Contents

[What is API 1](#_Toc4945989)

[What is URL, URL & URI? 1](#_Toc4945990)

[What are different HTTP methods? 1](#_Toc4945991)

[What is HTTP Status code 3](#_Toc4945992)

[What is Rest (REpresentational State Transfer) 5](#_Toc4945993)

[What are principles of REST 5](#_Toc4945994)

[Difference between Rest & Rest API 6](#_Toc4945995)

[What is SOAP 6](#_Toc4945996)

[Difference between Rest & SOAP 7](#_Toc4945997)

[What is Accept & Content-type? 7](#_Toc4945998)

[What is HTTP Headers in API 7](#_Toc4945999)

### What is API

**An API is a set of codes which allows two or more than two applications to communicate each other internally or externally and provide a result to end users or to another API.** An API is developed to serve a purpose/functionality which may fall in one of these categories:

1. **Creating data**
2. **Retrieving data**
3. **Modifying data**
4. **Delete data**

### ****What is URL, URL & URI?****

URN – Uniform ***Resource*** Name  
URL – Uniform ***Resource*** Locator   
URI – Uniform ***Resource*** Identifier

A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource. Each URI begins with a scheme name, as defined in Section 3.1, that refers to a specification for assigning identifiers within that scheme.

Resource is a physical or virtual component, which supports/extends/represents/constructs a system. Consider a real time example of Motor Bike. Wheels, breaks, headlight, fuel tank etc all are resources of a Motor bike. All resources have names, location and identifiers.

### What are different HTTP methods?

List of HTTP methods are as follows -

#### ****GET****: This HTTP method is used to read/retrieve resource representation only. It is called Safe methods as it can not modify information. It should retrieve same information for multiple identical requests until any other API has changed the state of resource. That’s why it is also called as idempotent method. Response is returned in JSON/XML format.

**Status Codes:**

**200 (OK)** –> If GET API finds the requested resource.

**404( Not Found)** –> If GET API does not find the requested resource.

**400 ( Bad Request)** –> If GET request is not formed properly.

#### POST: A HTTP POST method is used to create a new resource in collection of resources with a request body passed as a JSON/XML. If resource is created successfully at the end point, it returns a status code 201( Created) (Not always) and returns response body. It may return 200 (OK) and 204 (No Content) status code as well based on how it is created.

POST is not safe method as it is related to data creation. It is also not **idempotent** and invoking two identical POST requests will result in two different resources containing the same information with just different resource ids.

#### PUT: An HTTP PUT method is used to primarily update the resource information but it also can be used to create a new resource (Depends on API development) if requested resource is not available.  If PUT request is made to update resource, it should return 200 (OK) and 204 (No Content) status code. If PUT request is made to create a new resource, it must return a status code 201( Created).

PUT is not a safe method as it performs data creation and modifications but it is idempotent as if we hit the same request again, it operates on same existing resource. But note here that a PUT request can be made as non-idempotent as well.

#### DELETE: An HTTP DELETE method is used to delete an existing resource from collection of resources. On successful deletion of resource, it returns  200 (OK) and 204 (No Content) status code. It may return as 202 (Accepted) status code if request is queued.

It is not a safe method as it performs on modification of data. If we hit the same request again after first hit, it will give you 404 ( Not Found) . So DELETE request are idempotent after second call onward.

#### PATCH: An HTTP PATCH method is used to update information of resource partially. It is different from PUT as PUT updates/replace complete information of resource while PATCH updates  some information of resource. It returns  200 (OK) and 204 (No Content) status code

A PATCH method is not safe method as it operations on modification of data. It is also non-idempotent but can be made idempotent.

#### HEAD: An HTTP HEAD method is identical to GET method without response body. Instead of response body or resource information, a GET request returns meta information/headers contented in an HTTP GET method. This method can be used for obtaining meta information about the entity implied by the request without transferring the entity-body itself.

#### OPTIONS: An HTTP OPTIONS method which is used to get information about allowed operations on given URI. It returns a response header named “Allow”  with the list of available operation on given URI.

### What is HTTP Status code

When an HTTP request is hit, server returns status of that request as status line part of HTTP response. As per [RFC-2616](https://www.ietf.org/rfc/rfc2616.txt):

**The first line of a Response message is the Status-Line, consisting of the protocol version followed by a numeric status code and its associated textual phrase, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.**

1. **200 (OK)** : All is GOOD. It is the most positive status code which everyone expects. This status code is thrown when requested operation on server by client is successfully processed.
2. **201 (Created)** : This status code is thrown when an HTTP method to hit to create a new resource on server and resource is created successfully. This status code makes more sense for HTTP methods which are meant to create new resources.
3. **202(Accepted)** : If a request is queued for processing or takes longer time to process, this status code is thrown to client. The request might or might not be eventually acted upon, or even maybe disallowed when processing occurs.
4. **204 (No Content)** : When a request is processed successfully but returns no state representation of resource to be included in the response message body, throws 204 status code. Its success without response body.
5. **400 (Bad Request)** : It is a client side error made when user submits inappropriate request like malformed request syntax, invalid request message parameters, or deceptive request routing etc. Client should correct request before hitting again.
6. **401 (Unauthorised)** : When a client submits a request with no or wrong authorisation on a resource which is protected by authorisation , this status code is thrown. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.
7. **403 (Forbidden)** : When a user tries to access a resource on which user has no permission, this error will be thrown. It is not like 401. Request and proper authorization is provided in request body but user has no access on request resource.
8. **404(Not Found)** : If the request resource is not available at given URI or rest api can’t map the client’s URI to a resource, this status code is thrown.
9. **405(Method Not Allowed)** : When a client calls an HTTP method on a resource which is not applicable to on it, this status code is thrown. For example: If client hits a POST method on a GET resource, it will throw 405. Response includes  “Allow” header, which lists the HTTP methods that the resource supports.
10. **406(Not Acceptable)** : If API can not format data as per client’s provided media types in Accept request header, this status code is thrown.  For example, a client request for data formatted as application/xml will receive a 406 response if the API is only willing to format data as application/json.
11. **415 (Media Type)** : If API is not able to process the client’s supplied media type, as indicated by the Content-Type request header, this status code is thrown. It indicates that part of the request was in an unsupported format. For example, a client request including data formatted as application/xml will receive a 415 response if the API is only willing to process data formatted as application/json.
12. **500 (Internal Server Error)** : It is server side error. When server encounters an exception while processing a request, this status code is thrown. Please note here it is not client side error.
13. **501 (Not Implemented)** : When the server does not recognize the request method and is not capable of supporting it for any resource, this status code is thrown. The server either does not recognize the request method, or it lacks the ability to fulfill the request.

### What is Rest (REpresentational State Transfer)

It is an architectural style that defines a set of constraints to be used for creating web services or API. Web services which are built using REST are called REST API or RESTFul API or web services. These web services provide interoperability(Communication) between computer systems on the Internet. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations.

### What are principles of REST

***Client–server*** ***architecture:*** You request for available flight details from an Airlines provider on internet. At high level you are a client who is requesting data from an Airlines provider (Server) over HTTP. It is called Client-server architecture. It separates Client and Server related things. You can request airlines details from multiple platforms. Separating user interface concerns from the data storage concerns improves the portability of the user interface across multiple platforms improve scalability by simplifying the server components.

***Statelessness:*** In client-server architecture, server does not save any information about client between requests. Every request to server from any client must contain all required information so that server can understand and response with proper response. It does not matter if same client is sending request to same resource multiple times. Session state is therefore kept entirely on the client.

**Cacheable:** We know response is returned by server on request of client. This response may be cacheable which helps in reusing data for later equivalent requests. Well-managed caching partially or completely eliminates some client–server interactions, further improving scalability and performance. A REST API should be designed to encourage the storage of cacheable data.

**Uniform interface:** Client-server architecture separates or decouples the the architecture, which enables each part to evolve independently. Changes at either architecture, will not impact others. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

**Layered system:** In Client-server architecture , we generally see two layers but in fact there are so many layers with some responsibilities in between request and response. These Intermediary servers/layers can improve system scalability by enabling load balancing and by providing shared caches. They can also enforce security policies.

**Code on demand:** This is an optional constraints which allows for code or applets or client-side scripts such as JavaScript to be transmitted via the API for use within the application.

### Difference between Rest & Rest API

REST API and RESTFul API are not two different terms. It is same. You may find documents saying if an API follows all six constraints of REST, is called as RESTFul API. But core concept says that any API build using REST principles, is called a REST API.

### What is SOAP

SOAP is a **communication or messaging protocol** as it allows you to exchange information in form of XML between two independent systems or applications over a network i.e. Web Services. It is a XML based protocol as it uses **JAX-WS, which is a Java API for XML web services. SOAP**defines a set of rules for structuring messages that can be used for simple one-way messaging or request-response dialogues.

SOAP is a **platform and language independent** i.e. we can use any programming language and platform to develop web services using SOAP. It also uses other protocols like HTTP, SMTP protocols for receiving and transmitting XML-format messages.

A web service developed using SOAP, sends a request to an end point. This request is constructed using SOAP components such as Envelope, Header, Body and Fault. This request hits an end point and response is sent back to client in xml.

SOAP uses **WSDL (Web Services Description Language)** to describe a web services or says what this web service does in XML which includes Endpoints, Input/Output message format etc.

### Difference between Rest & SOAP

1. SOAP is a protocol to be followed while developing a web services using SOAP while REST is an architectural style which is followed by web services if developed using REST. A REST webservice can use SOAP.
2. SOAP supports XML message format while REST supports plain text, HTML,XML,JSON etc.
3. SOAP is function driven while REST is data driven or resource driven.
4. SOAP can use many protocols like HTTP,SMTP etc while REST uses only HTTP.
5. SOAP is more secure as it provides WS-Security and inbuilt ACID (Atomicity, Consistency, Isolation, Durability) compliances while REST supports HTTPS and SSL which make them less secure.
6. SOAP is complex, less flexible and requires more bandwidth and computing power because of its strict rules and tighter security. REST provides more flexibility and easy to use with less resources and bandwidth.
7. SOAP calls cannot be cached while REST call can be.
8. SOAP uses JAX-WS java API while REST uses JAX-RS.
9. SOAP invoke services using RPC (Remote Procedure Calls) while REST uses HTTP methods.
10. SOAP is a official standard while REST is not.
11. SOAP should be used for banking applications, payment gateways etc while REST can be used for less secure applications, public APIs.
12. SOAP can support Javascript which is difficult while REST provides inbuilt support for Javascript.

### What is Accept & Content-type?

### What is HTTP Headers in API

If you notice, you can see a “Headers” tab in both request and response section. When you click on that tab you can see many key-value pairs.

**Headers” or “HTTP Headers” are key – value pairs which are used by Server and Client to exchange additional information about Request and Response. We can also dat that headers are metadata of Request and Response.**

“Authorization” is a header of request which is automatically added when we pass access token. When we hit Get request, we got response. This response contains 16 headers (Server, Date etc.).

Some important headers are:

1. Authorization
2. Content-ID
3. Content-Length
4. Content-Type
5. Date
6. Connection
7. Accept
8. If-Match
9. If-None-Match
10. Location
11. Range