Swagger 3.0

Basic Structure

You can write OpenAPI definitions in [YAML](https://en.wikipedia.org/wiki/YAML) or [JSON](https://en.wikipedia.org/wiki/JSON). A sample OpenAPI 3.0 definition written in YAML looks like:

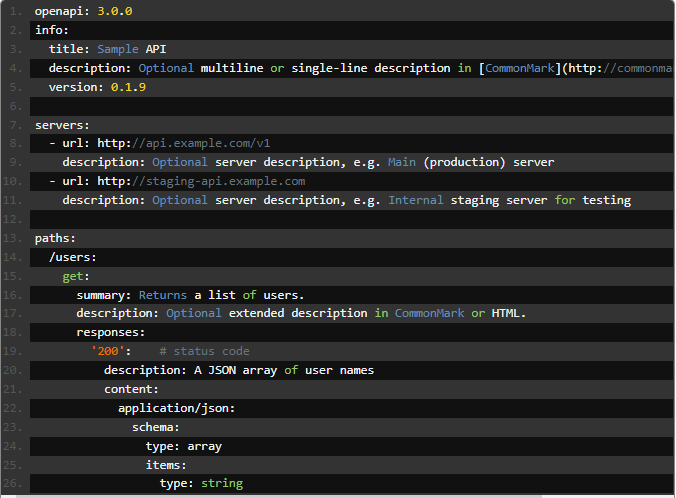


Fig 1: Sample OpenAPI 3.0 Definition

Metadata

Every API definition must include the version of the OpenAPI Specification that this definition is based on. The OpenAPI version defines the overall structure of an API definition – what you can document and how you document it. OpenAPI 3.0 uses [semantic versioning](http://semver.org/) with a three-part version number. The [available versions](https://github.com/OAI/OpenAPI-Specification/releases) are 3.0.0, 3.0.1, 3.0.2, and 3.0.3; they are functionally the same. The info section contains API information: title, description (optional), version.

### Servers

### The servers section specifies the API server and base URL. You can define one or several servers. All API paths are relative to the server URL.

Paths

The paths section defines individual endpoints (paths) in your API, and the HTTP methods (operations) supported by these endpoints. For example, GET /users can be described as:

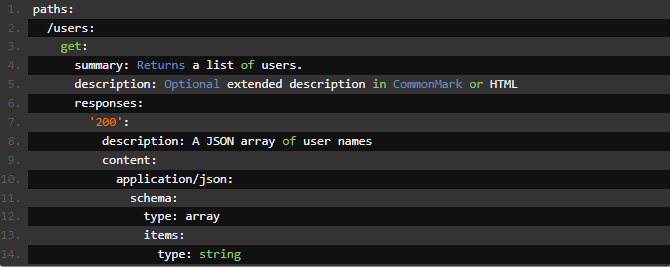


Fig 2: Path example

Parameters

Operations can have parameters passed via URL path (/users/{userId}), query string (/users?role=admin), headers (X-CustomHeader: Value) or cookies (Cookie: debug=0). You can define the parameter data types, format, whether they are required or optional, and other details.

### Request Body

### 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. OpenAPI 3.0 provides the requestBody keyword to describe request bodies.

### Responses

### An API specification 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.

### Inputs and Output Models

### The global components/schemas section lets you define common data structures used in your API. They can be referenced via $ref whenever a schema is required – in parameters, request bodies, and response bodies.

### 

Fig 3: A JSON Object

The JSON object in fig 3 can represented as:



Fig 4: Defining schema

And then referenced in the request body schema and response body schema as follows:

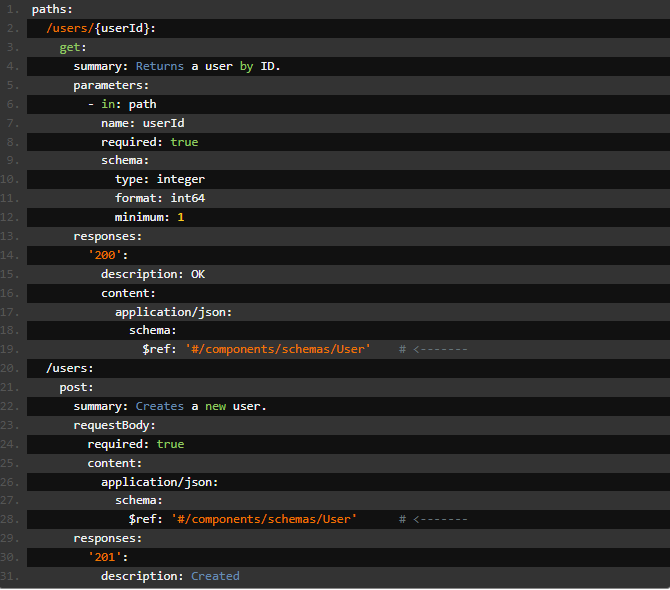


Fig 5: Referencing the schema

Authentication

OpenAPI uses the term **security scheme** for authentication and authorization schemes. OpenAPI 3.0 lets you describe APIs protected using the following security schemes:

* HTTP authentication schemes (they use the Authorization header):
  + [Basic](https://swagger.io/docs/specification/authentication/basic-authentication/)
  + [Bearer](https://swagger.io/docs/specification/authentication/bearer-authentication/)
  + other HTTP schemes as defined by [RFC 7235](https://tools.ietf.org/html/rfc7235) and [HTTP Authentication Scheme Registry](https://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml)
* [API keys](https://swagger.io/docs/specification/authentication/api-keys/) in headers, query string or cookies
  + [Cookie authentication](https://swagger.io/docs/specification/authentication/cookie-authentication/)
* [OAuth 2](https://swagger.io/docs/specification/authentication/oauth2/)
* [OpenID Connect Discovery](https://swagger.io/docs/specification/authentication/openid-connect-discovery/)