Last Updated: 28 Jul, 2021

What is API: Definition, Types, Specifications, Documentation

Reading time: 12 minutes

If you ever read tech magazines or blogs, you've probably seen the abbreviation API. It sounds solid, but what does it mean and why should you bother?

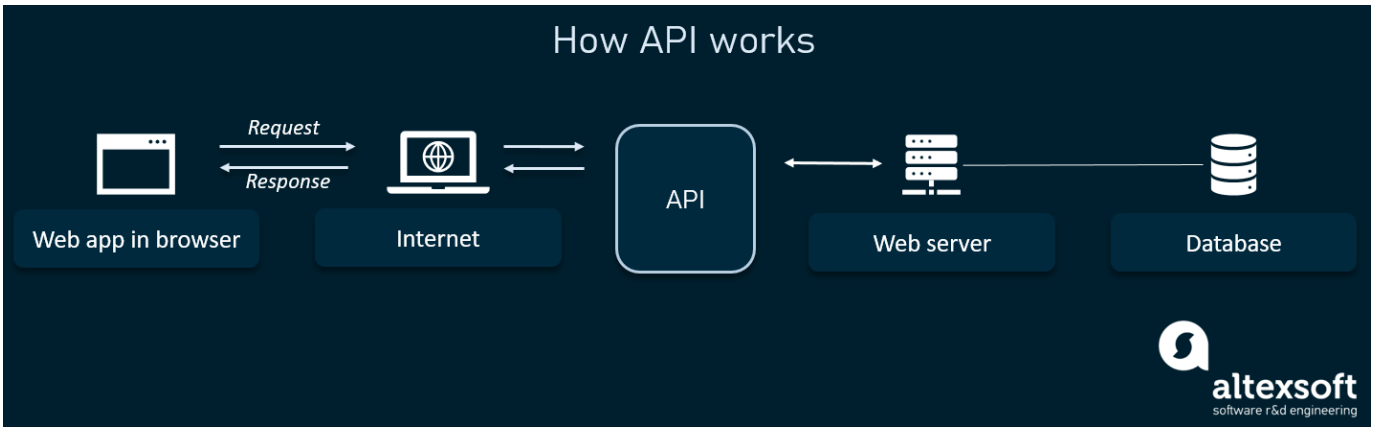
Let's start with a simple example: human communication. We can express our thoughts, needs, and ideas through language (written and spoken), gestures, or facial expressions. Interaction with computers, apps, and websites require user interface components – a screen with a menu and graphical elements, a keyboard, and a mouse.

Software or its elements don't need a graphical user interface to communicate with each other. Software products exchange data and functionalities via machine-readable interfaces – *APIs (application programming interfaces).*

You may also check our video explainer on APIs

What is an API?

An **API** is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange.



How API works.

Application programming interfaces consist of two components:

- Technical specification describing the data exchange options between solutions with the specification done in the form of a request for processing and data delivery protocols
- Software interface written to the specification that represents it

The software that needs to access information (i.e., X hotel room rates for certain dates) or functionality (i.e., a route from point A to point B on a map based on a user's location) from another software, calls its API while specifying the requirements of how data/functionality must be provided. The other software returns data/functionality requested by the former application.

And the interface by which these two applications communicate is what the API specifies.

The Red Hat specialists note that APIs are sometimes considered contracts, where documentation is an agreement between the parties: *"If party first sends a remote request structured a particular way, this is how the second party's software will respond."* The API documentation is a manual for developers that includes all necessary information on how to work with the API and use the services it provides. We will talk more about the documentation in one of the next sections.

Each API contains and is implemented by **function calls** – language statements that request software to perform particular actions and services. Function calls are phrases composed of verbs and nouns, for example:

- Start or finish a session
- Get amenities for a single room type
- Restore or retrieve objects from a server.

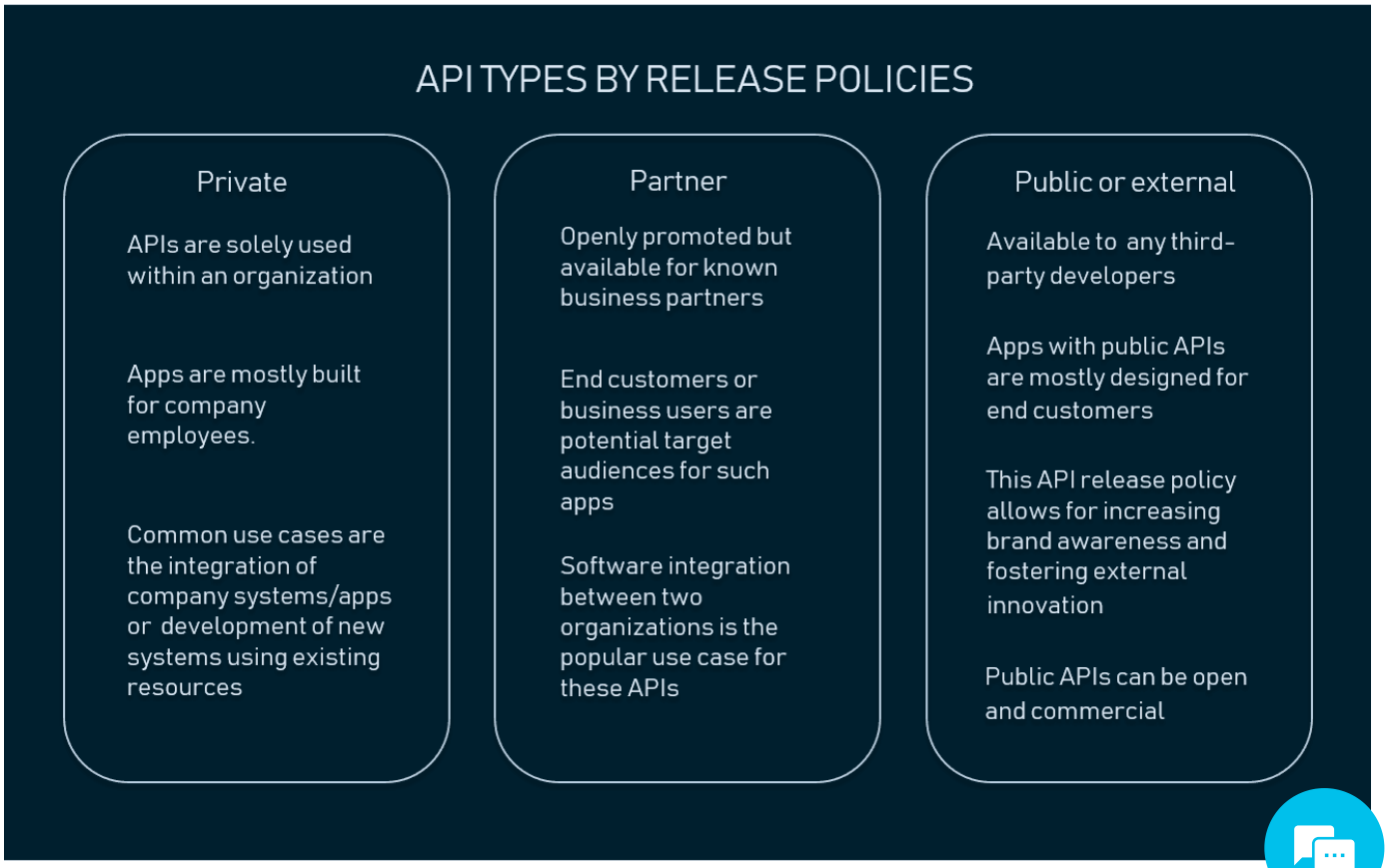
Function calls are described in the API documentation.

APIs serve numerous purposes. Generally, they can simplify and speed up software development. Developers can add functionality (i.e., recommender engine, accommodation booking, image recognition, payment processing) from other providers to existing solutions or build new applications using services by third-party providers. In all these cases, specialists don't have to deal with source code, trying to understand how the other solution works. They simply connect their software to another one. In other words, APIs serve as an abstraction layer between two systems, hiding the complexity and working details of the latter.

Types of APIs

APIs by availability aka release policies

In terms of release policies, APIs can be private, partner, and public.



Private APIs. These application software interfaces are designed for improving solutions and services within an organization. In-house developers or contractors may use these APIs to integrate a company's IT systems or applications, build new systems or customer-facing apps leveraging existing systems. Even if apps are publicly available, the interface itself remains available only for those working directly with the API publisher. The private strategy allows a company to fully control the API usage.

Partner APIs. Partner APIs are openly promoted but shared with business partners who have signed an agreement with the publisher. The common use case for partner APIs is software integration between two parties. A company that grants partners with access to data or capability benefits from extra revenue streams. At the same time, it can monitor how the exposed digital assets are used, ensure whether third-party solutions using their APIs provide decent user experience, and maintain corporate identity in their apps.

Public APIs. Also known as developer-facing or external, these APIs are available for any third-party developers. A public API program allows for increasing brand awareness and receiving an additional source of income when properly executed.

There are two types of public APIs – open (free of charge) and commercial ones. The Open API Definition suggests that all features of such an API are public and can be used without restrictive terms and conditions. For instance, it's possible to build an application that utilizes the API without explicit approval from the API supplier or mandatory licensing fees. The definition also states that the API description and any related documentation must be openly available, and that the API can be freely used to create and test applications.

Commercial API users pay subscription fees or use APIs on a pay-as-you-go basis. A popular approach among publishers is to offer free trials, so users can evaluate APIs before purchasing subscriptions. Learn more about how businesses benefit from opening their APIs for public use in our detailed article on API economy.

APIs by use cases

APIs can be classified according to the systems for which they are designed.

Database APIs. Database APIs enable communication between an application and a database management system. Developers work with databases by writing queries to access data, change tables, etc. The Drupal 7 Database API, for example, allows users to write unified queries for different databases, both proprietary and open source (Oracle, MongoDB, PostgreSQL, MySQL, CouchDB, and MSSQL).

Another example is ORDS database API, which is embedded into Oracle REST Data Services.

Operating systems APIs. This group of APIs defines how applications use the resources and services of operating systems. Every OS has its set of APIs, for instance, Windows API or Linux API (kernel–user space API and kernel internal API).

Apple provides API reference for macOS and iOS in its developer documentation. APIs for building applications for Apple's macOS desktop operating system are included in the Cocoa set of developer tools. Those building apps for the iOS mobile operating system use Cocoa Touch – a modified version of Cocoa.

Remote APIs. Remote APIs define standards of interaction for applications running on different machines. In other words, one software product accesses resources located outside the device that requests them, which explains the name. Since two remotely located applications are connected over a communications network, particularly the internet, most remote APIs are written based on web standards. Java Database Connectivity API and Java Remote Method Invocation API are two examples of remote application programming interfaces.

Web APIs. This API class is the most common. Web APIs provide machine-readable data and functionality transfer between web-based systems which represent client-server architecture. These APIs mainly deliver requests from web applications and responses from servers using Hypertext Transfer Protocol (HTTP).

Developers can use web APIs to extend the functionality of their apps or sites. For instance, the Pinterest API comes with tools for adding users' Pinterest data like boards or Pins to a website. Google Maps API enables the addition of a map with an organization's location.

Most businesses use more than one API to connect applications and share information. Some end up needing an API management tool to help them control, distribute, and analyze different APIs. Learn more about API management in our detailed article.

API specifications/protocols

The goal of API specifications is to standardize data exchange between web services. In this case, standardization means the ability of diverse systems, written in different programming languages and/or running on different OSs, or using different technologies, to seamlessly communicate with each other.

Several API specifications are in place

Remote Procedure Call (RPC)

Web APIs may adhere to resource exchange principles based on a Remote Procedure Call. This protocol specifies the interaction between client-server based applications. One program (client) requests data or functionality from another program (server), located in another computer on a network, and the server sends the required response.

RPC is also known as a subroutine or function call. One of two ways to implement a remote procedure call is SOAP.

Service Object Access Protocol (SOAP)

SOAP is a lightweight protocol for exchanging structured information in a decentralized, distributed environment, according to the definition by Microsoft that developed it. Generally speaking, this specification contains the syntax rules for request and response messages sent by web applications. APIs that comply with the principles of SOAP enable XML messaging between systems through HTTP or Simple Mail Transfer Protocol (SMTP) for transferring mail.

Extensible markup language (XML) is a simple and very flexible text format widely used for data storage and exchange over the internet or other networks. XML defines a set of rules for encoding documents in a format that both humans and machines can read. The markup language is a collection of symbols that can be placed in the text to delineate and label the parts of the text document. XML text documents contain self-descriptive tags of data objects, which makes them easily readable.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:RequestHeader
         soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"
         soapenv:mustUnderstand="0"
         xmlns:ns1="https://www.google.com/apis/ads/publisher/v201905">
      <ns1:networkCode>123456</ns1:networkCode>
      <ns1:applicationName>DfpApi-Java-2.1.0-dfp_test</ns1:applicationName>
    </ns1:RequestHeader>
  </soapenv:Header>
  <soapenv:Body>
    <getAdUnitsByStatement xmlns="https://www.google.com/apis/ads/publisher/v201905">
      <filterStatement>
        <query>WHERE parentId IS NULL LIMIT 500</query>
      </filterStatement>
    </getAdUnitsByStatement>
  </soapenv:Body>
</soapenv:Envelope>
```

An example of a SOAP XML request call in Google Ad Manager. Source: Google Ad Manager

SOAP is mostly used with enterprise web-based software to ensure high security of transmitted data. SOAP APIs are preferred among providers of payment gateways, identity management and CRM solutions, as well as financial and telecommunication services. PayPal's public API is one of the commonly known SOAP APIs. It's also frequently used for legacy system support.

Representational State Transfer (REST)

The term *REST* was introduced by computer scientist Roy Fielding in a dissertation in 2000. Unlike SOAP, which is a protocol, REST is a software architectural style with six constraints for building applications that work over HTTP, often web services. The World Wide Web is the most common realization and application of this architecture style.

REST is considered a simpler alternative to SOAP, which many developers find difficult to use because it requires writing a lot of code to complete every task and following the XML structure for every message sent. REST follows another logic since it makes data available as resources. Each resource is represented by a unique URL, and one can request this resource by providing its URL.

Web APIs that comply with REST architectural constraints are called RESTful APIs. These APIs use HTTP requests (AKA methods or verbs) to work with resources: GET, PUT, HEAD, POST, PATCH, CONNECT, TRACE, OPTIONS and DELETE.

RESTful systems support messaging in different formats, such as plain text, HTML, YAML, XML, and JSON, while SOAP only allows XML. The ability to support multiple formats for storing and exchanging data is one of the reasons REST is a prevailing choice for building public APIs these days.

Social media giants and travel companies provide external APIs to improve their brand visibility even more. Twitter has numerous RESTful APIs; Expedia has both SOAP and RESTful APIs for its partners. If you consider redefining your travel and hospitality business offering, dive deep into the world of travel and booking APIs with our dedicated article.

JavaScript Object Notation (JSON) is a lightweight and easy-to-parse text format for data exchange. Its syntax is based on a subset of the Standard ECMA-262 3rd Edition. Each JSON file contains collections of name/value pairs and ordered lists of values. Since these are universal data structures, the format can be used with any programming language.

```
curl -i -X GET \
-H 'Content-Type:application/json' \
-H 'Authorization:bearer 4b0e2253-1a34-4977-bb09-1c46a69a
'https://platform.otqa.com/sync/directory'
```


```
"offset": 0,
"limit": 100,
"total_items": 35034,
"items": [{
    "rid": 2,
    "name": "Thirsty Bear",
    "address": "661 Howard St.",
    "address2": null,
    "city": "San Francisco",
    "state": "CA",
    "country": "US",
    "latitude": "37.7856500",
    "longitude": "-122.3997340",
    "postal_code": "94105",
    "phone_number": "4159740905",
    "metro_name": "San Francisco Bay Area",
    "is_restaurant_in_group": false,
    "reservation_url": "http://www.opentable.com/restaurant
    "profile_url": "http://www.opentable.com/restaurant/pro
    "natural_profile_url": "https://www.opentable.com/r/bou
    "natural_reservation_url": "https://www.opentable.com/r
    "aggregate_score": "4.400",
    "price_quartile": "$$",
    "review_count": 155,
    "category": ["Spanish", "Brewery"]
}
```


A GET request for restaurant details with a response in JSON. Source: Open Table

JSON has been widely adopted thanks to the popularity of REST.

gRPC

gRPC is an open-source universal API framework that is also classified under RPC. Unlike SOAP, gRPC is much newer and was released publicly in 2015 by Google. With gRPC, the client application can directly call methods from a server application located on a different computer as if it was a local object. This makes it easier to create distributed services and applications.

Like SOAP and REST, the transport layer for gRPC is HTTP. However, similar to RCP, gRPC allows developers to define any kind of function calls, rather than selecting from predefined options such as PUT and GET in the case of REST.

By default, gRPC uses protocol buffers instead of JSON or XML as the Interface Definition Language (IDL) for serializing structured data. Here, the developer needs to first define the structure of the data they want to serialize. Once the data structures have been specified, they use the protocol buffer compiler to generate the data access classes in the programming language of your choice. The data is then compressed and serialized in binary format at runtime. Learn more about gRPC in our detailed article.

```
// The greeter service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

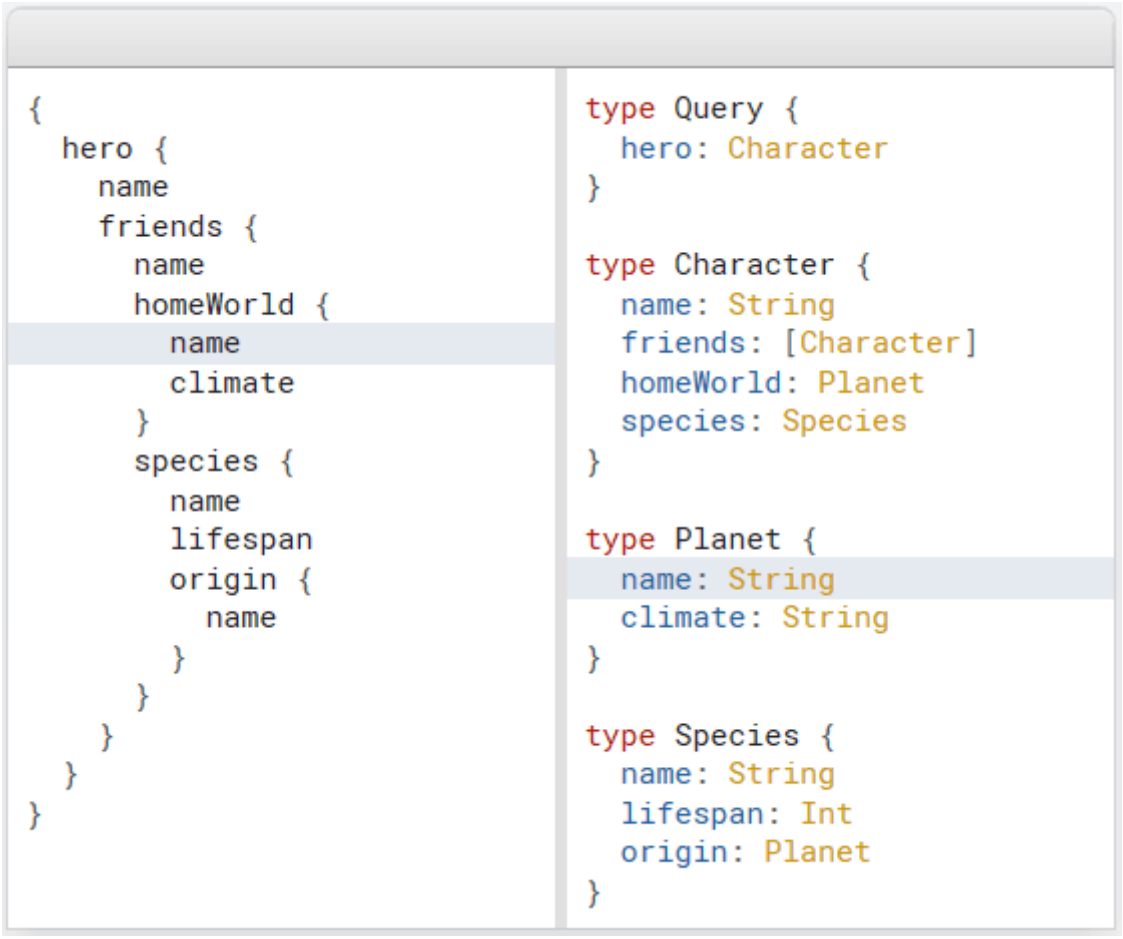
Example of RPC method parameters and return types. Source: gRCP

gRPC is mostly used for communication between microservices because it is available in multiple languages and has a high performance.

GraphQL

The need for faster feature development, more efficient data loading due to increased mobile adoption, and a multitude of clients, made the developers look for other approaches to software architecture. GraphQL, initially created by Facebook in 2012 for internal use, is the new REST with organizations like Shopify, Yelp, GitHub, Coursera, and *The New York Times* using it to build APIs.

GraphQL is a query language for APIs. It allows the client to detail the exact data it needs and simplifies data aggregation from multiple sources, so the developer can use one API call to request all needed data. Another special feature of GraphQL is that it uses a *type system* to describe data.



```
{
  hero {
    name
    friends {
      name
      homeWorld {
        name
        climate
      }
      species {
        name
        lifespan
        origin {
          name
        }
      }
    }
  }
}
```

```
type Query {
  hero: Character
}

type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
  species: Species
}

type Planet {
  name: String
  climate: String
}

type Species {
  name: String
  lifespan: Int
  origin: Planet
}
```

Using types to describe data allows apps to specify what data they need to get

Apps using GraphQL control what data they need to fetch from a server, which allows them to run fast even when the mobile connection is slow. You can see how GraphQL, REST, RPC, and SOAP are compared in the linked article.

API documentation

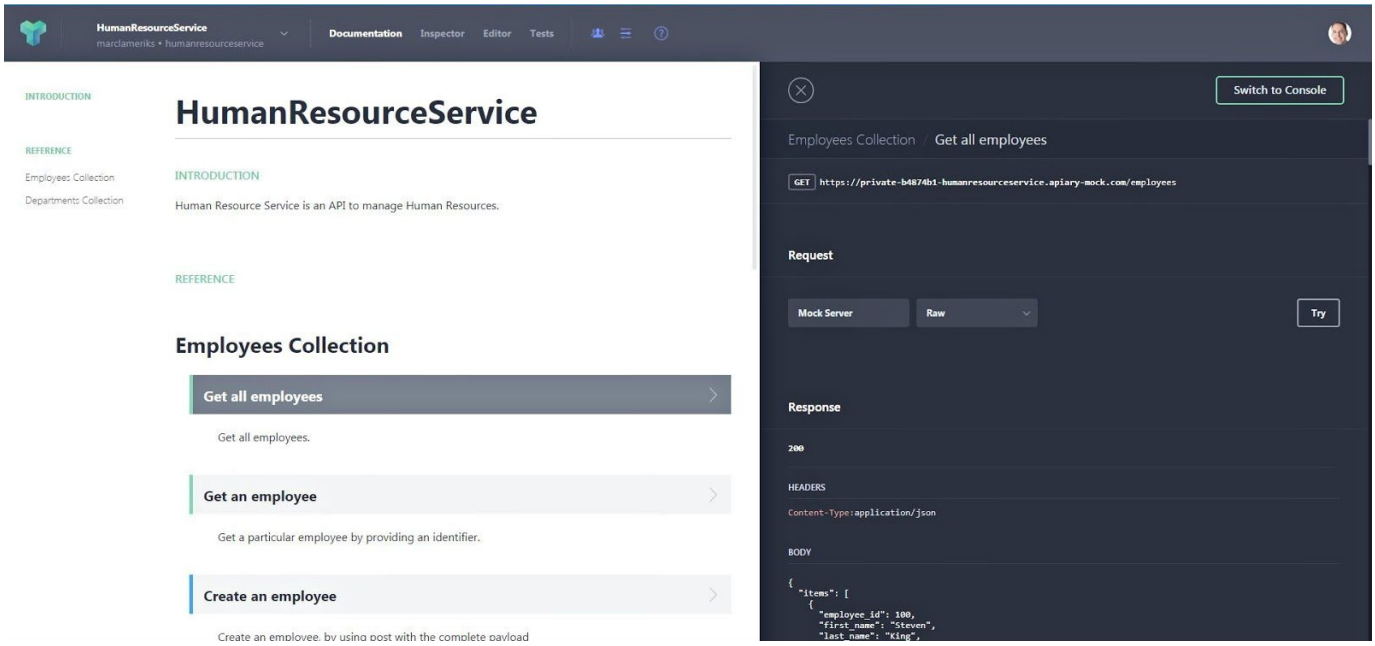
No matter how many opportunities for creating or extending software products API gives, it would remain an unusable piece of code if developers didn't understand how to work with it. Well-written and structured API documentation that explains how to effectively use and integrate an API in an easy-to-comprehend manner will make a developer happy and eager to recommend the API to peers.

The API documentation is a reference manual with all needed information about the API, including functions, classes, return types, and arguments.

Numerous content elements make good documentation, such as:

- a quick start guide
- authentication information
- explanations for every API call (request)
- examples of every request and return with a response description, error messages, etc.
- samples of code for popular programmatic languages like Python, Java, JavaScript, or PHP;
- tutorials
- SDK examples (if SDKs are available) illustrating how to access the resource, etc.

Documentation may be *static* and *interactive.* The latter allows for trying out APIs and see return results and usually consists of two columns: human and machine. The human column contains API descriptions, and the machine one has a console to make calls and contains info that clients and servers will be interested in when testing the API.



Human and machine columns in the documentation Code examples on the machine column (right) after a user clicked for an action ("Get all employees"). Source: AMIS

Generation is the process of documenting APIs by developers and technical writers. The specialists may use API documentation solutions (i.e., Swagger tools, Postman, Slate, or ReDoc) to unify documentation structure and design.

Examples of APIs

Here are some examples of well-known APIs that use different protocols and specifications. Check their documentation to get more information and references.

Google Maps. It is no secret that Google is among the tech giants, and they have set the standards in the way other companies operate. Most websites that have an integrated map are using the Google Maps APIs. For example, Google's Directions API uses an HTTP request to return XML or JSON-formatted directions between geolocations.

Vulkan. Vulkan is a cross-platform API that works on the operating system level. It enables developers to create high-quality, real-time graphics in applications and drives the communication between an application and graphical processing unit. Check Vulkan API documentation if you're interested.

Skyscanner Flight Search. Skyscanner is a metasearch platform that lets travelers look for flights at the best rates from Skyscanner's database of prices. Also, Skyscanner provides its affiliate partners with RESTful API supporting both XML and JSON as the data exchange formats. In order to improve security, they encourage partners to use only HTTPS protocol to make requests. You may check their documentation here.

WeatherAPI. This is a free geolocation and weather information provider with lots of different APIs ranging from the weather forecast, IP lookup, sports, astronomy, geolocation, and time zone. It provides access to geodata and weather using a JSON/XML RESTful API. Developers can use either HTTP or HTTPS to request the API. They provide developers with detailed documentation on how to use all of their APIs.

Sabre Air Availability. This is a Sabre SOAP API used to search for flights and the corresponding availability information for given dates, origins, and destinations. Since it is a SOAP API, it uses XML as the data exchange format, and HTTP or HTTPS protocols for requests.

Yelp API. This is a GraphQL API that provides users recommendations and reviews of the best restaurants, things to do, nightlife, and more. It uses the HTTP request method to access data from the servers. The API connects to data sources through endpoints, which developers can add to their apps. It uses JSON as the data exchange format.

Final word

The role of APIs is considerably greater if we look at it not only from the software development angle but also from the business collaboration angle. These machine-readable interfaces for resource exchange are like delivery services that work under the hood and enable that needed technological connectivity. According to the Fourth State of API Integration Report of 2020, 83 percent of participants find API integration "critical" to their businesses and IT infrastructures.

In this regard, the two main tasks for decision makers and developers are to select the API that works for a company's specific business needs and understand how to effectively use it.

Subscribe to our newsletter

Enter your email

Subscribe

Yes, I understand and agree to the Privacy Policy

2 Comments >

Further Reading

Travel and Booking APIs for Online Travel and Tourism Service Providers

Car Rental APIs: Integrations with GDSs, OTAs, and Tech Providers

GraphQL: Core Features, Architecture, Pros and Cons