**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SONY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1.What are the common API Testing Types?**

1.Validation Testing \_**How you Validate API’s**

2.Functional Testing – **Functionality of your API’s are working or not**

3.Load Testing -**JMeter**

4.Runtime/Error Detection **– Integration ,Error Codes ,JSON Response**

5.Security Testing –

**It is imp ,how your api is Authorised ,Pass the Headers ,what are the headers, Username and Passwords**

**Authentication Token and Session Ids**

6.Penetration Testing –

**Again it’s part of Security Testing , it is very imp to check my application behaviour in hacking point of view**

7.Interoperability Testing-

How exactly we are doing integration testing

**2.What is the Procedure to perform API Testing ?**

<https://www.youtube.com/watch?v=BkPzzbc3MIA>

1.What is API ?

2. What is Webservice

3. What is SOAP

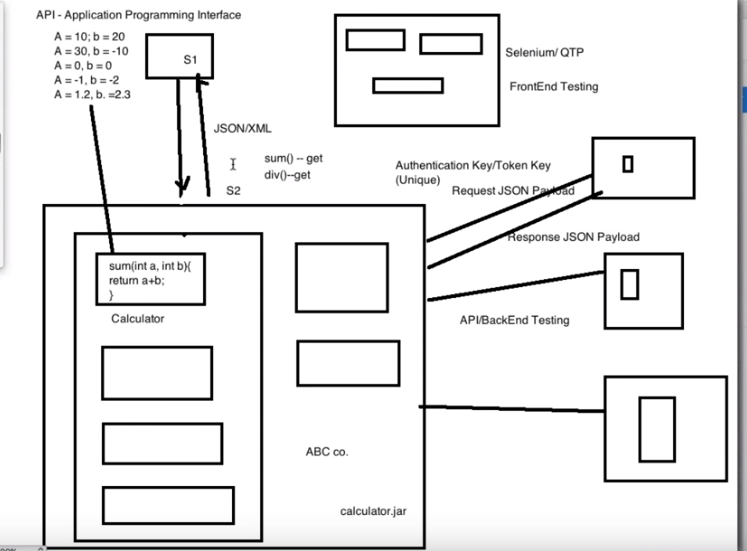
4.What is status code 501 ?

5.What is WSDL

6.What is Collections Post man

7.Difference between SOAP and REST API

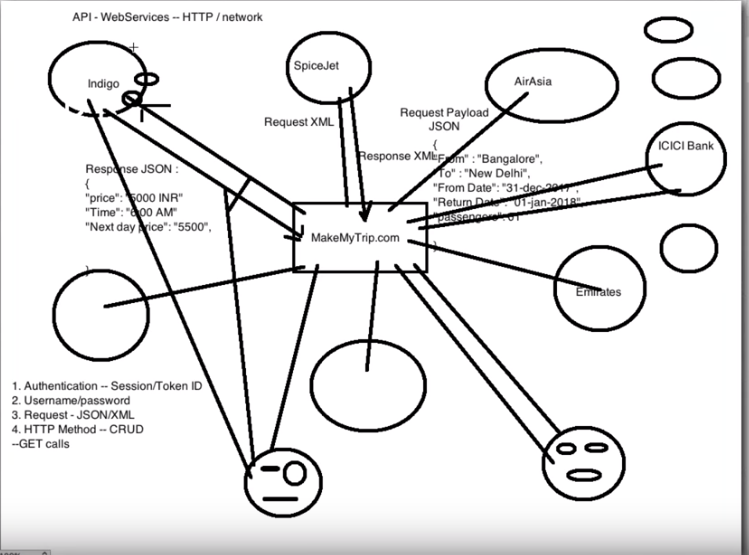
**1.What is API ?**



API – API stand for Application program Interface through which 2 systems s/w can interact each other .And these 2 systems totally independent technology wise, business wise

* API Medium is JSON/XML
* Authentication Key/Token Key - Unique
* Request JSON Payload
* Response JSON Payload
* API is Backend ,there is no UI
* API is jar locally available in the system

2.What is Webservice ?



Webservice -A Method of communication between two devices over the Network/HTTP

🡪Webservice is a language independent way of communication

🡪It is Client Server Application

🡪It is collection of standards or protocols for exchanging information between two devices or application

🡪Require Authentication – Session/Token ID and Username and Password to communicate two different applications

🡪Request JSON/XML

🡪HTTP Method – CRUD

-Get calls

**API** **Webservices**

1.All API’s are not Webservices 1.All API’s Webservices

2.An API doesn’t need any Network for 2.A webservice need Network for it’s Operations it’s Operation

3.It Supports HTTP Protocol :URL Request/Response 3.It Supports only HTTP Protocols

Headers ,Caching, Versioning, Content formats

4.API Use only Style Communication 4.Use only three styles of use : SOAP,REST and XML -RPC for communication

There are 2 Types of Webservices

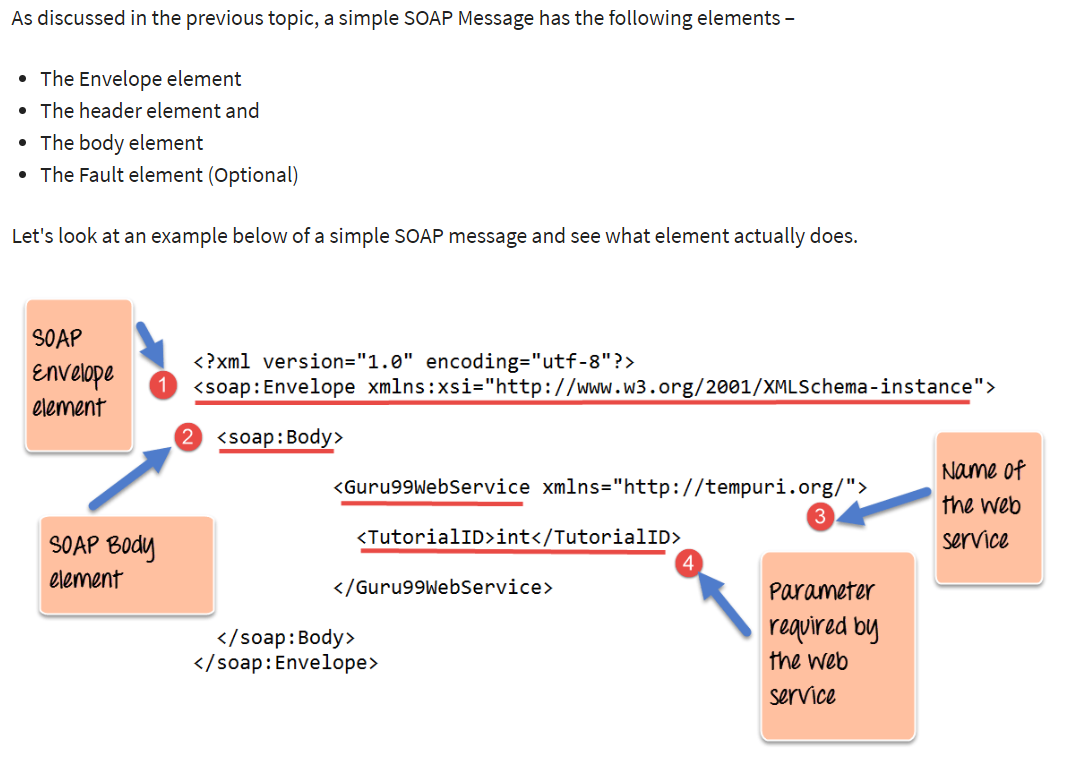
1. SOAP -Simple object Access Protocol
2. REST -Reprasentational State Transfer

SOAP – SOAP is a XML based protocol for accessing webservices over the HTTP. It has some

specification which could be crossed across all the applications .SOAP is Platform Independent and

also is a O.S independent So SOAP protocol can work any programming language based applications

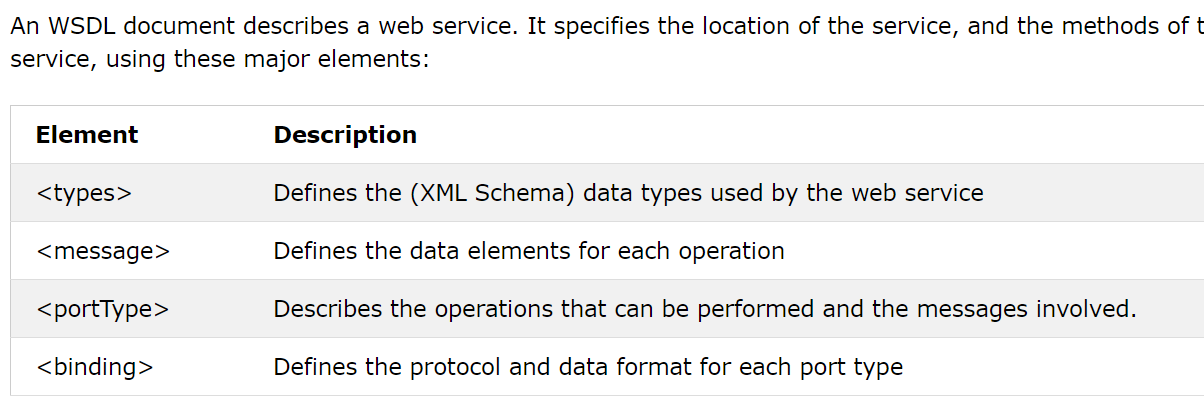
on both windows and Linux .



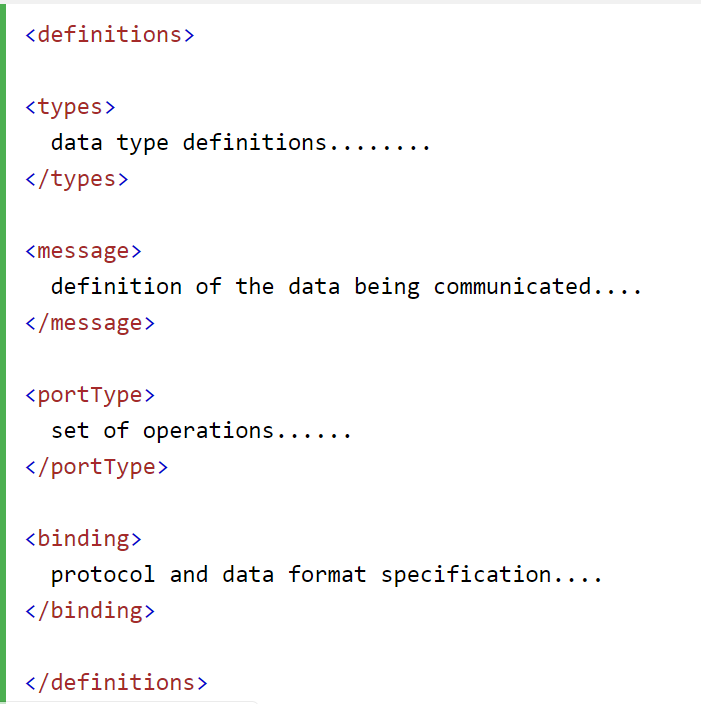
What is WSDL ?

WSDL – Stands for Webservice Description Language ,It is used to describe Webservices , It is

Written in XML

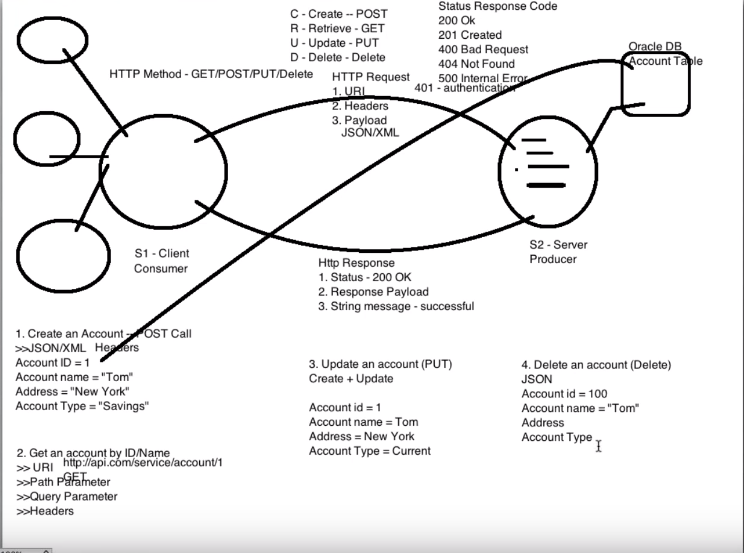


Structure of WSDL :



**What is REST ?**

Any web service that is defined on the principles of REST can be called a RestFul web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components



**Note: Rest Client – POSTMAN , JEMETER , BROWSER , SOAP UI POSTMAN is REST Client**

**What are all HTTP Methods ?**

**1.CRUD**

**CREATE -POST – To create a new entity**

**RETRIVE -GET – To Retrieve the data**

**UPDATE -PUT- To Create a New entity and as well as to update the Existing Entity**

**DELETE- DELETE – To delete the entity**

**OPTIONS -**It indicates the supported techniques.  
 **HEAD** – It returns meta information about the request URL.

**What are all common setup/ parameters required to run the HTTP Method**

* **JSON/XML**
* **HEADERS**
* **REQUEST JSON BODY**
* **PATH PARAMETERS / QUERRY PARAMETERS**
* **PAYLOAD**
* **URI**

**WHAT IS URI?**

**URI -Uniform Resource Identifier**

**URI = End Point URL + API URL**

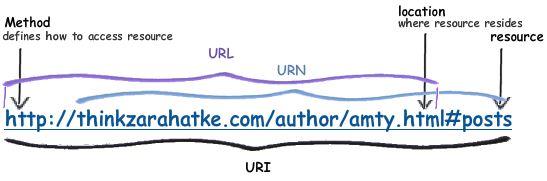
### Q-13. What is URI? Explain its purpose in REST based web services. What is its format?

**Ans.** URI stands for Uniform Resource Identifier. URI is the identifier for the resource in REST architecture.

The purpose of a URI is to locate a resource(s) on the server hosting the web service. A URI is of the following format-

**<protocol>://<service-name>/<ResourceType>/<ResourceID>**

<http://bg-etender-ser:8080/tenderservices/api/document/tender>



**What is Swagger ?**

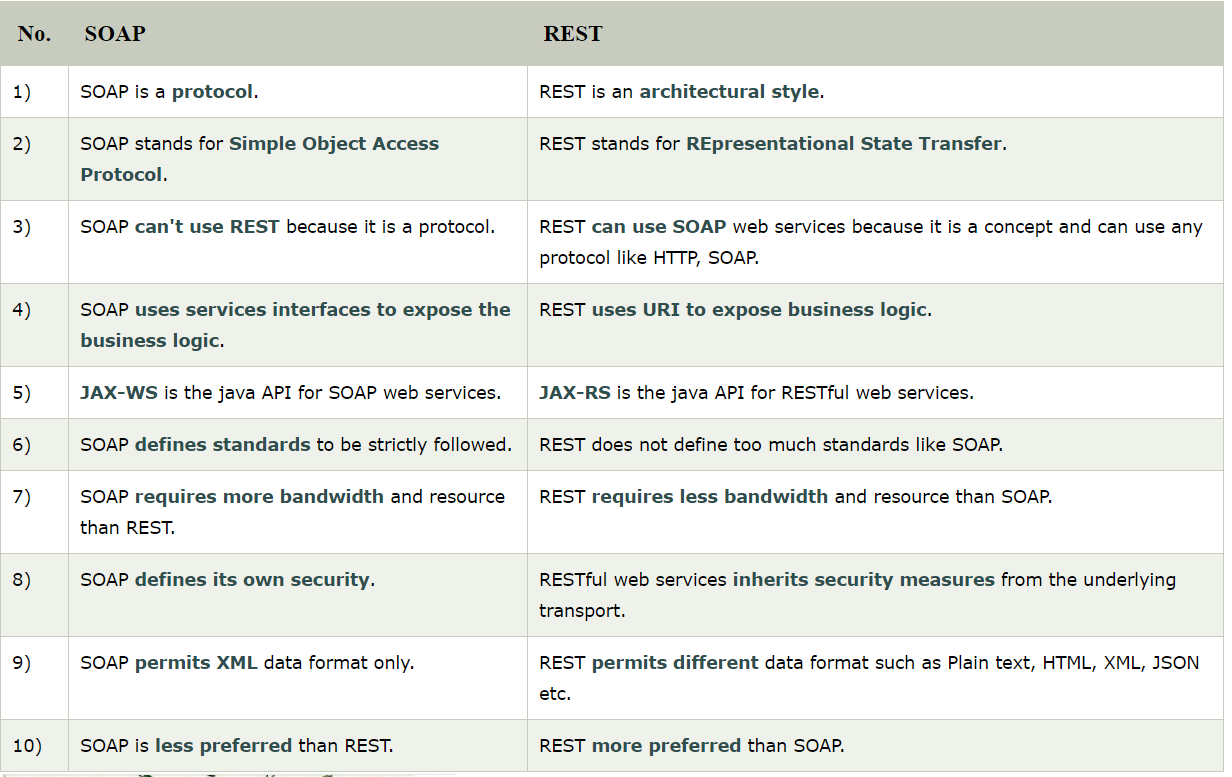
**It is a API documentation or technical Documents or to know what are all API’s available**

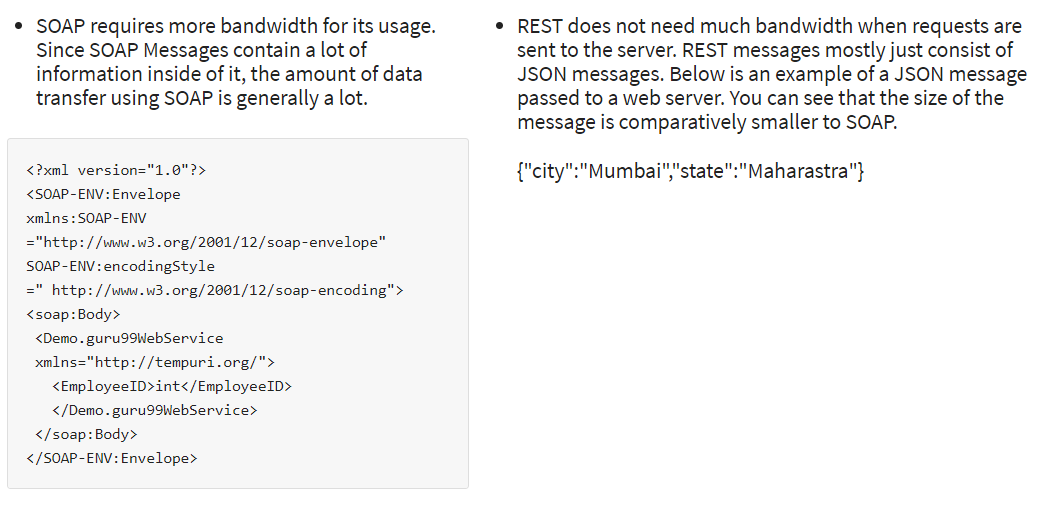
**Q #16) What is Payload?**

The request data which is present in the body part of every HTTP message is referred as ‘Payload’.  In Restful web service, the payload can only be passed to the recipient through POST method.

There is no limit of sending data as payload through POST method but the only concern is that more data with consuming more time and bandwidth. This may consume much of user’s time also.

1. What is difference between SOAP and REST API ?





**Q #20) What are HTTP status codes? Enlist few with meaning.**

HTTP status codes basically are the representation of the status of the task that has been performed on the server, with the mode of some codes. Every code has their own meaning.

---------------------------------------------------------------------------------

200 -OK -This indicates Success / Api Server Response is correct

201 -Created Successfully – Resource has been created successfully

204 – No Content – There is No content in the Response Body

---------------------------------------------------------------------------------

400 – Bad Request - Payload incorrect /no in proper format /Attributes missed in payload

End Point URI is Wrong

404 - Not Found –The Server not found that matches URI Ex:Update the Account that does not Exist

401 – Unauthorized Error -Passing Incorrect Username /Password

500 – Internal Server Error - Net work Issue/ Server id down

**Some of the HTTP status codes with their meaning are as follows:**

* **Code 200:** This indicates success.
* **Code 201:** This indicates resource has been successfully created.
* **Code 204:** This indicates that there is no content in the response body.
* **Code 400:** **Bad Request -**The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications
* **Code 404:** **Not Found -**The server has not found anything matching the Request-URI
* **Code 401: Not Authorized** - The request requires user authentication. The response MUST include a WWW-Authenticate header field
* **Code 403: Forbidden** - The request understood by the server but refusing to fulfil it. Authorisation may not help in this case.

**API:** As Others Said API stands for Application Programming Interface through which software’s can interact with each other. Note it is not a human interaction.

**Where it is used**

An example, you are buying an item in online through your credit card. You will provide credit card details and press continue button. It will tell you

Whether your information is correct or not. To provide these results, there are lot of things in the background.

The application will send your credit card details to a remote application which will validate your information and send the result back your application. API is used in this scenario.

**API Monitoring and Testing**: Getting Started

**API Testing**: A Run scope API test is a group of one or more HTTP requests executed sequentially to evaluate the uptime, performance and correctness of an API.

For each step in the test, you can define Assertions to validate response data and Variables to extract data to be used in subsequent requests.

A test passes if all the assertions pass. A test fails if any assertion fails, or another error is encountered like a network connection problem.

Most API’s not just used by single endpoint, Lot of our work flows involved creating dynamic data, extract dynamic data, passing data b/w the requests

An endpoint is a URL pattern used to communicate with an API.

**Validate response data**

An API may be up and running fast, but if it’s returning the wrong data, then all your customers see is a broken app and a bad experience. Add assertions to your API monitors to make sure your APIs are returning the right data.

:param method: method for the new :class:`Request` object.

:param url: URL for the new :class:`Request` object.

:param params: (optional) Dictionary or bytes to be sent in the query string for the :class:`Request`.

:param data: (optional) Dictionary, bytes, or file-like object to send in the body of the :class:`Request`.

:param json: (optional) json data to send in the body of the :class:`Request`.

:param headers: (optional) Dictionary of HTTP Headers to send with the :class:`Request`.

:param cookies: (optional) Dict or CookieJar object to send with the :class:`Request`.

:param files: (optional) Dictionary of ``'name': file-like-objects`` (or ``{'name': file-tuple}``) for multipart encoding upload.

``file-tuple`` can be a 2-tuple ``('filename', fileobj)``, 3-tuple ``('filename', fileobj, 'content\_type')``

or a 4-tuple ``('filename', fileobj, 'content\_type', custom\_headers)``, where ``'content-type'`` is a string

defining the content type of the given file and ``custom\_headers`` a dict-like object containing additional headers

to add for the file.

:param auth: (optional) Auth tuple to enable Basic/Digest/Custom HTTP Auth.

:param timeout: (optional) How long to wait for the server to send data

before giving up, as a float, or a :ref:`(connect timeout, read

timeout) <timeouts>` tuple.

:type timeout: float or tuple

:param allow\_redirects: (optional) Boolean. Enable/disable GET/OPTIONS/POST/PUT/PATCH/DELETE/HEAD redirection. Defaults to ``True``.

:type allow\_redirects: bool

:param proxies: (optional) Dictionary mapping protocol to the URL of the proxy.

:param verify: (optional) whether the SSL cert will be verified. A CA\_BUNDLE path can also be provided. Defaults to ``True``.

:param stream: (optional) if ``False``, the response content will be immediately downloaded.

:param cert: (optional) if String, path to ssl client cert file (.pem). If Tuple, ('cert', 'key') pair.

:return: :class:`Response <Response>` object

:rtype: requests.Response

**200 OK**

The request has succeeded. The information returned with the response is dependent on the method used in the request, for example:

GET an entity corresponding to the requested resource is sent in the response;

HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body;

POST an entity describing or containing the result of the action;

TRACE an entity containing the request message as received by the end server.

**400 Bad Request**

The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.

Wikipedia

The request cannot be fulfilled due to bad syntax.

General error when fulfilling the request would cause an invalid state. Domain validation errors, missing data, etc. are some examples.

**403 Forbidden**

The server understood the request, but is refusing to fulfil it. Authorization will not help and the request SHOULD NOT be repeated.

If the request method was not HEAD and the server wishes to make public why the request has not been fulfilled,

It SHOULD describe the reason for the refusal in the entity. If the server does not wish to make this information available to the client, the status code 404 (Not found) can be used instead.

**404 Not Found**

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other response is applicable.

Wikipedia

The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.

Used when the requested resource is not found, whether it doesn't exist or if there was a 401 or 403 that, for security reasons, the service wants to mask.

**401 Unauthorized**

The request requires user authentication. The response MUST include a WWW-Authenticate header field (section 14.47) containing a challenge applicable to the requested resource. The client MAY repeat the request with a suitable Authorization header field (section 14.8). If the request already included Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user SHOULD be presented the entity that was given in the response, since that entity might include relevant diagnostic information. HTTP access authentication is explained in "HTTP Authentication: Basic and Digest Access Authentication".

Wikipedia

Similar to 403 Forbidden, but specifically for use when authentication is possible but has failed or not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.

Error code response for missing or invalid authentication token.

Request URL, Response Body, Response Code and Response Headers

**API Monitoring & Testing: Assertions**

Assertions allow you to specify expected data in the response to a request made in a test run. When a test is run, the outcome is determined by whether or not all the assertions pass. If any assertion fails, the test fails. Assertions can be made against response header values, status code, response time/size, and content (like JSON or XML).

## API Monitoring & Testing: Test Steps

API tests are comprised of a series of steps, most often HTTP requests. In addition to requests, you can also add additional types of steps to your tests like pauses and conditions.

The **Editor** is where you'll define the steps (HTTP requests, pauses, etc.) and execution order that make up a test. For each request in a test, you can specify the HTTP request data, [assertions](https://www.runscope.com/docs/api-testing/assertions), [variables](https://www.runscope.com/docs/api-testing/variables) and [scripts](https://www.runscope.com/docs/api-testing/scripts) by clicking on a request.

#### Request Lifecycle

When a request step is executed each of the associated assertions, variables and scripts will be processed. The execution order is as follows:

1. Pre-request Scripts are executed. The variable context from initial/variables and scripts and previous steps are available via variables. Get ().
2. The HTTP request is executed and a response is returned.
3. Variables defined in the editor are processed on the response.
4. Post-response Scripts are processed. Initial and request-specific variable values extracted from previous steps are available for use.
5. Assertions defined in the test editor are processed on the response. If the response object was modified by a Post-response Script, the data is not available to be evaluated by an Assertion.

### Q-1. Explain REST?

**Ans.** REST stands for Representational State Transfer. REST is an architectural style of developing web services.

In REST architecture, a REST Server provides access to resources and REST client accesses and presents those resources. Here each resource is identified by URIs or global IDs. REST uses different ways to represent a resource like text, JSON, and XML.XML and JSON are the most popular representations of resources these days.

### Q-2. Explain the RESTFul Web Service?

**Ans.** Mostly, there are two kinds of Web Services which are quite popular.

**1.** SOAP (Simple Object Access Protocol) which is an XML-based way to expose web services.

**2.** Web services developed using REST style are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation such as JSON and set of HTTP Methods.

### Q-4. What is the most popular way to represent a resource in REST?

**Ans.** REST uses different representations to define a resource like text, JSON, and XML.

XML and JSON are the most popular representations of resources.

### Q-5. Which protocol is used by RESTful web services?

**Ans.** RESTful web services make use of HTTP protocol as a medium of communication between client and server.

### Q-7. State the core components of an HTTP Request?

**Ans.** Each HTTP request includes five key elements.

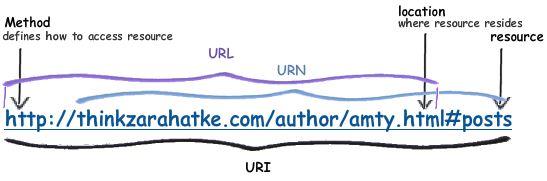
**1.** The Verb which indicates HTTP methods such as GET, PUT, POST, DELETE.  
**2.** URI stands for Uniform Resource Identifier (URI).It is the identifier for the resource on the server.  
**3.** HTTP Version which indicates HTTP version, for example-HTTP v1.1.  
**4.** Request Header carries metadata (as key-value pairs) for the HTTP Request message. Metadata could be a client (or browser) type, the format that client supports, message body format, and cache settings.  
**5.** Request Body indicates the message content or resource representation.

### Q-8. State the core components of an HTTP response?

**Ans.** Every HTTP response includes four key elements.

**1.** Status/Response Code – Indicates Server status for the resource present in the HTTP request. For example, 404 means resource not found and 200 means response is ok.  
**2.** HTTP Version – Indicates HTTP version, for example-HTTP v1.1.  
**3.** Response Header – Contains metadata for the HTTP response message stored in the form of key-value pairs. For example, content length, content type, response date, and server type.  
**4.** Response Body – Indicates response message content or resource representation.

* URI (uniform resource identifier) identifies a resource (text document, image file, etc)
* URL (uniform resource locator) is a subset of the URIs that include a network location
* URN (uniform resource name) is a subset of URIs that include a name within a given space, but no location



### Q-9. Name the most commonly used HTTP methods supported by REST?

**Ans.** There are a few HTTP methods in REST which are more popular.

**1.** GET -It requests a resource at the request-URL. It should not contain a request body as it will get discarded. Maybe it can be cached locally or on the server.  
**2.** POST – It submits information to the service for processing; it should typically return the modified or new resource.  
**3.** PUT – At the request URL it updates the resource.  
**4.** DELETE – It removes the resource at the request-URL.  
**5.** OPTIONS -It indicates the supported techniques.  
**6.** HEAD – It returns meta information about the request URL.

**Q #17) Enlist some of the HTTP methods with description.**

Mentioned below is the list of HTTP methods with their descriptions:

* **GET:** This is a read only operation which fetches the list of users on the server.
* **PUT:** This operation is used for the creation of any new resource on the server.
* **POST:** This operation is used for updating an old resource or for creating a new resource.
* **DELETE:** As the name suggests, this operation is used for deleting any resource on the server.
* **OPTIONS:** This operation fetches the list of any supported options of resources that are available on the server.

### Q-10. Mention, whether you can use GET request instead of PUT, to create a resource?

**Ans.** No, you shouldn’t use a PUT or POST method. Instead, apply the GET operation which has view-only rights.

### Q-11. Is there any difference between PUT and POST operations? Explain it.

**Ans.** PUT and POST operation are almost same. The only difference between the two is in the terms of the result generated by them.

PUT operation is idempotent while POST operation can give a different result.

Let’s take an example.

**1.** PUT puts a file or resource at a particular URI and exactly at that URI. If the resource already exists, then PUT updates it. If it’s a first-time request, then PUT creates one.  
**2.** POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource.

**Q #18) what is the difference between PUT method and POST method?**

The major difference between the PUT and POST method is that the result generated with PUT method is always same no matter how many times the operation is performed. On the other hand, the result generated by POST operation is always different every time.

### Q-13. What is URI? Explain its purpose in REST based web services. What is its format?

**Ans.** URI stands for Uniform Resource Identifier. URI is the identifier for the resource in REST architecture.

The purpose of a URI is to locate a resource(s) on the server hosting the web service. A URI is of the following format-

**<protocol>://<service-name>/<ResourceType>/<ResourceID>**

<http://bg-etender-ser:8080/tenderservices/api/document/tender>

### Q-14. What do you understand by payload in RESTFul web service?

**Ans.** Request body of every HTTP message includes request data called as Payload. This part of the message is of interest to the recipient.

We can say that we send the payload in POST method but not in <GET> and <DELTE> methods.

**Q #16) What is Payload?**

The request data which is present in the body part of every HTTP message is referred as ‘Payload’.  In Restful web service, the payload can only be passed to the recipient through POST method.

There is no limit of sending data as payload through POST method but the only concern is that more data with consuming more time and bandwidth. This may consume much of user’s time also.

### Q-17. List the main differences between SOAP and REST?

**Ans.**

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| **1.** SOAP is a protocol through which two computer communicates by sharing XML document. | **1.** Rest is a service architecture and design for network-based software architecture. |
| **2.** SOAP supports the only XML format. | **2.** It supports many different data formats. |
| **3.** SOAP does not support caching. | **3.** It supports caching. |
| **4.** SOAP is like custom desktop application, closely connected to the server. | **4.** A REST client is just like a browser and uses standard methods. An application has to fit inside it. |
| **5.** SOAP is slower than REST. | **5.** It is faster than SOAP. |
| **6.** It runs on HTTP but envelopes the message. | **6.** It uses the HTTP headers to hold meta information. |

### Q-15. What is the upper limit for a payload to pass in the POST method?

**Ans.** <GET> appends data to the service URL. But, its size shouldn’t exceed the maximum URL length. However, <POST> doesn’t have any such limit.

So, theoretically, a user can pass unlimited data as the payload to POST method. But, if we consider a real use case, then sending POST with large payload will consume more bandwidth. It’ll take more time and present performance challenges to your server. Hence, a user should take action accordingly.

### Q-18. What are the tools available for testing web services?

**Ans.** Following tools can help in testing the SOAP and RESTful web services.

**1.** SOAP UI tool.  
**2.** ARC (Advance REST Client)

**3.** The Postman extension for Chrome.

**Q #10) What is a ‘Resource’?**

Just like the ‘Object’ instance, we have learned in object orient programming Language, in the same way, ‘Resource’ is defined as an object of a type which can be an image, HTML file, text data, and any type of dynamic data. There are varieties of representation formats available in order to represent a resource.

**Some most common are enlisted below:**

* JSON
* YAML
* XML
* HTML

**Q #11) Why proper representation of Resource is required?**

Representation is very important because it determines the easy identification of resources. With proper representations of resource in the proper format, allows the client to easily understand the format.

**Q #12) Enlist some important points that should be kept in mind while designing Resources representation for RESTful web services.**

As there are no restrictions on the format in which the resource representation is done but just that the main requirement is the format of the representation should be as per the client requirement. A good resource representation is designed by considering the following main points:

* The resource representation format should be easily understood by the client and server.
* The representation should be complete regardless of its format structure, which may be complex or simple.
* In the case of the link of the resources to other resources, such cases should also be considered and handled.

**Q #20) What are HTTP status codes? Enlist few with meaning.**

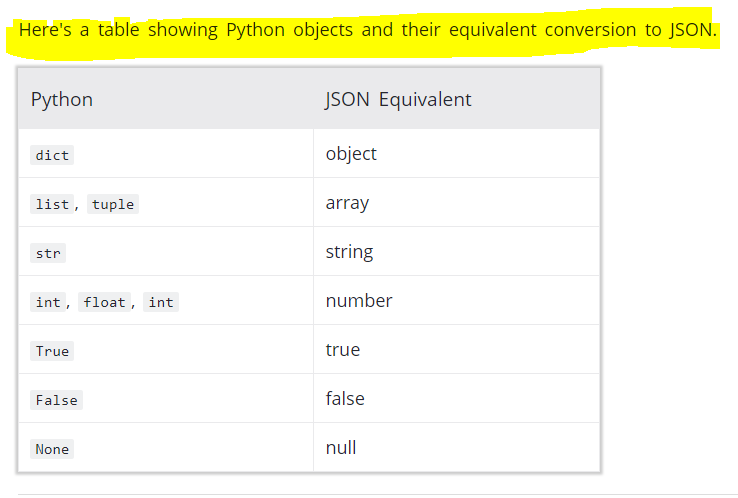
HTTP status codes basically are the representation of the status of the task that has been performed on the server, with the mode of some codes. Every code has their own meaning.

**Some of the HTTP status codes with their meaning are as follows:**

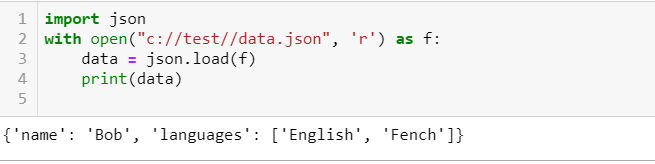
* **Code 200:** This indicates success.
* **Code 201:** This indicates resource has been successfully created.
* **Code 204:** This indicates that there is no content in the response body.
* **Code 400:** **Bad Request -**The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications
* **Code 404:** **Not Found -**The server has not found anything matching the Request-URI
* **Code 401: Not Authorized** - The request requires user authentication. The response MUST include a WWW-Authenticate header field
* **Code 403: Forbidden** - The request understood by the server but refusing to fulfil it. Authorisation may not help in this case.

405:  Method Not Allowed. The method received in the request-line is known by the origin server but not supported by the target resource. The origin server MUST generate an Allow header field in a 405 response containing a list of the target resource's currently supported **methods**.

305: general. The 305 response from the Web server should always include an alternative **URL** to which **redirection** should occur. If it does, a Web browser will immediately retry the alternative **URL**.

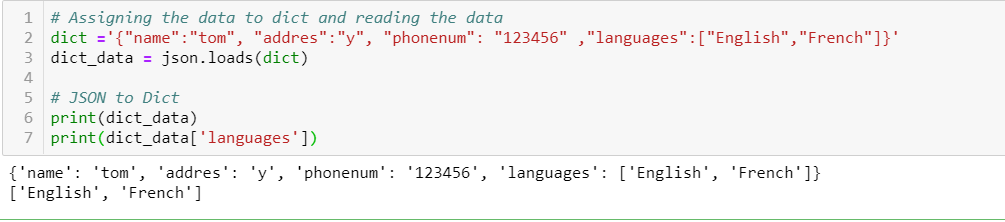


# To Read json data from a File “JSON.LOAD”



CONVERT THE JSON TO PYTHON

If you have a JSON string, you can parse it by using the json.loads() method.

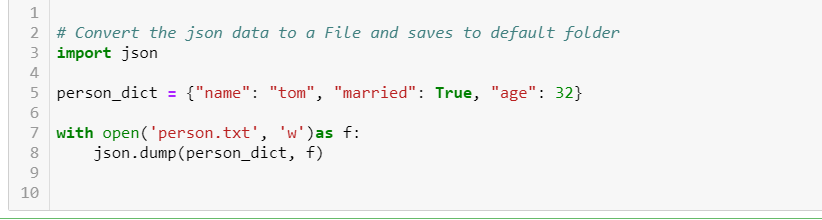


CONVERT PYTHON TO JSON

If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.



CONVERT JSON TO FILE :



Convert Python objects into JSON strings, and print the values:

