API

An API, or application programming interface, is a set of defined rules that enable different applications to communicate with each other

It acts as an intermediary layer that processes data transfers between systems, letting companies open their application data and functionality to external third-party developers, business partners, and internal departments within their companies.

The definitions and protocols within an API help businesses connect the many different applications they use in day-to-day operations, which saves employees time and breaks down silos that hinder collaboration and innovation.

How an API works

A simple way to understand how APIs work is to look at a common example—third-party payment processing

This function relies on APIs to make the connection.

The request is processed from an application to the web server via the API’s Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.

• After receiving a valid request from the product webpage, the API makes a call to the external program or web server, in this case, the third-party payment system.

• The server sends a response to the API with the requested information.

• The API transfers the data to the initial requesting application, here the product website.

• While the data transfer will differ depending on the web service being used, the requests and responses all happen through an API.

API benefits

• Improved collaboration.

APIs enable integration so the platforms and apps can seamlessly communicate with one another.

• Accelerated innovation.

• Data monetization

Many companies choose to offer APIs for free, at least initially, so that they can build an audience of developers around their brand and forge relationships with potential business partners

• System security.

APIs separate the requesting application from the infrastructure of the responding service, and offer layers of security between the two as they communicate.

• End-user security and privacy

Just as APIs provide added protection within a network, they can also provide another layer of protection for personal users

APIs common examples

• Universal logins

A popular API example is the function that enables people to log in to websites by using their Facebook, Twitter, or Google profile login details

• Internet of Things (IoT)

These “smart devices” offer added functionality, such as internet-enabled touchscreens and data collection, through APIs.

• Travel booking

Travel booking sites aggregate thousands of flights, showcasing the cheapest options for every date and destination.

• Mapping apps

In addition to the core APIs that display static or interactive maps, these apps use other APIs and features to provide users with directions

• SaaS applications

APIs are an integral part of the growth in software-as-a-service (SaaS) products

Types of APIs

• Open APIs are open-source application programming interfaces you can access with the HTTP protocol.

• Partner APIs connect strategic business partner

• Partner APIs connect strategic business partners. Typically, developers access these APIs in self-service mode through a public API developer portal.

• Internal APIs remain hidden from external users. These private APIs aren't available for users outside of the company and are instead intended to improve productivity and communication across different internal development teams

• Composite APIs combine multiple data or service APIs. They allow programmers to access several endpoints in a single call.

Example : API common example

API protocols

• SOAP (Simple Object Access Protocol): Built with XML, SOAP enables endpoints to send and receive data through SMTP and HTTP. SOAP APIs make it easier to share information between apps or software components that are running in different environments or written in different languages.

SMTP, or Simple Mail Transfer Protocol, is a fundamental technology that powers the process of sending emails over the internet.

HTTP Request / Response

Communication between clients and servers is done by requests and responses:

1. A client (a browser) sends an HTTP request to the web

2. A web server receives the request

3. The server runs an application to process the request

4. The server returns an HTTP response (output) to the browser

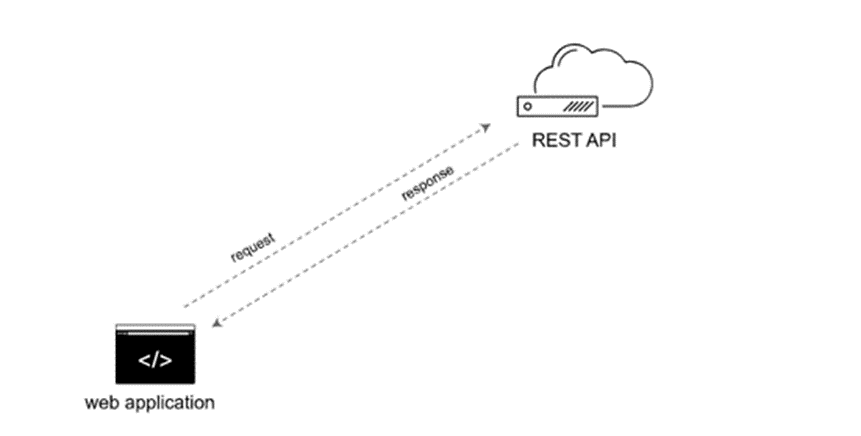
5. The client (the browser) receives the response

• XML-RPC (XML-Remote Procedure Call): The XML-RPC protocol relies on a specific XML format to transfer data. XML-RPC is older than SOAP, but much simpler, and relatively lightweight in that it uses minimum bandwidth.

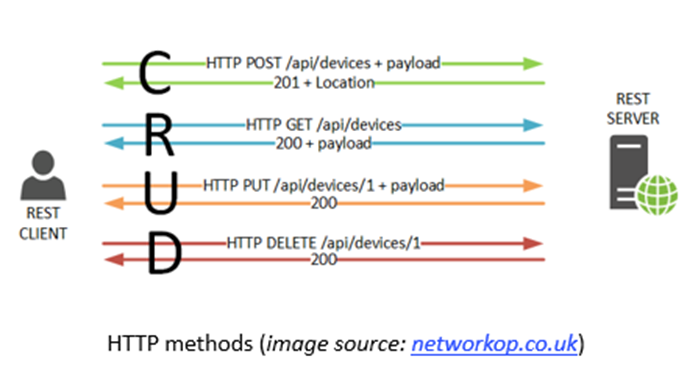
• JSON-RPC: Like XML-RPC, JSON-RPC is a remote procedure call, but JSON (JavaScript Object Notation) is used instead of XML to transfer the data.

• REST (Representational State Transfer): REST is a set of web API architecture principles. REST APIs—also known as a RESTful API)—are APIs that adhere to certain REST architectural constraints. It’s possible to build RESTful APIs with SOAP protocols, but the two standards are usually viewed as competing specifications.

REST APIs







REST APIs involve requests and responses, not too unlike visiting a web page. You make a request to a resource stored on a server, and the server responds with the requested information. The protocol used to transport the data is HTTP. “REST” stands for Representational State Transfer.

• Open up software’s data and functionality to other internal and external developers.

• Provides building blocks to help build custom applications and functionalities.

• Makes software interactions developer-friendly, easily accessible, and secure by use of standard protocols

Web Services APIs

REST, SOAP, XML, JSON

Library-based APIs

JavaScript, TWAIN

Class-based APIs

Java API, Android API

Object Remoting APIs

COBRA, .NET Remoting

Hardware APIs

Video acceleration, PCI buses

Rest

REST is an architectural style which is protocol independent

Constraints define that RESTful system:

* Client-Server: Client and the server should be separate from each other
* Stateless: Agnostic of the state of one another. Each request contains all of the data necessary to complete itself successfully
* Cache: Designed to encourage the storage of cacheable data.
* Uniform Interface: Simplifies and decouples the architecture, which enables each part to evolve independently.
* Layered System: Helps create a more scalable and modular application
* Code on Demand: Allows for code or applets to be transmitted via the API

Restful APIs

APIs based on REST architecture are known as restful API

HTTP-based RESTful APIs are defined with the following aspects:

• Base URL: such as

http://<host>/urest/v1/<resource\_name>

• An internet media type (MIME type) (that defines state transition data elements. Resources are represented via media types such as JSON, XML, HTML, etc.

• Standard Uniform interface. In the case of HTTP, this consists if Standard HTTP method

CRUD

• POST: Create resource

• GET: Read/ retrieve resource details

• PUT/PATCH : Update resource details

• DELETE : Delete resource

Advantage of Restful API

* Interoperability: Exchange of data using JSON over HTTP protocols, which are language and platform independent.
* UI independent – Direct interaction with the backend data independent of the UI
* Testability – Increases the depth and scope of tests and helps you identify bugs at early stages of application development.
* Time and cost saving – Requires less bandwidth and resources. Simpler data structures provide faster test coverage and are more cost effective than automated GUI testing.
* Security – ITOM APIs use X-Auth protocol for authentication and TLS for secured communication over HTTPS

JSON - JAVA Script Object Notation

JOSN was originally created to hold structured data to be used in JavaScript. It is now the most popular way of sending data for Web APIs for all types of applications.

• Basic types: string, number, Boolean, null.

• Arrays (lists) consist of comma-separated lists(if we need some information from the list)

• Use square brackets [ ] (can be numbred string, bolean)

• Can mix data types



Objects

Objects (dictionaries) consist of keys and values:

Use curly brackets { }

Keys and values can be any data type

Keys and values are separated by a colon :

Authentication and Authorization

• A user provides valid username and password

• This typically results in the return of an access token

Authorization

• When you make an API request, you send the server an access token

• This token determines what API requests you can make

OAuth is a popular open protocol:

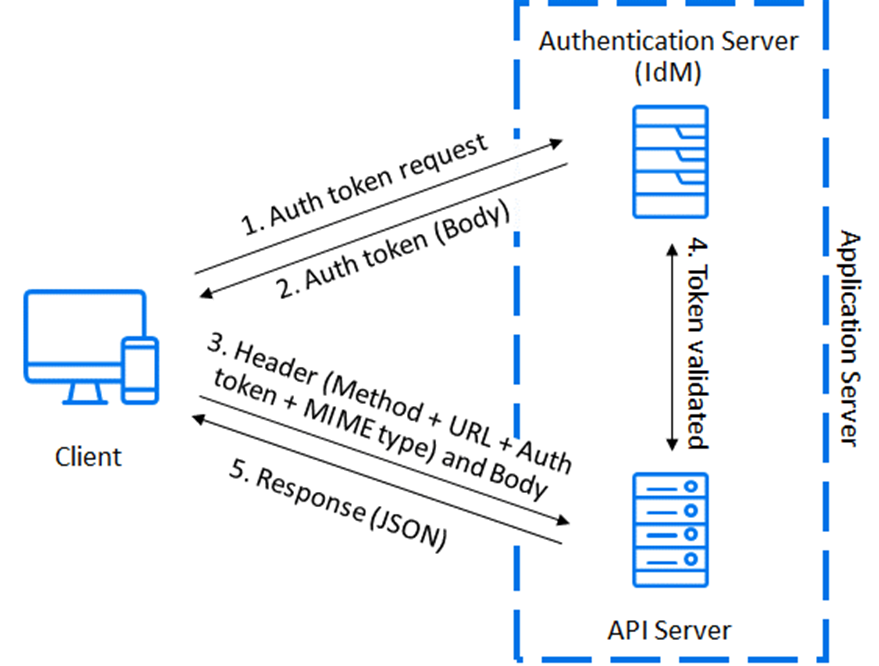
• Each API user has an access token.

• The token tells the server what the API user has access to.

OAuth uses two most common ways to handle authorization:

• One-legged: For user data

• Three-legged: For protected user data



• Payload

• Response

Method and URL

POST: Create resource

GET: Read/retrieve resource details

PUT/PATCH: Update resource details

DELETE: Delete resource

URL stands for "Uniform Resource Locator“.

• Also known as web address

• Can start with http or https

• Includes server and resource information

Query Parameters

• Query parameters are key value pairs that contain additional data you can send to the server.

• They are the part of the URL after the question mark

• Consists of parameters ( key ), operators (LIKE, EQ, NE), and attributes (values).

• Typically used to modify or filter the data that is returned.

• Key and value is separated by equals sign.

• Key/value pairs are separated by ampersand.

Headers

• Headers are a standard part of HTTP; contain key value pairs.

• Typically used to specify data format and for authorization.

• It is also used for browser cookies.

• You can use a different format when sending vs. receiving data

Payload

• Usually JSON or XML

• Can be a media file (image, video, sound file)

• Only have a payload for POST, PUT, and PATCH; not GET or DELETE

Response

• Response contains the requested data from the server.

• In case of error, might contains the error details.

Status/Error codes

Errors are returned by one of two ways:

• HTTP status codes

• Elements in the response body

o HTTP status codes are part of the HTTP protocol

o A status code is always returned with the response

They were designed for web pages, but have been repurposed for APIs.

HTTP status code series: 1xx: (100) Informational responses

2xx: Success

3xx: Redirection

4xx: Client errors

5xx: Server errors

Parameters

Parameters are options you can pass with the endpoint to influence the response

There are four types of parameters: header parameters, path parameters, query string parameters, and request body parameters

Header parameters: Parameters included in the request header, usually related to authorization.

Path parameters: Parameters within the path of the endpoint, before the query string (?). These are usually set off within curly braces.

Query string parameters: Parameters in the query string of the endpoint, after the ?.

Request body parameters: Parameters included in the request body. Usually submitted as JSON.

https://api.openweathermap.org/surfreport{beachId}).

Regardless of the parameter type, define the following with each parameter:

Data type

Max and min value

Data types for parameters

Response example and schema

The response example shows a sample response from the request example; the response schema defines all possible elements in the response.

The description of the response is known as the response schema

Identify URL

• The base path (or base URL or host) refers to the common path for the API.

http://apiserver.com.

API Server: Build a REST API from your DB with a few clicks

http://apiserver.com

• The endpoint refers to the end path of the endpoint

, /homes

part of the endpoint contains query string parameters for the endpoint. Numeric, alpha numeric

?limit=5&format=json

• Classes will interact with each other with one another classes with method statements using parameters or data

Authentication and Authorization

Authentication

• A user provides valid username and password

• This typically results in the return of an access token

Authorization

• When you make an API request, you send the server an access token

• This token determines what API requests you can make

OAuth is a popular open protocol:

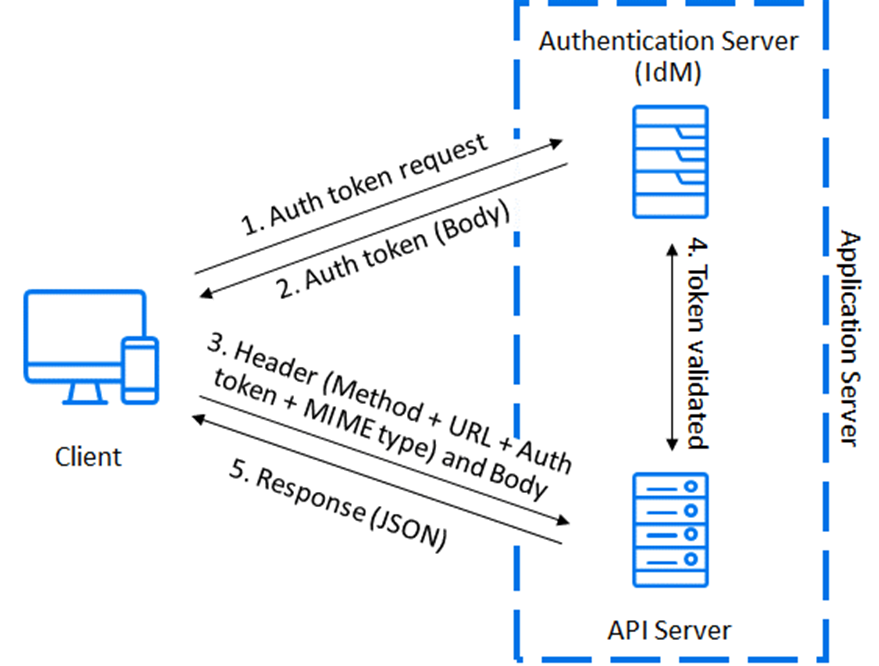
• Each API user has an access token.

• The token tells the server what the API user has access to.

OAuth uses two most common ways to handle authorization:

• One-legged: For user data

• Three-legged: For protected user data



HTTP Requests

HTTP requests are used to make API calls from the client to the server.

A typical HTTP request has following parts:

• Method and URL (Base URL + Resource)

• Query parameter (URL parameter)

• Header

• Payload

• Response

Method and URL

POST: Create resource

GET: Read/retrieve resource details

PUT/PATCH: Update resource details

DELETE: Delete resource

URL stands for "Uniform Resource Locator“.

• Also known as web address

• Can start with http or https

• Includes server and resource information

Query Parameters

• Query parameters are key value pairs that contain additional data you can send to the server.

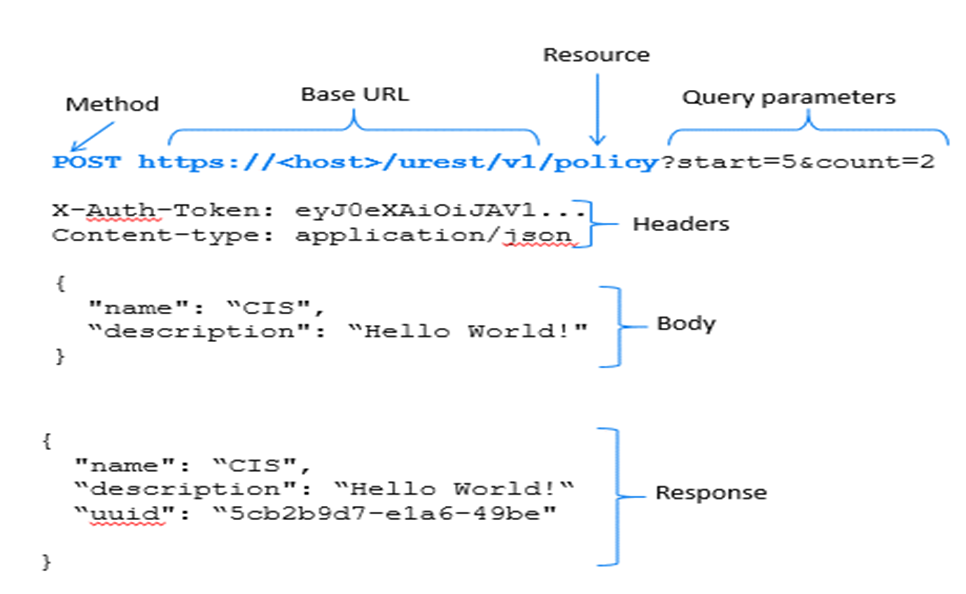
• They are the part of the URL after the question mark

• Consists of parameters ( key ), operatos (LIKE, EQ, NE), and attributes (values).

• Typically used to modify or filter the data that is returned.

• Key and value is separated by equals sign.

• Key/value pairs are separated by ampersand.



Headers

• Headers are a standard part of HTTP; contain key value pairs.

• Typically used to specify data format and for authorization.

• It is also used for browser cookies.

• You can use a different format when sending vs. receiving data

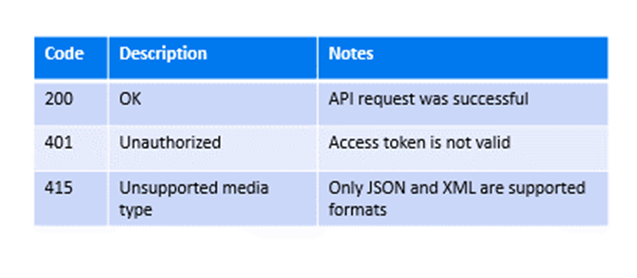
Payload

• Usually JSON or XML

• Can be a media file (image, video, sound file)

• Only have a payload for POST, PUT, and PATCH; not GET or DELETE

Response



• Response contains the requested data from the server.

• In case of error, might contains the error details.

Status/Error codes

Errors are returned by one of two ways:

• HTTP status codes

• Elements in the response body

o HTTP status codes are part of the HTTP protocol

o A status code is always returned with the response

They were designed for web pages, but have been repurposed for APIs.

HTTP status code series: 1xx: (100) Informational responses

2xx: Success

3xx: Redirection

4xx: Client errors

5xx: Server errors

Parameters

Parameters are options you can pass with the endpoint to influence the response

There are four types of parameters: header parameters, path parameters, query string parameters, and request body parameters

Header parameters: Parameters included in the request header, usually related to authorization.

Path parameters: Parameters within the path of the endpoint, before the query string (?). These are usually set off within curly braces.

Query string parameters: Parameters in the query string of the endpoint, after the ?.

Request body parameters: Parameters included in the request body. Usually submitted as JSON.

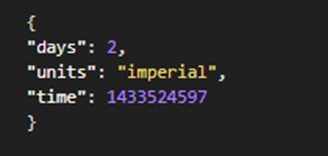
https://api.openweathermap.org/surfreport{beachId}).

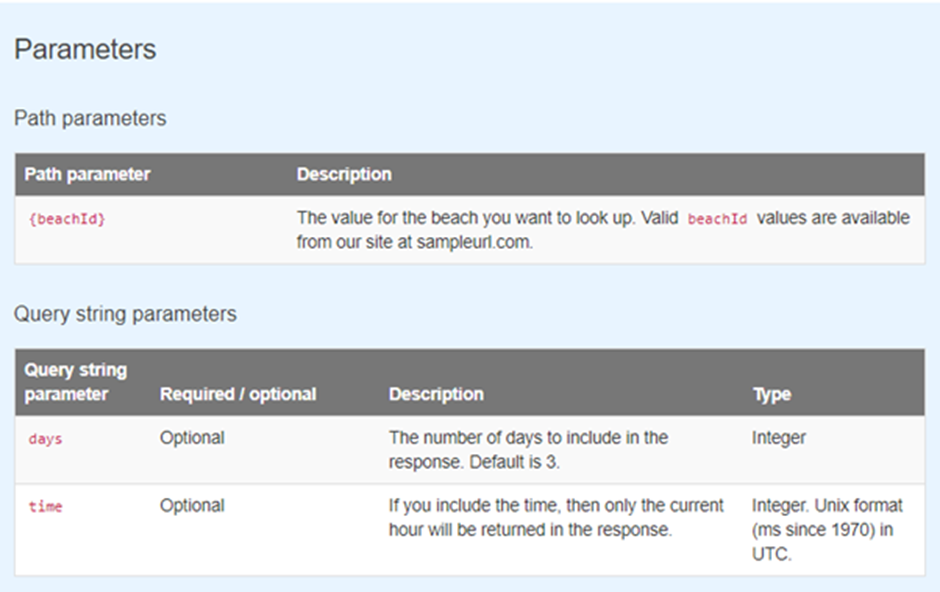
Regardless of the parameter type, define the following with each parameter:

Data type

Max and min value

Data types for parameters

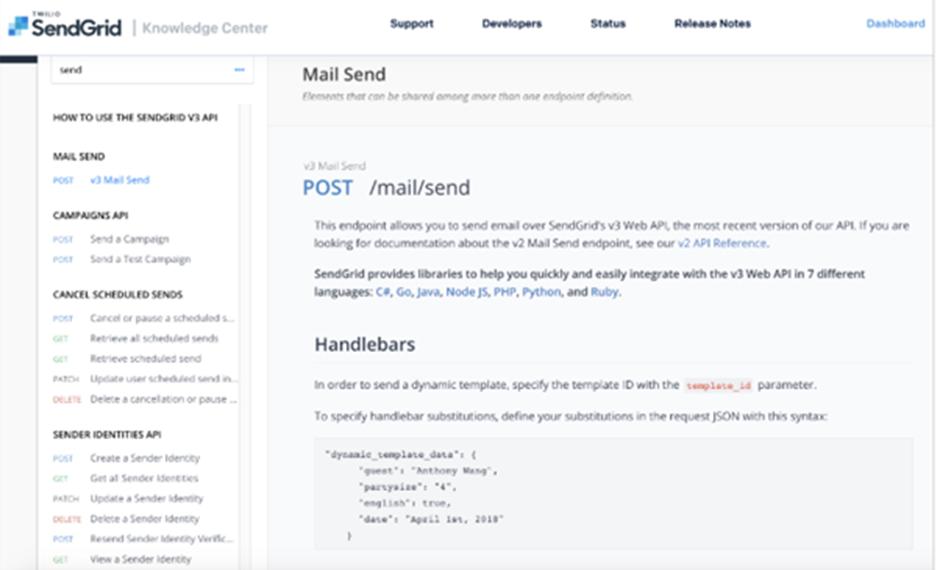


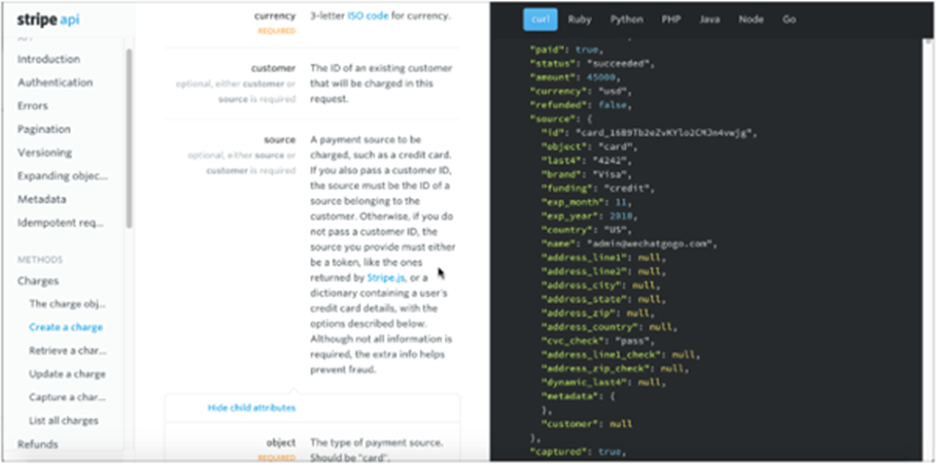


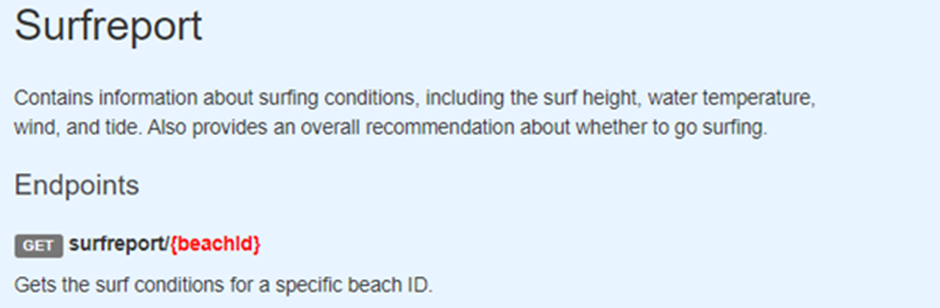
Response example and schema

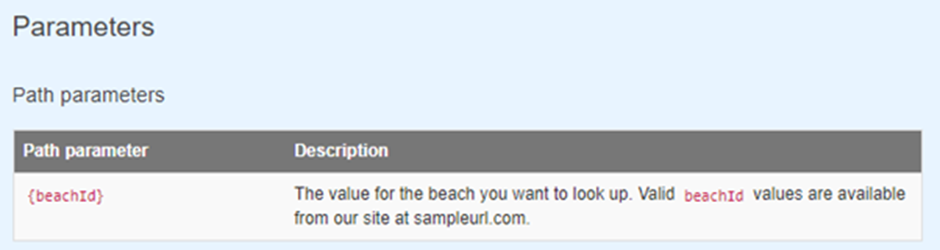
The response example shows a sample response from the request example; the response schema defines all possible elements in the response.

The description of the response is known as the response schema











 Identify URL

• The base path (or base URL or host) refers to the common path for the API.



<http://apiserver.com>

<http://apiserver.com>

The endpoint refers to the end path of the endpoint

, /homes

part of the endpoint contains query string parameters for the endpoint. Numeric, alpha numeric

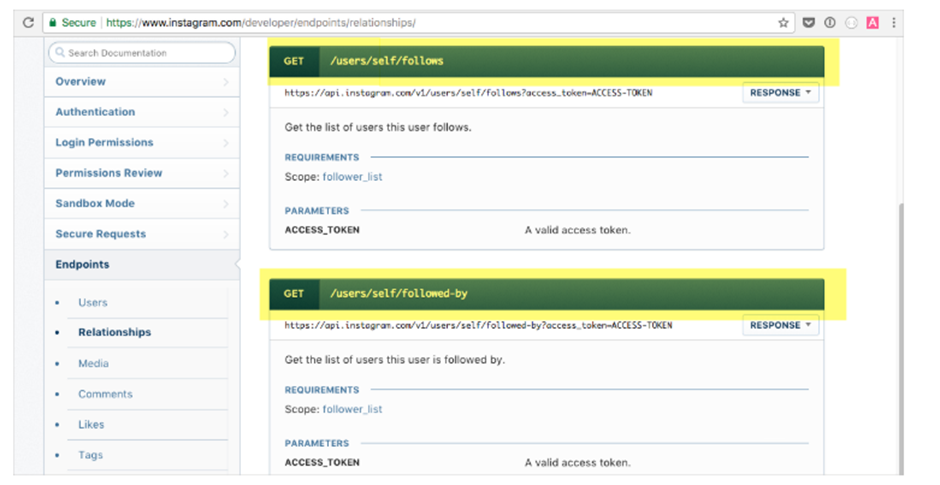
?limit=5&format=json

• Classes will interact with each other with one another classes with method statements using parameters or data



Endpoints and methods

The endpoints indicate how you access the resource, while the method indicates the allowed interactions (such as GET, POST, or DELETE) with the resource.



Four types of parameters

REST APIs have four types of parameters:

• Header parameters: Parameters included in the request header, usually related to authorization.

• Path parameters: Parameters within the path of the endpoint, before the query string (?). These are usually set off within curly braces.

• Query string parameters: Parameters in the query string of the endpoint, after the ?.

• Request body parameters: Parameters included in the request body. Usually submitted as JSON.

Regardless of the parameter type, define the following with each parameter:

• Data type

• Max and min value

• Data types for parameters

These data types are the most common with REST APIs:

• string: An alphanumeric sequence of letters and/or numbers

• integer: A whole number — can be positive or negative

• boolean: True or false value

• object: Key-value pairs in JSON format

• array: A list of values

Max and min values for parameters

In addition to specifying the data type, the parameters should indicate the maximum, minimum, and allowed values if appropriate

Not every parameter needs max and min values, however. Note these exceptions:

• Booleans: With Booleans, the only options are true or false, so there’s no need for max/min values.

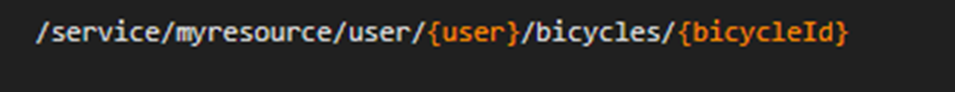
• Strings that use enums: If a string restricts possible values to enums (an enumerated list), the max/min values wouldn’t be appropriate

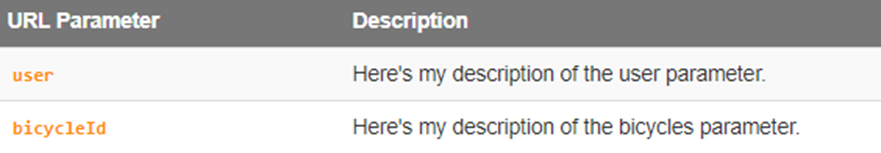
Header parameters

Header parameters are included in the request header. Usually, the header just includes authorization parameters that are common across all endpoints; as a result, the header parameters aren’t usually documented with each endpoint. Instead, the authorization details in header parameters are documented in the authorization requirements section.

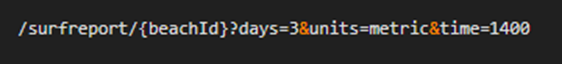
Path parameters

Path parameters are part of the endpoint itself and are not optional

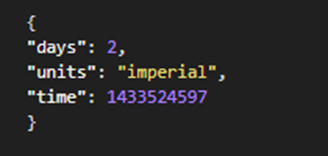




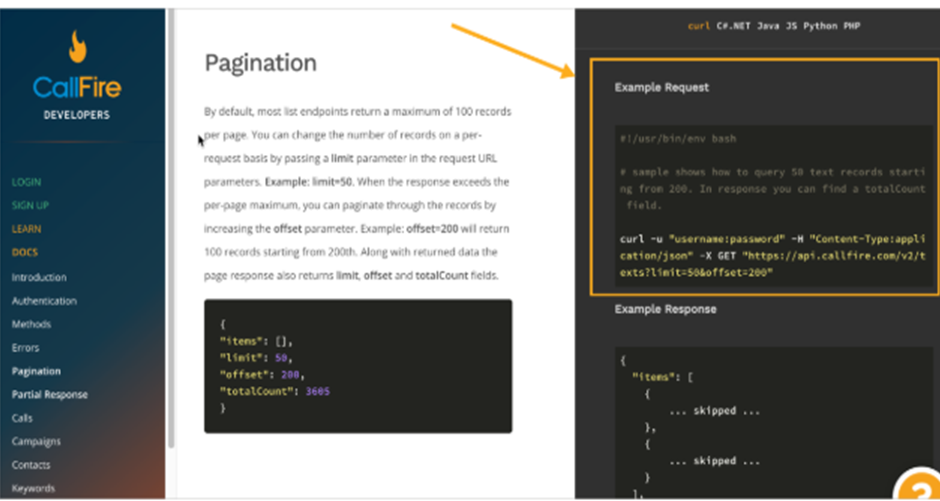
Query string parametersQuery string parameters appear after a question mark (?) in the endpoint. The question mark followed by the parameters and their values is referred to as the “query string.” In the query string, each parameter is listed oe right after the other with an ampersand ( & ) separating them. The order of the query string parameters does not matter.



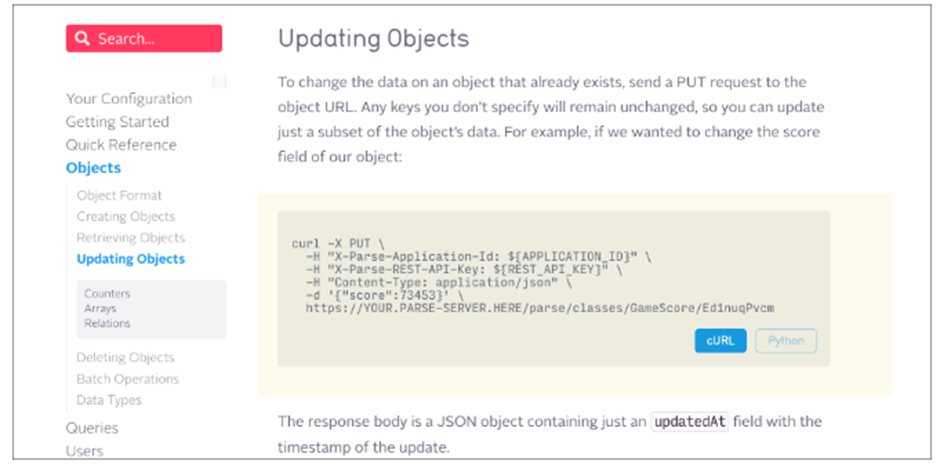
Request body parametersFrequently, with POST requests , you submit a JSON object in the request body. This is known as a request body parameter, and the format is usually JSON.

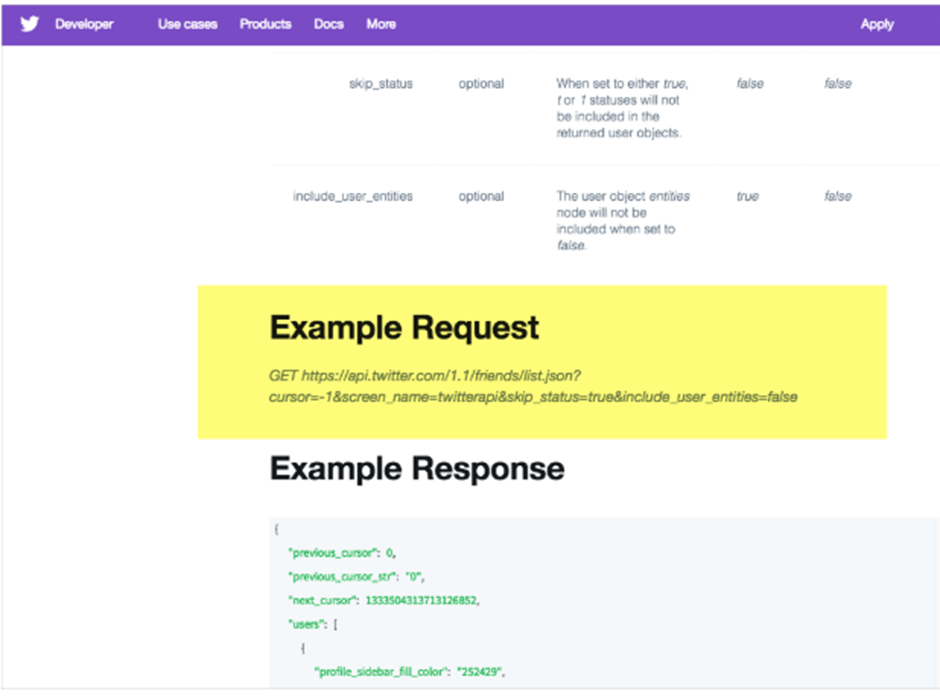


Request exampleThe request example includes a sample request using the endpoint, showing some parameters configured.



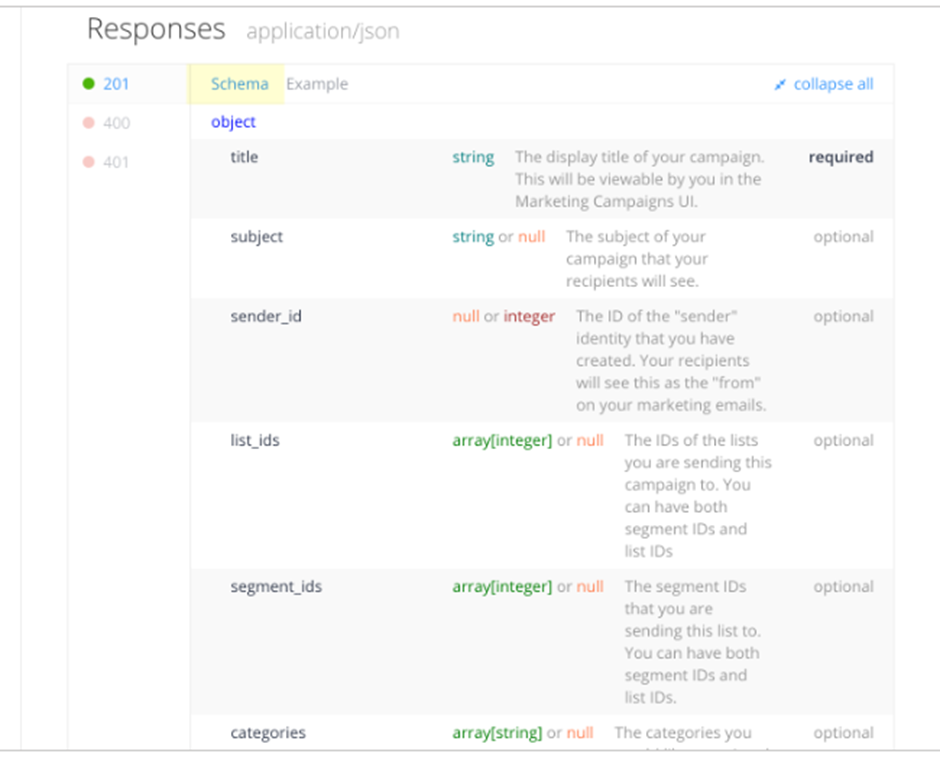


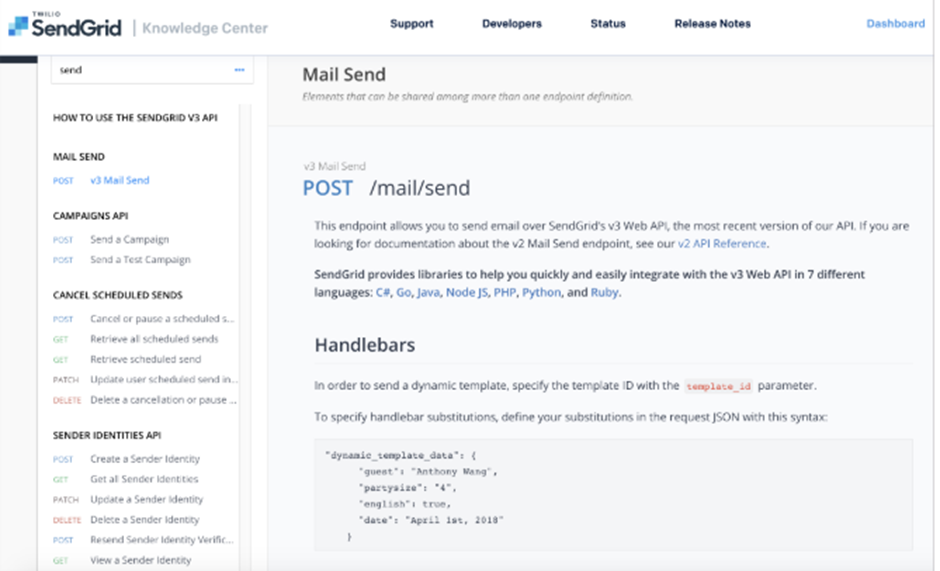




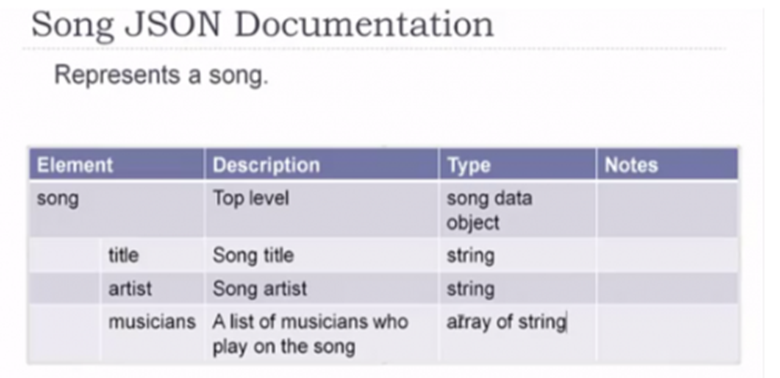


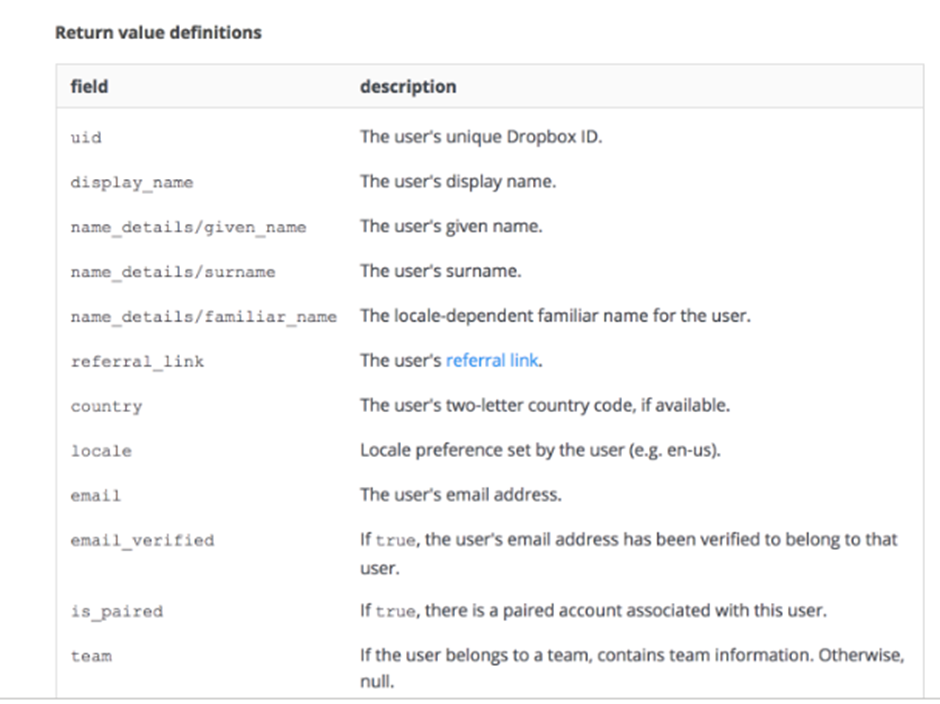
Response example and schema

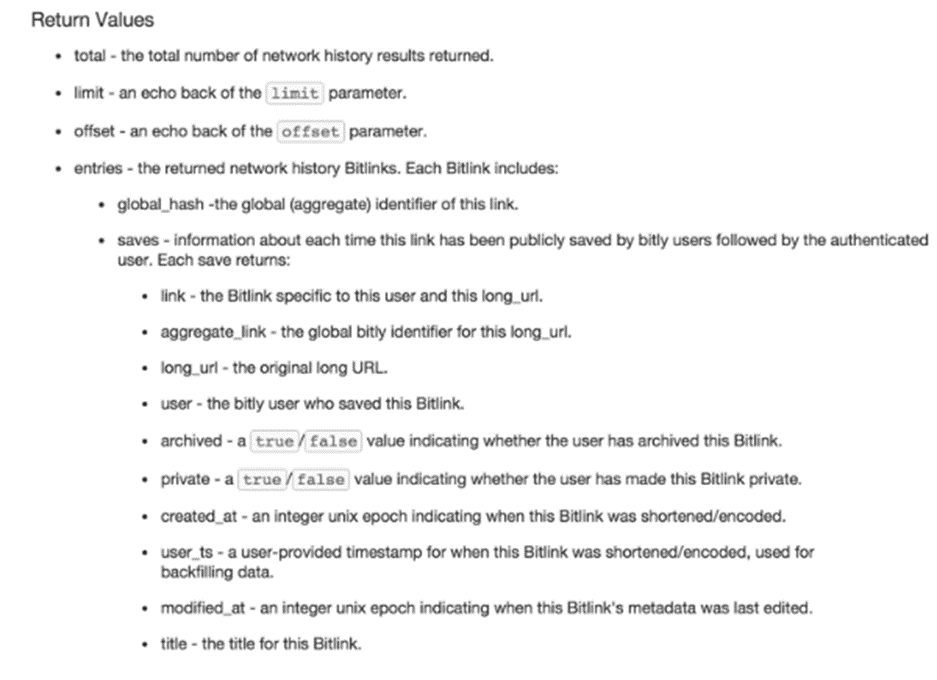
The response example shows a sample response from the request example; the response schema defines all possible elements in the responseble elements in the response.

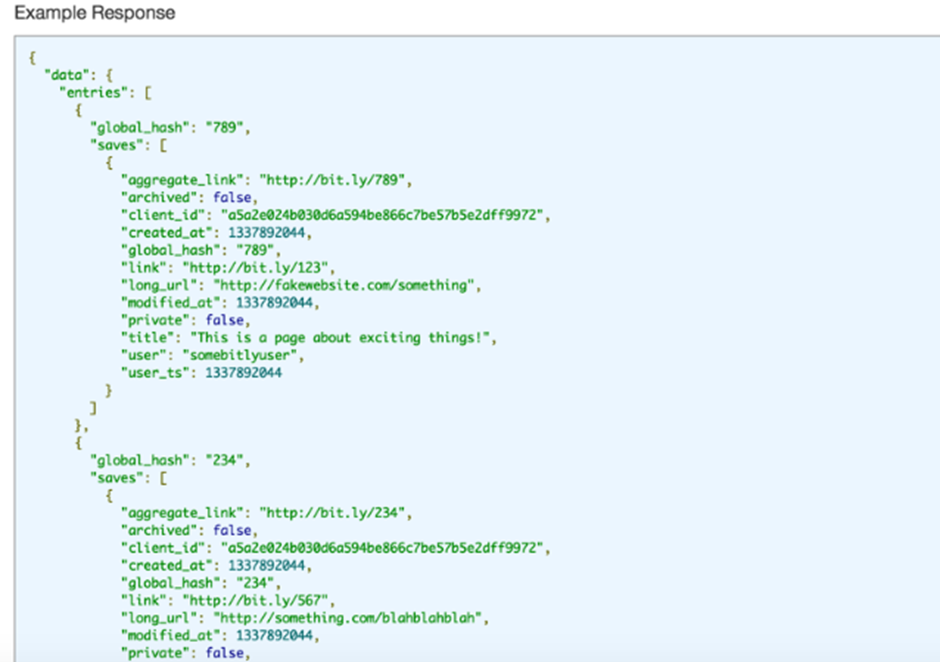


The schema l provides the following:• Description of each property• Definition of the data type for each property• Whether each property is required or optional================



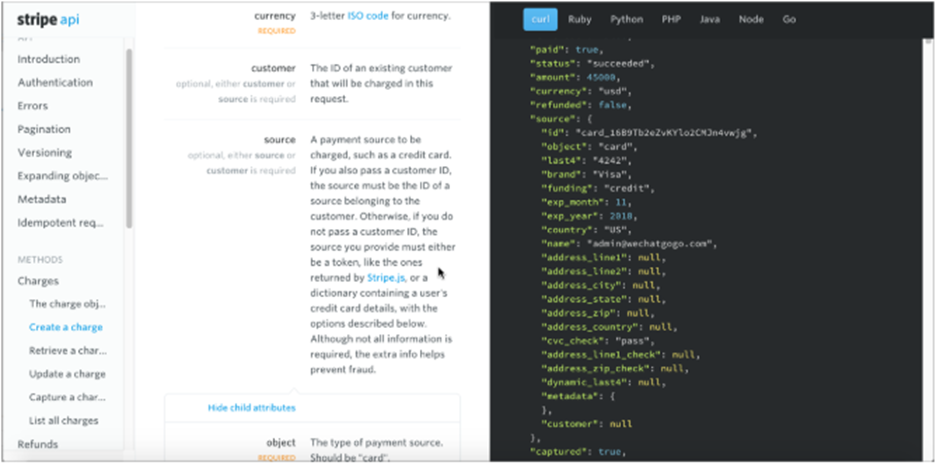






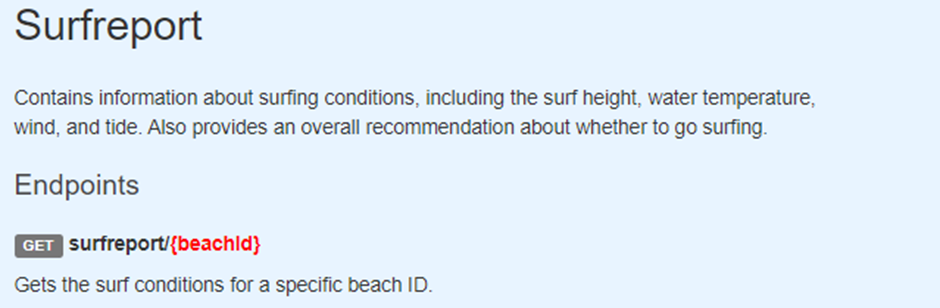
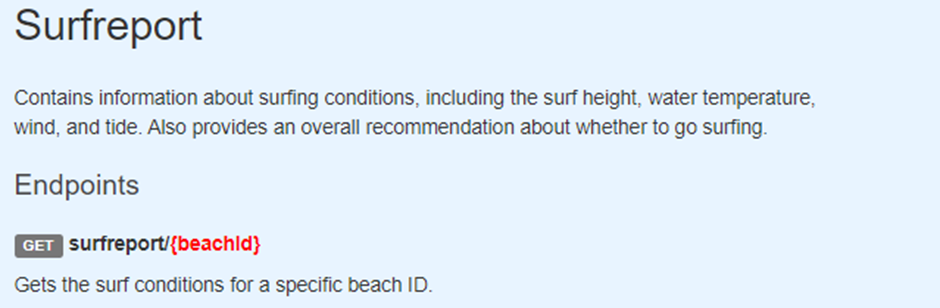
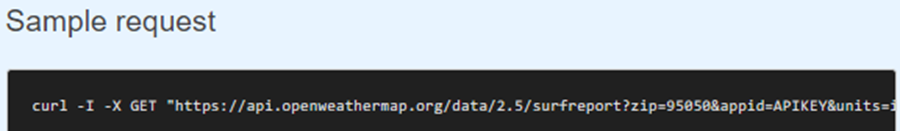


Three-column designs



tatus codes

The responses section sometimes briefly lists the possible status and error codes returned with the responses.



PI documentation

API documentation is a set of human-readable instructions for using and integrating with an API.

API documentation includes detailed information about an API's available endpoints, methods, resources, authentication protocols, parameters, and headers, as well as examples of common requests and responses.

Common types of API documentation

Reference documentation

Reference documentation typically provides a rundown of every endpoint, including its methods, parameters, and accepted data types.

Tutorials

API documentation is presented in the form of tutorials, which provide step-by-step instructions for using the API.

Examples and code samples

Sample-based documentation provides examples of common API requests and responses

Release notes

Release notes include updates on important changes to an API, such as new features, bug fixes, or security patches.

Details to be included when creating API documentation

* Authentication instructions
* Authentication helps keep an API's data safe and secure, and it is the first hurdle that a developer must cross when using a new API.
* API documentation must therefore include a clear explanation of the available authentication methods and provide thorough, step-by-step instructions for obtaining and using authentication credentials.
* Detailed information about every endpoint, operation, and resource
* API documentation should offer a comprehensive overview of every API endpoint and operation, including parameters, headers, and request and response bodies
* Examples of common requests and responses

Examples help consumers understand endpoint behavior under a variety of conditions. Producers should include example requests in several client languages, as well as example responses, which enable consumers to troubleshoot issues they might encounter.

* Terms of Use

Public API documentation should include a Terms of Use, which is a legal agreement that helps producers ensure their API's data and functionality is not abused by consumers.

To write API documentation

• Understand the API

API documentation needs to not only understand the API's purpose, but also be familiar with its endpoints, methods, parameters, accepted data types, and authentication mechanisms

* Know your audience

API documentation is consulted by a wide range of audiences that may have different levels of technical knowledge. It's therefore important to identify your primary audience—and understand their needs—to ensure your documentation is useful.

• Provide detailed instructions for the most common use cases

While writing API's complete functionality, you should pay special attention to the most common use cases. Additional details, such as code samples and example requests

• Review, test, and verify the documentation

. Update the documentation as and when required

Details to Include in API Documentation

An Overview

The overview should contain a summary of the API and the problem it is solving. It could also include the benefits of using this particular API over other similar APIs.

Tutorials

It should include the different content formats you are using to explain the concept of the API to the user. It can also include links for reference and a step-by-step guide for integrating the API and consuming it so it functions properly.

Examples

Once you've explained how the API works and/or provided itemized steps, it's a good idea to show examples of calls, responses, error handling, and other operations that have to do with how the developer interacts with the API.

Glossary

<https://www.postman.com/doordash/doordash/documentation/p3d90qw/doordash-drive?entity=request-19023848-6efcddb7-6ea1-4ea3-a630-1825870cc6cc>

https://www.postman.com/salesforce-developers/salesforce-developers/documentation/oj0opxa/mulesoft-anypoint-platform-apis?entity=request-28629827-f1eb8f70-6c45-4ce8-94d0-091e2c95b90ahttps://www.postman.com/salesforce-developers/salesforce-developers/documentation/oj0opxa/mulesoft-anypoint-platform-apis?entity=request-28629827-f1eb8f70-6c45-4ce8-94d0-091e2c95b90a