

- [SwaggerHub](#)
- [Swagger Inspector](#)
- [Open Source Tools](#)
- [Specification](#)
 - [What Is OpenAPI?](#)
 - [Basic Structure](#)
 - [API Server and Base Path](#)
 - [Media Types](#)
 - [Paths and Operations](#)
 - [Describing Parameters](#)
 - [Parameter Serialization](#)
 - [Describing Request Body](#)
 - [Describing Responses](#)
 - [Data Models \(Schemas\)](#)
 - [Adding Examples](#)
- [Authentication](#)
 - [Basic Authentication](#)
 - > [API Keys](#)
 - [Bearer Authentication](#)
 - [OAuth 2.0](#)
 - [OpenID Connect](#)
 - [Discovery](#)
 - [Cookie Authentication](#)
- [Links](#)
- [Callbacks](#)
- [Components Section](#)
- [Using \\$ref](#)
- [API General Info](#)
- [Grouping Operations](#)
- [With Tags](#)
- [OpenAPI Extensions](#)
- [2.0](#)



On-Demand Webinar
Building and Enforcing
API Standards at Scale
with Swagger Tools



WATCH NOW

OAS 3 This page applies to OpenAPI 3 – the latest version of the OpenAPI Specification. If you use OpenAPI 2 (fka Swagger), visit [OpenAPI 2 pages](#).

API Keys

Some APIs use API keys for authorization. An API key is a token that a client provides when making API calls. The key can be sent in the query string:

```
1. GET /something?api_key=abcdef12345
```

or as a request header:

```
1. GET /something HTTP/1.1
2. X-API-Key: abcdef12345
```

or as a [cookie](#):

```
1. GET /something HTTP/1.1
2. Cookie: X-API-KEY=abcdef12345
```

API keys are supposed to be a secret that only the client and server know. Like [Basic authentication](#), API key-based authentication is only considered secure if used together with other security mechanisms such as HTTPS/SSL.

Describing API Keys

In OpenAPI 3.0, API keys are described as follows:

```
1. openapi: 3.0.0
2. ...
3.
4. # 1) Define the key name and location
5. components:
6.   securitySchemes:
7.     ApiKeyAuth # arbitrary name for the security scheme
8.       type: apiKey
9.       in: header # can be "header", "query" or "cookie"
10.      name: X-API-KEY # name of the header, query parameter or cookie
11.
12. # 2) Apply the API key globally to all operations
13. security:
14.   - ApiKeyAuth: [] # use the same name as under securitySchemes
```

This example defines an API key named **X-API-Key** sent as a request header **X-API-Key: <key>**. The key name *ApiKeyAuth* is an arbitrary name for the security scheme (not to be confused with the API key name, which is specified by the **name** key). The name *ApiKeyAuth* is used again in the **security** section to apply this security scheme to the API. **Note:** The **securitySchemes** section alone is not enough; you must also use **security** for the API key to have effect. **security** can also be set on the operation level instead of globally. This is useful if just a subset of the operations need the API key:

```
1. paths:
2.   /something:
3.     get:
4.       # Operation-specific security:
5.       security:
6.         - ApiKeyAuth: []
7.     responses:
8.       '200':
9.         description: OK (successfully authenticated)
```

Multiple API Keys

Some APIs use a pair of security keys, say, API Key and App ID. To specify that the keys are used together (as in logical AND), list them in the same array item in the **security** array:

```
1. components:
2.   securitySchemes:
3.     apiKey:
4.       type: apiKey
5.       in: header
6.       name: X-API-KEY
7.     appId:
8.       type: apiKey
9.       in: header
10.      name: X-APP-ID
11.
12. security:
13.   - apiKey: []
14.     appId: [] # <-- no leading dash (-)
```

Note the difference from:

```
1. security:
2.   - apiKey: []
3.   - appId: []
```

which means either key can be used (as in logical OR). For more examples, see [Using Multiple Authentication Types](#).

401 Response

You can define the 401 “Unauthorized” response returned for requests with missing or invalid API key. This response includes the **WWW-Authenticate** header, which you may want to mention. As with other common responses, the 401 response can be defined in the global **components/responses** section and referenced elsewhere via **\$ref**.

```
1. paths:
2.   /something:
3.     get:
4.       ...
5.     responses:
6.       ...
7.       '401':
8.         $ref: "#/components/responses/UnauthorizedError"
9.     post:
10.      ...
11.    responses:
12.      ...
13.      '401':
14.        $ref: "#/components/responses/UnauthorizedError"
15.
16. components:
17.   responses:
18.     UnauthorizedError:
19.       description: API key is missing or invalid
20.       headers:
21.         WWW_Authenticate
22.       schema:
23.         type: string
```

To learn more about describing responses, see [Describing Responses](#).



Swagger Open Source

- [Open Source License](#)
- [Swagger Forum](#)
- [Swagger IRC](#)
- [Swagger Community](#)
- [Swagger Projects](#)


Swagger

- [About Swagger](#)
- [Blog](#)
- [Support](#)
- [News](#)
- [Contact Us](#)

Pro Tools

- [SwaggerHub](#)
- [Swagger Inspector](#)
- [SwaggerHub Enterprise](#)
- [SwaggerHub vs OSS](#)
- [SwaggerHub Integrations](#)

Resources

- [OpenAPI Specification](#)
- [Resources](#)
- [Open Source Docs](#)
- [Swagger Inspector Docs](#)
- [SwaggerHub Docs](#) 

SMARTBEAR

Products

Over 6 million developers, testers and operations teams use SmartBear products to deliver the world's best applications.

[Explore SmartBear Products](#) >

Company

- [About SmartBear](#)
- [Careers](#)
- [Newsroom](#)
- [Upcoming Events](#)

Global Headquarters

450 Artisan Way
Somerville, MA 02145

[+1 \(617\) 684-2600](#)

Info@SmartBear.com

