's team management feature allows you to set up specific working groups to help you prioritize and work effectively. Assign different teams to the right APIs based on strategic and technological goals, with real-time communication and issue tracking to support your collaborative needs.

Store, secure and centralize your APIs

Keep your APIs securely hosted on one centralized platform for all your teams and organization members to work on. Provide the appropriate access and privacy to the right people and APIs based on your organizational needs, and effectively manage your APIs across multiple stakeholders and consumers.

Flexibly Integrate with your external tools

Give your team the flexibility to build better APIs. SwaggerHub offers integrations with popular third party services that tie into every phase of the API lifecycle. Work with enterprise grade Source Control Hosts, API Management platforms or trigger external services with custom webhooks. We offer tailor made integrations to all your external services with SwaggerHub Enterprise as well.

Get started today! 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