SMARTBEAR

SWAGGER | White Paper

Documenting Your Existing APIs:

API Documentation Made Easy with OpenAPI & Swagger

Good user experience is key to using any product, and the same holds true for APIs. The better the interface that's used to consume APIs, the higher the chance of achieving your business and technological objectives. Since the advent of mobile and cloud computing, APIs have gone mainstream, with more and more companies and organizations understanding the business value of creating APIs. With a lot of web services emerging, the need to have clear API documentation for adopting these services became clear.

API documentation is the information that is required to successfully consume and integrate with an API. This can be in the form of technical writing, code samples and examples for better understanding how to consume an API. 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.

Why documentation matters

A survey by ProgrammableWeb found that API consumers consider complete and accurate documentation as the biggest factor in 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. Documentation is part of the overall user experience, and is one of the biggest factors for increased API growth and usage.

Challenges of API documentation

APIs, like so many other products, tend to evolve rapidly during development and release cycles. 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. The second issue is facilitating interaction between multiple web services. Applications are made up of multiple services that constantly communicate and interact with each other.

As RESTful services grow in number, so do the programming languages that are used to implement them, making it harder for them to communicate. API documentation can be thought of as the interface for consuming an API and needs to facilitate interaction between these different web services. Regular API interfaces, be it text documentation, or others like Javadocs, do not allow them to communicate with each other. These challenges, along with other API pain points, led to the creation of the Swagger Specification.

In the next section, we'll take a closer look at how the OpenAPI Specification (formerly known as the Swagger Specification) can help address your documentation challenges.

Why use OpenAPI for API documentation

The Swagger Specification, which was renamed to the OpenAPI Specification (OAS), after the Swagger team joined SmartBear and the specification was donated to the OpenAPI Initiative in 2015, has become the de factor standard for defining RESTful APIs.

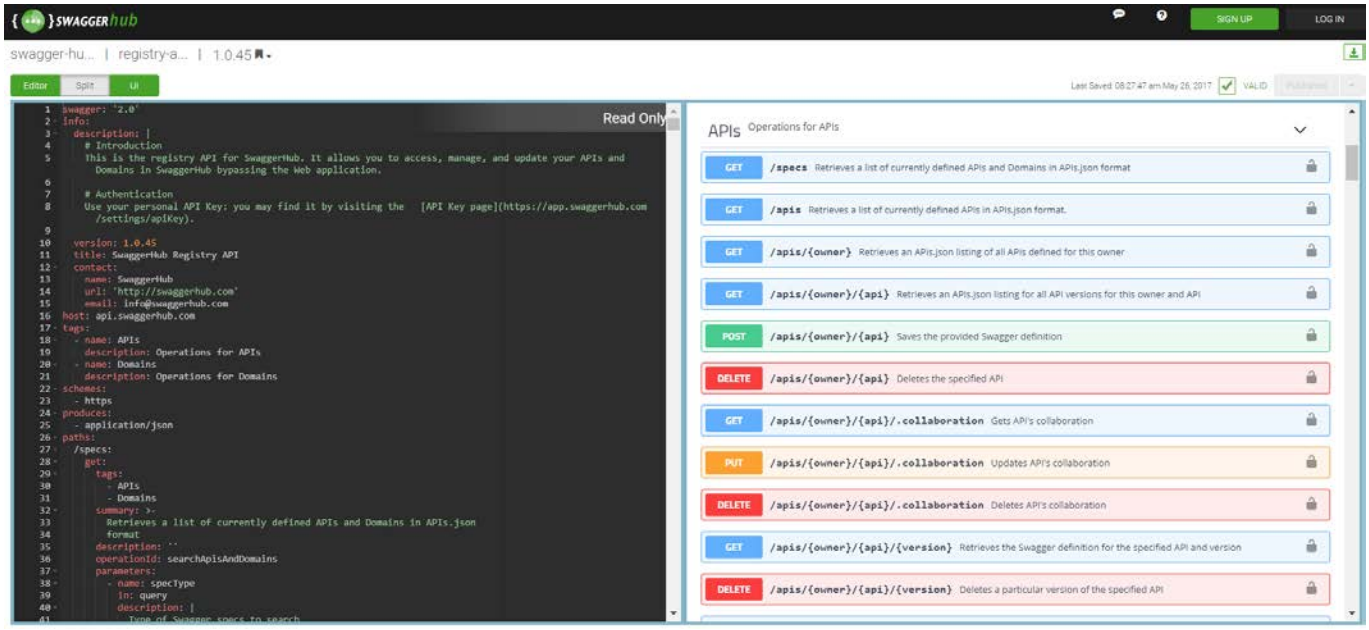
(Note: We will be using the term OpenAPI and OAS throughout this resource. This is meant to reference the Specification.)

OAS defines an API's contract, allowing all the API's stakeholders, be it your development team, or your end consumers, to understand what the API does and interact with its various resources, without having to integrate it into their own application. This contract is language agnostic and human readable, allowing both machines and humans to parse and understand what the API is supposed to do.

The OAS contract describes what the API does, it's request parameters and response objects, all without any indication of code implementation. Web services defined with OAS can communicate with each other irrespective of the language they're built in, since OAS is language agnostic and machine readable.

How does OAS help with documentation?

One of the biggest reasons why Swagger first gained adoption, was its ability to help streamline the documentation for RESTful APIs. Using a tool like Swagger UI — either open source or within the SwaggerHub platform — you can convert your OAS contract into an interactive API console that consumers can use to interact with the API and quickly learn how the API is supposed to behave.



Generating documentation for your API is just one of the advantages of defining your API with OpenAPI. Other benefits include:

- **Help internal team members understand the API and agree on its attributes:** An API definition allows documentation tools like the Swagger UI to visualize APIs. You can visualize all of our internal APIs so that developers could use upstream and downstream services quickly and easily. Team-based tools like SwaggerHub allow collaboration on the API's documentation and host them for internal consumption.

- **Help external folks understand the API and what it can do:** Again, Swagger UI is a well-used visualization tool for documentation. Not just for internal consumption, but for external adoption. The Swagger UI has interactivity built in, so external consumers can try each and every resource of an API and get comfortable with it before using it in their code base. Using the SwaggerHub platform, organizations can also provide controlled access to their external consumers.

- **Create tests for your API:** Your OAS definition provides a contract that describes what a successful response will look like when someone calls your API. This contract can also be re-purposed to generated test cases which can drastically decrease the amount of setup team needed for testing your APIs.

- **Generate implementation code and SDKs:** In addition to generating documentation, the OpenAPI definition can also be used to accelerate development by scaffolding implementation code and SDKs for the API. API Definition files can be imported into a number of different tools, like Swagger Codegen and SwaggerHub, to create these stubs code in many different languages, like Java, Scala, and Ruby.

Now that we've covered why your team should adopt OAS and Swagger tools into your API development workflow, the next question is how do you actually get started? In the next section, we'll take a closer look at different approaches to getting started with OAS.

Approaches to OpenAPI

When it comes creating the OAS definition, two important schools of thoughts have emerged: the "Design First" and the "Code First" approach to API development.

The Design First approach advocates for designing the API's contract first before writing any code. This is a relatively new approach, but is fast catching on, especially with the use of OAS. In the design-first approach, the API contract acts 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. 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.

If your team is ready to transition to a design first approach, you'll first need to get comfortable with writing an API definition. Here are some resources to help you get started in the process:

Tutorial: Learning the New OpenAPI Specification You can find documentation for OpenAPI 3.0 and Swagger 2.0 Specification on Swagger.io.

How to Design and Document APIs with the Latest OpenAPI Specification [Recorded Webinar]
This training provides a live demonstration of defining a new API using OpenAPI 3.0 in SwaggerHub.

OpenAPI 3.0 Official GitHub Repository
The OpenAPI Initiative

OpenAPI 3.0 Tutorial Overview
IdRatherBeWriting.com

OpenAPI Visual Documentation
APIHandyman.com

"Code First" Approach

The Code First approach (also commonly known as the "bottoms up" approach) is a more traditional approach to building APIs, with development of code happening after the business requirements are laid out, then the documentation of the API is done from the code.

For many API teams, getting started with OAS means starting with a "code first" approach, and generating the definition from an existing set of APIs. Let's explore a few of the other popular methods for generating an OAS definition when you already have existing APIs.

Generating an OAS definition with Swagger Inspector

Swagger Inspector is an easy to use developer tool to quickly execute any API request, validate its responses and generate a corresponding OpenAPI definition. Use Swagger Inspector to quickly generate your OAS-based documentation for existing REST APIs by calling each end point and using the associated response to generate OAS-compliant documentation, or string together a series of calls to generate a full OAS document for multiple API endpoints.

Perform quick API calls right from your browser window with Swagger Inspector. After you perform your first call, you can create a free account and save your call history within Inspector.

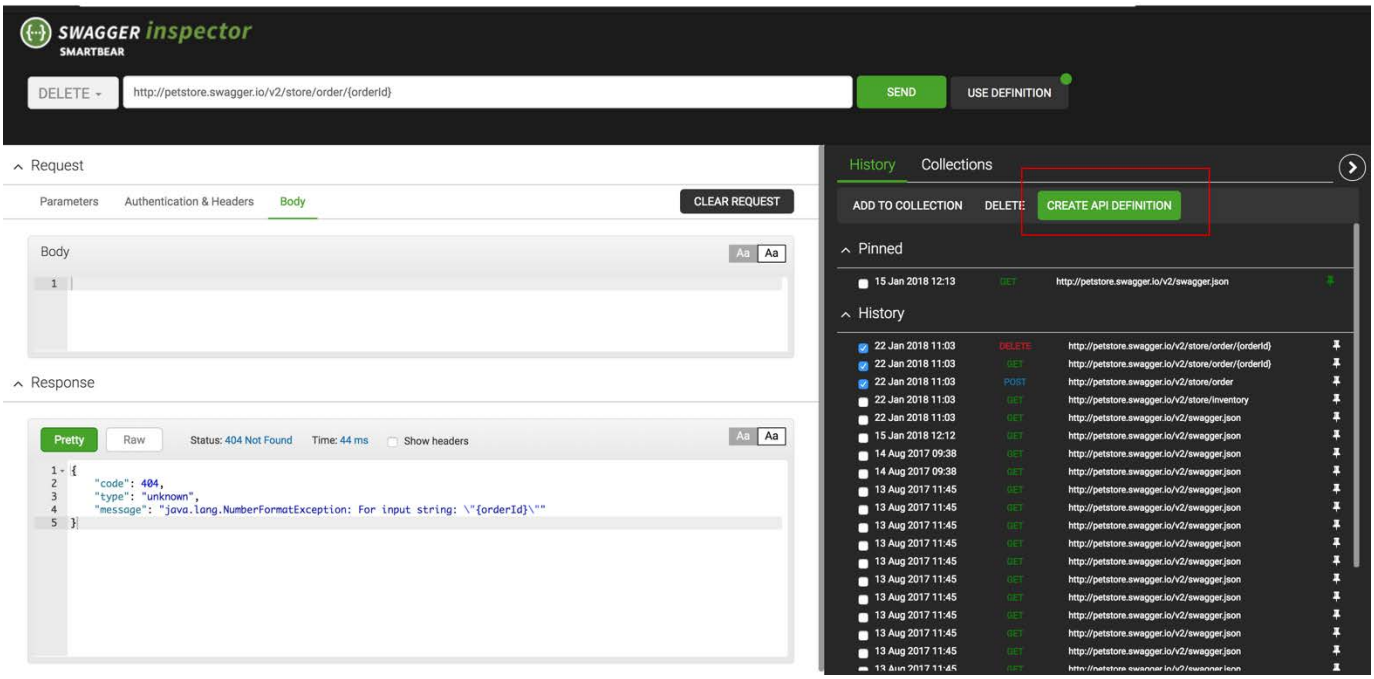
With Swagger Inspector, you can automatically generate the OpenAPI file for any end point you call, saving valuable development time, and allowing your technical writers to get started on creating great documentation.

Swagger Inspector is integrated with SwaggerHub, the API design and documentation platform for teams. The integration allows developers to automatically host and visualize their API documentation on SwaggerHub to enable API discovery and consumption by internal and external stakeholders.

How to generate OpenAPI from existing APIs

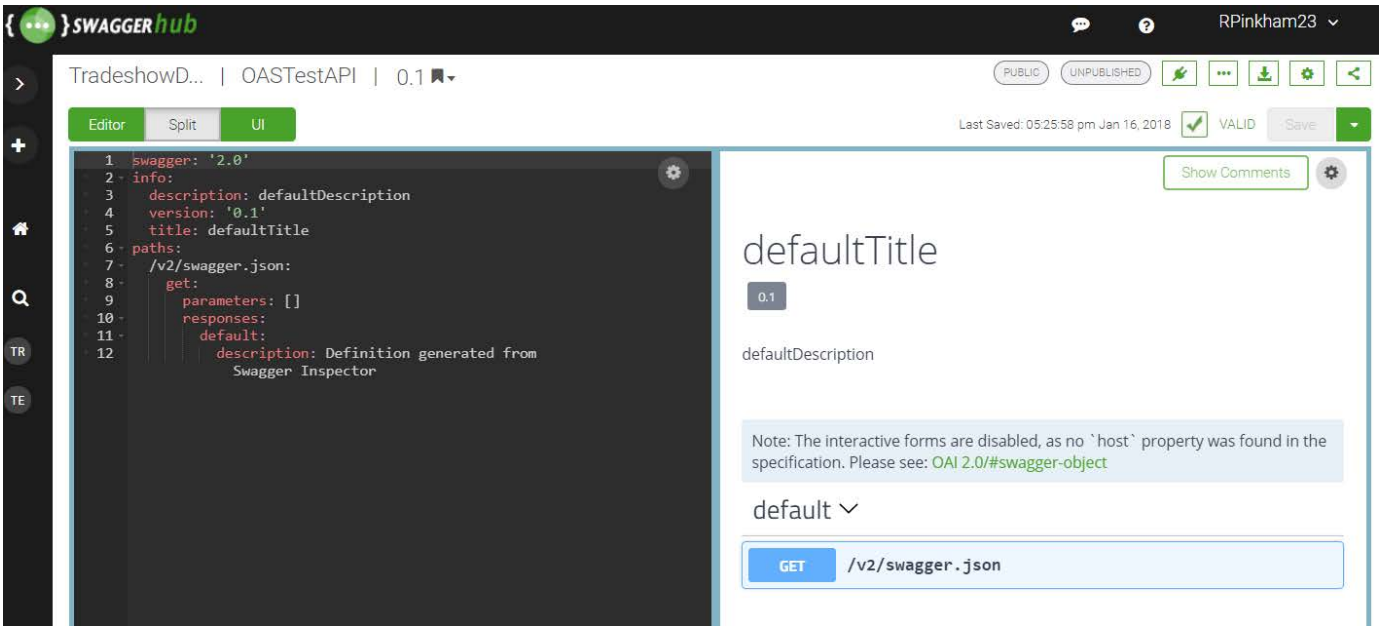
Head over to Swagger Inspector, and insert the end point of the resource you want to have documented. You can then navigate to the right panel from the History section of Swagger Inspector, and click "Create API definition" to create the OAS definition.

The cool thing about Inspector is that you can select multiple end points and consolidate their documentation in one single OpenAPI file through a Collection.



You will need a SwaggerHub account to host the generated OpenAPI file on SwaggerHub, as well as to save your history of calls in Swagger Inspector.

If you already have a SwaggerHub account, then you can log into Swagger Inspector with your credentials. When you create a Swagger Inspector account, you automatically join the SwaggerHub family. After you create an account, you can easily access all your tests in your history, anywhere at any time, and also generate the corresponding OpenAPI definition with the click of a button.



The generated definition will provide an OAS-compliant structure for your team to build out your API documentation. With the definition in place, you can add in important details like: supported content types, descriptions, examples, authentication types, and more.

We'll go into more detail on how you can continue to build out your API documentation later in this resource, but first, let's explore a few of the other popular methods for generating an OAS definition.

OAS Generation During Runtime

Swagger-core is the Java implementation of Swagger. Current version supports JAX-RS and plain servlets.

In this method, the Swagger/OAS contract is generated from an API based on the meta-data added against the various resources, methods and controllers. This meta-data will generate the contract, client-side code, and other artifacts during runtime. Typically, this meta-data will be in the form of code annotations. The tools trigger as the various methods and functions are called against their resources, and produces the OAS contract from the metadata defined in the API.

To better elaborate this process, let's consider a case where we have to generate the OpenAPI specification from an API coded using JAX-RS, with the Jersey framework.

There are three steps required to generate an OAS document from an existing API:

1. Adding dependencies to your application
2. Hooking Swagger Core to the Application
3. Initialize the OAS Contract

The Swagger project uses maven for build and deployment of artifacts, available on Maven Central. These maven dependencies would need to be added to your JAX-RS coded API for Swagger Core to run.

A typical maven dependency would look like:

```
<dependency>
    <groupId>io.swagger</groupId>
    <artifactId>swagger-jersey-jaxrs</artifactId>
    <version>1.5.12</version>
</dependency>
```

Because of differences in major versions of the Jersey REST framework, users should use the swagger-jersey2-jaxrs dependency for Jersey 2.x. The next step is to hook up Swagger Core into your API. Depending on the way Jersey is configured in your web service, you could hook up Swagger Core to your application using Spring, the Jersey's container Servlet, or manually.

Finally, based on the code annotations added in the previous steps, the OAS definition can be initialized within your application during its runtime. The generated OAS definition will be in two files, defined in JSON and YAML respectively.

Here are some additional resources to better understand this process:

1. Generating OAS for a Jersey Project
2. Generating OAS for Spring based APIs
3. Generating OAS for PHP based APIs

OAS Generation During Build Time

In this method, the OAS contract is generated when preprocessing the API, that is, before runtime. Comments against various resources, methods and functions within the API help generate the OAS definition. These comments are usually in a predefined, special syntax, based on the type of tool you use to generate the contract. The tool scans your API code for these special comments and produces the OAS contract as an output. Cakephp-swagger and grape-swagger are prominent examples of tools that generate the OAS contract during build time.

There are disadvantages and advantages offered by any method. In terms of ease of use and speed, Swagger Inspector beats the rest. With less than five clicks, users can have a fully structured OAS definition from their existing APIs hosted on SwaggerHub. Swagger Inspector generates only the foundation of the final documentation, and writers still have to spend time to accurately detail the resources, methods and the way you'd use them to a consumer. Generating the OAS specification during runtime produces a more accurate API contract from the code, at the cost of software load in the form of additional dependencies, development time, and may add some overhead to the system.

Conversely, generating the OAS contract before runtime of the API is a more lightweight process, but there's a good

chance that the OAS contract produced may not accurately describe your API, as it must be manually maintained. In both approaches, there will likely be some additional work needed to ensure the OAS file generated accurately represents the operations of your API.

Documenting the API from the OAS Definition

No matter which approach you take to generating your OAS definition, there is still a good amount of additional work that will be needed to build out your API documentation.

With great tools like Swagger Inspector or Swagger Core, you'll have an OAS-compliant structure in place that will make it easy to fill in important details for each of your API endpoints. The generated file is the basis of your API's technical and interactive documentation.

Documentation from the generated contract will mean adding meaningful, understandable information that your end consumers can use to achieve API success. OAS lets you describe important details, including:

- **Media types:** Media type is a format of a request or response body data. Web service operations can accept and return data in different formats, the most common being JSON, XML and images.

- **Endpoints/Resources:** These may have an optional short summary and a longer description for documentation purposes. This information is supposed to be relevant to all operations in this endpoint. Description can be multi-line and supports Markdown(CommonMark) for rich text representation.

- **Request bodies:** Request bodies are typically used with "create" and "update" operations (POST, PUT, PATCH). For example, when creating a resource using POST or PUT, the request body usually contains the representation of the resource to be created.

- **Responses:** An API definition needs to specify the responses for all API operations. Each operation must have at least one response defined, usually a successful response. A response is defined by its HTTP status code and the data returned in the response body and/or headers.

- **Authentication:** Authentication is described by using the securityDefinitions and security keywords. You use securityDefinitions to define all authentication types supported by the API, then use security to apply specific authentication types to the whole API or individual operations.

- **Examples:** You can add examples to parameters, properties and objects to make OpenAPI specification of your web service clearer. Examples can be read by tools and libraries that process your API in some way.

There are just a few examples of the type of information that can be defined within your OAS definition. You can learn more about documenting your API using OAS here.

Remember that documentation is the usage manual of your API, and is one of the biggest drivers to achieving your API's business goals. 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.

In the final section, we'll take a look at how SwaggerHub can help further your API documentation workflow with OAS.

API Design and Documentation in SwaggerHub

Documentation can be a tricky process. It's a manual, collaborative operation that expects a lot of time, quality and empathy from the writers. When traversing the journey from API code to documentation, the most important thing to have is a seamless workflow that doesn't make you break a sweat with additional setup. It is usually recommended to give API documentation its own, unique care and treatment, since documentation is the first interface that's used by users and customers to consume your API offering.

SwaggerHub is an integrated API design and documentation platform, built for teams to drive consistency and discipline across the API development workflow. It is a dedicated platform for all the work, with all the configuration and hosting taken care of, allowing

you to seamlessly integrate documentation into your API workflow.

Once your API's contract is generated from your existing API code, you can import it in SwaggerHub, and continue your API journey. The interactive documentation is automatically generated and hosted on SwaggerHub. The definition can be edited, with your technical writers adding the right information in your API that can gives its consumers the required information to integrate with it. SwaggerHub's built-in tools further assist in the documentation process.

Some of them include:

- **Secure Cloud-Hosting:** Store your API definitions in a platform built for APIs. SwaggerHub auto-saves your work throughout the documentation process and provides a central place to host your documentation, without the need to setup a server.

- **Standardization and Governance:** Have all your APIs compliant with your organizational design standards to improve your consumer's experience. No more wiki pages, GitHub documents or PDFs that your developers have to reference to maintain style consistency across your APIs, with SwaggerHub standardizing them for you.

- **Collaboration & Issue Tracking:** Work with multiple stakeholders on a centralized platform. SwaggerHub provides the platform for teams to collaborate throughout the design and documentation process, collecting feedback and tracking issues in real-time with comments in the SwaggerHub editor.

- **Deploy to API Gateways:** SwaggerHub acts as the source of truth for your API definitions, letting you handle your design and documentation work in the cloud, and seamlessly push your OAS definitions to API gateways like AWS, Microsoft Azure, Apigee, and more.

Get started today!

Using OAS with the Swagger tools alleviates documentation concerns, creating interactive documentation, that's auto generated and needs minimal maintenance. Need to generate an OpenAPI definition for an existing set of APIs?

Try out Swagger Inspector.

Looking to standardize your design and documentation process? Get started with SwaggerHub today.

About SmartBear Software

Supporting more than five million software professionals and over 20,000 companies in 194 countries, SmartBear is the leader in software quality tools for teams. The company's products help deliver the highest quality and best performing software possible while helping teams ship code at nearly impossible velocities. With products for API testing, UI testing, code review and performance monitoring across mobile, web and desktop applications, SmartBear equips every development, testing and operations team member with the tools to ensure quality at every stage of the software cycle. For more information, visit: http://www.smartbear.com, or for the SmartBear community, go to: Facebook, Twitter, LinkedIn or Google+.