What is an API

**API** stands for Application Programming Interface. An **API** is a software intermediary that allows two applications to talk to each other. In other words, an **API** is the messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you

An API defines functionalities that are independent of their respective implementations, which allows those implementations and definitions to vary without compromising each other. Therefore, a good API makes it easier to develop a program by providing the building blocks.

How API works

Imagine a waiter in a restaurant.  You, the customer, are sitting at the table with a menu of choices to order from, and the kitchen is the provider who will fulfill your order.

You need a link to communicate your order to the kitchen and then to deliver your food back to your table. It can’t be the chef because she’s cooking in the kitchen. You need something to connect the customer who’s ordering food and the chef who prepares it.  That’s where the waiter — or the API —  comes into picture.

Types of API’s

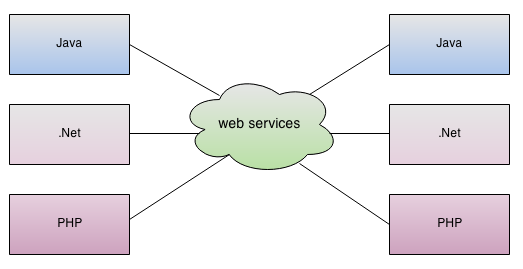
There are numerous types of APIs. For example, you may have heard of Java APIs, or interfaces within classes that let objects talk to each other in the Java programming language. Along with program-centric APIs, there are also Web APIs such as the Simple Object Access Protocol (SOAP), Remote Procedure Call (RPC), and perhaps the most popular—at least in name— Representational State Transfer (REST). There are 15,000 publicly available APIs, according to Programmable Web, and many thousands of more private APIs that companies use to expand their internal and external capabilities.

What is Web Service

A **Web Service** is can be defined by following ways:

* is a client server application or application component for communication.
* method of communication between two devices over network.
* is a software system for interoperable machine to machine communication.
* is a collection of standards or protocols for exchanging information between two devices or application.

Let's understand it by the figure given below:



As you can see in the figure, java, .net or PHP applications can communicate with other applications through web service over the network. For example, java application can interact with Java, .Net and PHP applications. So web service is a language independent way of communication.

## Types of Web Services

There are mainly two types of web services.

1. SOAP web services.
2. RESTful web services.

# Web Service Components

There are three major web service components.

1. SOAP
2. WSDL
3. UDDI

## SOAP

SOAP is an acronym for Simple Object Access Protocol.

SOAP is a XML-based protocol for accessing web services.

SOAP is a W3C recommendation for communication between applications.

SOAP is XML based, so it is platform independent and language independent. In other words, it can be used with Java, .Net or PHP language on any platform.

## WSDL

WSDL is an acronym for Web Services Description Language.

WSDL is a xml document containing information about web services such as method name, method parameter and how to access it.

WSDL is a part of UDDI. It acts as a interface between web service applications.

WSDL is pronounced as wiz-dull.

## UDDI

UDDI is an acronym for Universal Description, Discovery and Integration.

UDDI is a XML based framework for describing, discovering and integrating web services.

UDDI is a directory of web service interfaces described by WSDL, containing information about web services.

# SOAP Web Services

SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.

SOAP is a W3C recommendation for communication between two applications.

SOAP is XML based protocol. It is platform independent and language independent. By using SOAP, you will be able to interact with other programming language applications.

## Advantages of Soap Web Services

**WS Security**: SOAP defines its own security known as WS Security.

**Language and Platform independent**: SOAP web services can be written in any programming language and executed in any platform.

## Disadvantages of Soap Web Services

**Slow**: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

**WSDL dependent**: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

# RESTful Web Services

REST stands for REpresentational State Transfer.

REST is an architectural style not a protocol.

## Advantages of RESTful Web Services

**Fast**: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.

**Language and Platform independent**: RESTful web services can be written in any programming language and executed in any platform.

**Can use SOAP**: RESTful web services can use SOAP web services as the implementation.

**Permits different data format**: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

**REST API Methods:**

RESTful APIs enable you to develop any kind of web application having all possible CRUD (create, retrieve, update, delete) operations. REST guidelines suggest you to use specific HTTP method on specific type of call made to server (though technically it is possible to violate this guideline, yet it is highly discouraged).

* HTTP GET
* HTTP POST
* HTTP PUT
* HTTP DELETE
* HTTP PATCH

## HTTP GET

Use GET requests **to retrieve resource representation/information only** – and not to modify it in any way. As GET requests do not change the state of resource, these are said to be **safe methods**. Additionally, GET APIs should be **idempotent**, which means that making multiple identical requests must produce same result everytime until another API (POST or PUT) has changed the state of resource on server.

If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

For any given HTTP GET API, if resource is found on server then it must return HTTP response code 200 (OK) – along with response body which is usually either XML of JSON content (due to their platform independent nature).

In case resource is NOT found on server then it must return HTTP response code 404 (NOT FOUND). Similarly, if it is determined that GET request itself is not correctly formed then server will return HTTP response code 400 (BAD REQUEST).

## HTTP POST

Use POST APIs **to create new subordinate resources**, e.g. a file is subordinate to a directory containing it or a row is subordinate to a database table. Talking strictly in terms of REST, POST methods are used to create a new resource into the collection of resources.

Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a [Location](https://en.wikipedia.org/wiki/HTTP_location) header.

Many times, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.

Responses to this method are **not cacheable**, unless the response includes appropriate [Cache-Control](https://en.wikipedia.org/wiki/Web_cache#Cache_control)or [Expires](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html) header fields.

Please note that POST is **neither safe nor idempotent** and invoking two identical POST requests will result in two different resources containing the same information (except resource ids).

## HTTP PUT

Use PUT APIs primarily **to update existing resource** (if resource does not exist then API may decide to create a new resource or not). If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201 (Created) response and if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

*The difference between the POST and PUT APIs can be observed in request URIs. POST requests are made of resource collections whereas PUT requests are made on individual resource.*

## HTTP DELETE

As the name applies, DELETE APIs are used **to delete resources** (identified by the Request-URI).

A successful response of DELETE requests SHOULD be HTTP response code 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has been queued, or 204 (No Content) if the action has been performed but the response does not include an entity.

DELETE operations are **idempotent**. If you DELETE a resource, it’s removed from collection of resource. Repeatedly calling DELETE API on that resource will not change the outcome – however calling DELETE on a resource a second time will return a 404 (NOT FOUND) since it was already removed. Some may argue that it makes DELETE method non-idempotent. It’s matter of discussion and personal opinion.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are **not cacheable**.

## HTTP PATCH

HTTP PATCH requests are **to make partial update on a resource**. If you see PUT requests also modify a resource entity so to make more clear – PATCH method is the correct choice for partially updating an existing resource and PUT should only be used if you’re replacing a resource in it’s entirety.

Please note that there are some challenges if you decide to use PATCH APIs in your application:

* Support for PATCH in browsers, servers and web application frameworks is not universal. IE8, PHP, Tomcat, django, and lots of other software has missing or broken support for it.
* Request payload of PATCH request in not straightforward as it is for PUT request. e.g.

HTTP GET /users/1

produces below response:

{id: 1, username: 'admin', email: 'email@example.org'}

A sample patch request to update the email will be like this:

HTTP PATCH /users/1

[  
{ “op”: “replace”, “path”: “/email”, “value”: “new.email@example.org” }  
]

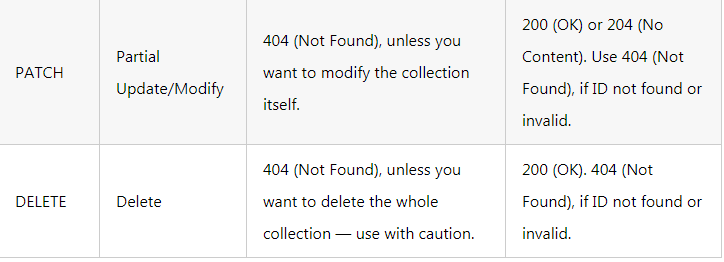
There may be following possible operations are per HTTP specification.

[  
{ "op": "test", "path": "/a/b/c", "value": "foo" },  
{ "op": "remove", "path": "/a/b/c" },  
{ "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] },  
{ "op": "replace", "path": "/a/b/c", "value": 42 },  
{ "op": "move", "from": "/a/b/c", "path": "/a/b/d" },  
{ "op": "copy", "from": "/a/b/d", "path": "/a/b/e" }  
]

PATCH method is not a replacement for the POST or PUT methods. It applies a delta (diff) rather than replacing the entire resource.

## Summary of HTTP Methods for RESTful APIs





### Safe Methods

As per HTTP specification, the **GET and HEAD methods should be used only for retrieval of resource representations** – and they do not update/delete the resource on server. Both methods are said to be considered “**safe**“.

This allows user agents to represent other methods, such as **POST, PUT and DELETE**, in a special way, so that the user is made aware of the fact that a possibly unsafe action is being requested – and they can **update/delete the resource on server** and so should be used carefully.

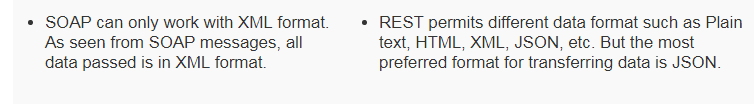
### Idempotent Methods

The term idempotent is used more comprehensively to describe an **operation that will produce the same results if executed once or multiple times**. This is a very useful property in many situations, as it means that an operation can be repeated or retried as often as necessary without causing unintended effects. With non-idempotent operations, the algorithm may have to keep track of whether the operation was already performed or not.

In HTTP specification, The methods **GET, HEAD, PUT and DELETE are declared idempotent methods**. Other methods OPTIONS and TRACE SHOULD NOT have side effects so both are also inherently idempotent.

**Difference between SOAP & REST :**

****

****

**http Status Codes :**

Every HTTP transaction has a status code sent back by the server to define how the server handled the transaction. Here is a list of the most common ones.

## List of Common HTTP Status Codes

* **200 - OK**
* **300 - Multiple Choices**
* **301 - Moved Permanently**
* **302 - Found**
* **304 - Not Modified**
* **307- Temporary Redirect**
* **400- Bad Request**
* **401- Unauthorized**
* **403- Forbidden**
* **404- Not Found**
* **410 - Gone**
* **500 - Internal Server Error**
* **501 - Not Implemented**
* **503- Service Unavailable**
* **550 - Permission denied**

**API Authentication :**

[**https://nordicapis.com/3-common-methods-api-authentication-explained/**](https://nordicapis.com/3-common-methods-api-authentication-explained/)

**Sample REST URLs for Testing:**

[**http://parabank.parasoft.com/parabank/services/bank/customers/12212/**](http://parabank.parasoft.com/parabank/services/bank/customers/12212/)

[**http://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22**](http://samples.openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22)

[**http://restcountries.eu/rest/v1/name/norway**](http://restcountries.eu/rest/v1/name/norway)

**Facebook Graph API details -** [**https://developers.facebook.com/tools/explorer/**](https://developers.facebook.com/tools/explorer/)

[**https://www.youtube.com/watch?v=WteK95AppF4**](https://www.youtube.com/watch?v=WteK95AppF4)

**SOAP Sample Project Link :**

[**https://www.soapui.org/tutorials/soap-sample-project.html**](https://www.soapui.org/tutorials/soap-sample-project.html)

**Interview Questions for SOAP/REST/Webservices:**

[**https://www.softwaretestinghelp.com/web-services-interview-questions/**](https://www.softwaretestinghelp.com/web-services-interview-questions/)

[**https://www.javatpoint.com/web-services-interview-questions**](https://www.javatpoint.com/web-services-interview-questions)

[**https://www.softwaretestinghelp.com/restful-web-services-interview-question/**](https://www.softwaretestinghelp.com/restful-web-services-interview-question/)

[**https://java2blog.com/web-services-interview-questions/**](https://java2blog.com/web-services-interview-questions/)