

Describe Your API With OpenAPI

Ben Ramsey • RESTfest • Sep 28, 2019

Hi, I'm Ben.

- Software Architect at ShootProof
- Organizer at Nashville PHP
- Hypermedia fan
- ramsey/uuid library
- Craft beer nerd
- @ramsey on Twitter



OpenAPI

What is it?

Who's behind it?

OpenAPI Initiative (OAI)

“The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how REST APIs are described. [...] the OAI is focused on creating, evolving and promoting a vendor neutral description format.”

openapis.org/about

OpenAPI Specification (OAS)

“defines a standard, programming language-agnostic interface description for REST APIs [...] When properly defined via OpenAPI, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.”

github.com/OAI/OpenAPI-Specification

- No need to rewrite existing APIs
- Doesn't require binding any software to a service
- You don't have to be the service owner to write a description for it
- Allows for describing the service capabilities
- Not all services can be described
- Does not mandate a specific development process

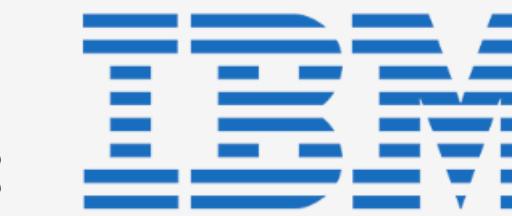
Linux Foundation Collaborative Project

“[Linux Foundation] Collaborative Projects are independently funded software projects that harness the power of collaborative development to fuel innovation...”

openapis.org/faq



Assertible



ORACLE + apiary



Swagger?

- The OpenAPI Specification was originally created by SmartBear Software
- It was originally named the “Swagger Specification”
- SmartBear donated Swagger to the OAI at its inception

But don't call it “Swagger”

What is Swagger now?

**A set of tools for
working with OpenAPI.**

Version History

- Swagger 1.0 released in 2011
- Swagger 1.2 released as a formal specification in early 2014
- Swagger 2.0 released in late 2014
- SmartBear donated Swagger to the OAI on December 31, 2015

- OpenAPI Specification 3.0.0 released on July 26, 2017
- Followed by very minor patch releases:
 - 3.0.1
 - 3.0.2 (current version)
- OAI is currently working on version 3.1:
<https://github.com/OAI/OpenAPI-Specification/issues/1466>

Boring! What does it look like?

```
paths:  
  /pets:  
    get:  
      summary: List all pets  
      operationId: listPets  
      tags:  
        - pets  
      parameters:  
        - name: limit  
          in: query  
          description: How many items to return at one time (max 100)  
          required: false  
        schema:  
          type: integer  
          format: int32  
      responses:  
        '200':  
          description: A paged array of pets  
          headers:  
            x-next:  
              description: A link to the next page of responses  
              schema:  
                type: string  
          content:  
            application/json:  
              schema:  
                $ref: "#/components/schemas/Pets"
```

Swagger UI Example X +

localhost:3200/swagger-ui.html#/

Swagger Supported by SMARTBEAR [Explore](#)

./examples/petstore.yaml

Swagger Petstore 1.0.0 OAS3

./examples/petstore.yaml

MIT

Servers

http://petstore.swagger.io/v1 ▾

pets

GET /pets List all pets

POST /pets Create a pet

GET /pets/{petId} Info for a specific pet

Schemas

Pet >

Pets >

Error >

This screenshot shows the Swagger UI interface for the Petstore API. At the top, it displays the title 'Swagger Petstore' with version '1.0.0 OAS3' and the file path './examples/petstore.yaml'. Below the title, there's a license section mentioning 'MIT'. The main content area is divided into sections: 'Servers' (containing a dropdown menu with the URL 'http://petstore.swagger.io/v1'), 'pets' (listing three endpoints: GET /pets, POST /pets, and GET /pets/{petId}), and 'Schemas' (listing three schema definitions: Pet, Pets, and Error). The 'pets' section is currently expanded, showing its three sub-endpoints.

ReDoc Example

localhost:3200/redoc.html#operation/listPets

Search...

pets

List all pets

GET List all pets

POST Create a pet

GET Info for a specific pet

Documentation Powered by ReDoc

List all pets

QUERY PARAMETERS

limit integer <int32>
How many items to return at one time (max 100)

Responses

200 A paged array of pets

default unexpected error

Create a pet

Responses

201 Null response

default unexpected error

Info for a specific pet

GET /pets

Response samples

200 default

application/json

```
[  
  - {  
      "id": 0,  
      "name": "string",  
      "tag": "string"  
    }  
]
```

Copy Expand all Collapse all

POST /pets

Response samples

default

application/json

```
{  
  "code": 0,  
  "message": "string"  
}
```

Copy Expand all Collapse all

GET /pets/{petId}

Parts of an OpenAPI Document

```
openapi: "3.0.2"
info:
  version: 1.0.0
  title: Media Library
  description: |-  
    This is an example API for  
    [RESTfest](https://www.restfest.org).
  contact:
    name: Media Library Support
    url: https://example.com
    email: support@example.com
  license:
    name: MIT
    url: https://opensource.org/licenses/MIT
paths:
  /book:
    get:
      summary: Get a list of books
      operationId: getBooks
      responses:
        200:
          description: A list of books.
```

- Name your root document **openapi.json** or **openapi.yaml**
- Key properties:
 - **openapi**: The version of the OpenAPI specification you are using
 - **info**: Metadata about your API
 - **paths**: Describes all your endpoints and their request and response bodies

Tips, Tricks, & Gotchas

\$ref

```
paths:  
  /book:  
    get:  
      summary: Get a list of books  
      operationId: getBooks  
      responses:  
        200:  
          content:  
            application/json:  
              schema:  
                title: List of Books  
                type: array  
                items:  
                  $ref: "#/components/schemas/book"  
components:  
  schemas:  
    book:  
      title: Book  
      type: object  
      properties:  
        title:  
          title: Name of the book  
          type: string
```

Inheritance?

- Not exactly.
- JSON Schema provides boolean logic for subschemas.
 - anyOf
 - oneOf
 - allOf
- Some use allOf to provide a kind of inheritance, but it can lead to logical impossibilities.

Inheritance?

- Not exactly.
- JSON Schema provides boolean logic for subschemas.
 - anyOf
 - oneOf
 - allOf
- Some use allOf to provide a kind of inheritance, but it can lead to logical impossibilities.

anyOf

anyOf:

- **type: string**
maxLength: 5
- **type: number**
minimum: 0

Passes

“short”

Fails

“too long”

12

-5

Inheritance?

- Not exactly.
- JSON Schema provides boolean logic for subschemas.
 - anyOf
 - oneOf
 - allOf
- Some use allOf to provide a kind of inheritance, but it can lead to logical impossibilities.

oneOf

```
type: number
oneOf:
  - multipleOf: 5
  - multipleOf: 3
```

<u>Passes</u>	<u>Fails</u>
10	2
9	15

Inheritance?

- Not exactly. **allOf**
 - JSON Schema provides boolean logic for subschemas.
 - anyOf
 - oneOf
 - allOf
 - Some use allOf to provide a kind of inheritance, but it can lead to logical impossibilities.
- allOf:**
– `type: string`
– `maxLength: 5`
- | <u>Passes</u> | <u>Fails</u> |
|---------------|--------------|
| “short” | “too long” |

Inheritance?

- Not exactly. **allOf**
- JSON Schema provides boolean logic for subschemas.
 - anyOf
 - oneOf
 - allOf
- Some use allOf to provide a kind of inheritance, but it can lead to logical impossibilities.

allOf:	Fails	Fails
<code>- type: string</code>		
<code>- type: number</code>		
	<code>“a string”</code>	42

Recursion

```
components:  
  schemas:  
    book:  
      title: Book  
      type: object  
      properties:  
        title:  
          title: Name of the book  
          type: string  
      relatedBooks:  
        type: array  
        items:  
          $ref: "#/components/schemas/book"
```

Custom Properties

- OpenAPI allows extension through x-properties.
- Example:

x-beta: true

- Custom properties have no meaning to tools and documentation-generators, unless you write your own logic for them.

Let's Build an API!

Tools

- Documentation:
 - Swagger UI: swagger.io/tools/swagger-ui/
 - ReDoc: github.com/Rebilly/ReDoc
 - Stoplight Studio: <https://stoplight.io/studio/>

Tools

- Much more:
 - Validators
 - Testing tools
 - Parsers
 - Mock Servers
- Check out [openapi.tools](#) by Matt Trask and Phil Sturgeon

Security

By poking around your OpenAPI specification, some tools can look out for attack vectors you might not have noticed.

Name	Language	v2	v3	GitHub
API Contract Security Audit - Upload OpenAPI file, get detailed report with located vulnerabilities, possible attack scenarios, ways to remediate.	SaaS	✓	✗	

Converters

Various tools to convert to and from OpenAPI standards. Useful for working with various API formats.

Name	Language	v2	v3	GitHub
Apimatic Transformer - Transform API Descriptions to and from RAML, API Blueprint, OAI v2/v3, WSDL, etc.	SaaS	✓	✓	
API Flow - Convert from and to multiple formats. Linking to Phil's fork because the original is completely broken	JavaScript	✓	⚠️	
Google Gnostic - Compile OpenAPI descriptions into equivalent Protocol Buffer representations	Go	✓	✓	
swagger2openapi - Upgrade specs from OpenAPI v2.0 to v3.0, bundling into one mega file or respecting \$refs	Node.js	✓	✓	
OAS RAML Converter - Converts between OAS and RAML API specifications	Node.js	✓	✓	
OData OpenAPI - OData 4.0 to OpenAPI 3.0.0 converter	XSLT	✓	✓	
OpenAPI Filter - Filter internal components from OpenAPI definitions	Node.js	✓	✓	
OData.OpenAPI - Convert an Edm (Entity Data Model) to OpenApi 3.0	.NET	✗	✓	
pyswagger - Client & converter in Python, which is type-safe, dynamic, spec-compliant.	Python	✓	⚠️	
odata2openapi - OData 4.0 to OpenAPI 2 converter	Node.js	✓	✗	
avantation - Generate OpenAPI 3.x specification from HAR.	TypeScript	✗	✓	
OpenAPI Generator - A template-driven engine to generate documentation, API clients	Java	✓	✓	

[openapi-viewer](#) - Browse and test a REST API described with the OpenAPI 3.0 Specification

Vue.js

[openapi-ui](#) - React based OpenAPI 3.0+ documentation generator

React.js

[ReDoc](#) - OpenAPI/Swagger-generated API Reference Documentation

React.js

[widdershins](#) - Generate Slate/Shins markdown from OpenAPI 2.0/3.0.x

Node.js

[openapi3-generator](#) - Use your API OpenAPI 3 definition to generate code, documentation, and literally anything you need.

Node.js

[MrinDoc](#) - Open API spec viewer.

Vue.js

[RapiDoc](#) - Custom Element to view OpenAPI spec.

Custom Element

[RapiPdf](#) - Custom Element to generate PDF from OpenAPI spec.

Custom Element

[Stoplight](#) - Create beautiful, customizable, interactive API documentation generated from your OpenAPI Specification, integrated with the Stoplight platform

SaaS

[Bump](#) - Bump generates elegant documentations and changelogs from your OpenAPI specifications. Git diff, for your API.

SaaS

Text Editors

Text editors to help build API docs.

Name	Language	v2	v3	GitHub
KaiZen-OpenAPI-Editor - Full-featured Eclipse editor for OpenAPI 2.0 and 3.0, also available on Eclipse Marketplace.	Java	✓	✓	
Atom/linter-swagger - This plugin for Atom Linter will lint Swagger 2.0 specifications or OpenAPI 3.0, both JSON and YAML using swagger-parser node package.	JavaScript	✓	✓	
Swagger Editor - Design, describe, and document your API on the first open source editor fully dedicated to OpenAPI-based APIs.	Node.js	✓	✓	
SwaggerHub - API design and documentation platform to improve collaboration, standardize development workflow and centralize their API discovery and consumption.	SaaS/On-Premise NodeJS	✓	✓	

Resources

- OpenAPI Initiative: <https://www.openapis.org> (*note the “s”*)
- OpenAPI Specification: <http://spec.openapis.org/oas/v3.0.2>
- Repository: <https://github.com/OAI/openapi-specification/>

Thank you!

benramsey.com

ben@benramsey.com

[@ramsey on GitHub](https://github.com/ramsey)

[@ramsey on Twitter](https://twitter.com/ramsey)

[@ramsey@phpc.social on Mastodon](https://mastodon.phpc.social/@ramsey)

