**IT Software Quality : API Test**

|  |  |
| --- | --- |
| **Author:** | Fahimeh Shafiei |
| **Date:** | 2 May 2023 |
| **Version:** | 1.0 |

What is **API**?

An API is an Application Programming Interface. This is an interface to an application  
designed for other computer systems to use. As opposed to a Graphical User Interface (GUI)  
which is designed for humans to use.

What Is a **URL**?  
URL is a Uniform Resource Locator and is the address we use to access websites and web  
applications.

When we want to call an HTTP API we need the URL for the **endpoint** we want to call e.g  
http://compendiumdev.co.uk/apps/mocktracks/projectsjson.php

What above URL says?

This is the locator that says “I want to call the apps/mocktracks/projectsjson.php resource  
located at compendiumdev.co.uk using the http protocol”

The above URL can be broken down into the form:  
scheme://host/resource

• scheme - http  
• host - compendiumdev.co.uk  
• resource - apps/mocktracks/projectsjson.php

When working with a Web Application or HTTP API the typical HTTP Verbs used are:

* GET, to read information.
* POST, to create information.
* PUT, to amend or create information.
* DELETE, to delete information, this is rarely used for Browser accessed applications,  
  but often used for HTTP APIs.

POST and PUT requests would usually have a message body. GET and DELETE would not.

If you want to test DELETE, please **communicate with business first, in PIT environment specially**.

HTTP Status Codes

The simple grouping for HTTP Status Codes is:  
• 1xx - Informational  
• 2xx - Success e.g. 200 Success  
• 3xx - Redirection e.g. 302 Temporary Redirect  
• 4xx - Client Error e.g. 400 Bad Request, 404 Not Found  
• 5xx - Server Error e.g. 500 Internal Server Error

What are payloads?

A Payload is the body of the HTTP request or response.

Typically, when working with an HTTP API we will send and receive **JSON** or **XML** payloads.

What is authentication?

When we send a message to a server we might need to be authenticated i.e. authorized to  
send a message and receive a response.

If you are not authenticated and try to send a message to a server then you are likely to  
receive a response from the server with a 4xx status code e.g.  
• 401 Unauthorized  
• 403 Forbidden

So, one of the test cases is above we called negative test, to verify when your are not approved to call API based on authentication key you have to get range 4xx as response.

What is REST API?

REST stands for Representational State Transfer. it very often means that the API will respond to  
HTTP verbs as commands.

* GET, to read information.
* POST, to create information.
* PUT, to amend information.
* DELETE, to delete information.
* PATCH, is used to modify the values of the resource properties.

**Stateless VS stateful**

A stateless system sends a request to the server and relays the response (or the state) back without storing any information. On the other hand, stateful systems expect a response, track information, and resend the request if no response is received.

| **Characteristics** | **Stateful** | **Stateless** |
| --- | --- | --- |
| Definition | Stateful Protocols require the server to save the state of a process. | Stateless Protocols do not need the server to save the state of a process. |
| Response mechanism | Stateful expects a response and if no answer is received, the request is resent. | In stateless, the client sends a request to a server, which the server responds to based on the state of the request. |
| Dependency | Server and Client are tightly coupled, as in extremely interdependent on each other. | Server and Client are more independent and hence, loosely coupled. |

Transfer protocol:

* HTTP(s)
* Rarely SNMP, SMTP

Test Design and coverage:

* What to test?
* How to test?
* What coverage is enough?
  + Functional test, all functionality must test and verified.
  + Non functional test is out of scope this document.

Rest API includes, and we test:

* Request between rest client and rest servers
* Response between rest client and rest servers
* Endpoints for example
  + Examples:  
    http://example.com/api/devices  
    http://example.com/api/devices/sonyz3  
    http://example.com/api/users  
    http://example.com/api/products  
    http://example.com/api/products/12345  
    <http://example.com/api/products/search?q=name:Key>
* Which end points your webservices provides check this with business and your SRS document?
* Each collection of end points needs to be tested, some of them will integrated to each other please check integration.
* For each resource type at least one end point needs to be tested.
* Test all positive tests for REST API and verify test result.
* Do some negative tests:
  + Such as request end point that doesn’t exist.
  + For post and put test with invalid values/wrong data type/empty object/empty string/filed value is null/required filed is absent/
  + Post already existing resource.
  + For Delete, test and execute to delete non-existing resource.
* If there were endpoints with search, filtering and sorting it needs to be verified by
  + Supported parameters.
  + Boundary values.
  + Special characters.
  + Max and minimum length of parameters.
* Negative test: try to request for search, filtering and sorting with wrong parameters or incorrect values.
* Test all request methods that each end point has it contain
  + Get
  + Post
  + Put
  + Delete
  + Patch
* Test with concurrent users for DELETE and GET request method.
* For test header of request do below negative tests:
  + Test Rest API in case of missing required header, one at time.
  + Test Rest API in case of wrong and unsupported or empty header value.
* Response code tests:
  + Check and verify which response code return based on request type, and make sure there are logically related with events.
    - 200 ok when we GET to retrieve information
    - 401 (Unauthorized) when we try to access information without the correct credentials.
    - 404 (Not Found) when we GET information that does not exist, or POST, PUT DELETE to an end point that does not exist.
    - 201 (created) when we use POST to create a new entity
    - 409 (Already Exists) we try to create an existing entity
    - 302 (Redirect) when we successfully issue a request, the system may redirect us.
* Response Body tests:
  + Verify structure of a response/Fields/Values/Data Types
* Rate limits and cashing
  + Verify cached Reponses received faster.
  + Verify cache expiry time.
  + Find out rate limits for different methods or end point.
  + Try to produced max allowed rate limit
  + Try to produce more than max allowed rate limit.
* Do some independent test:
  + Post object in one test, get and verify in another test time.
  + Add embedded object in one test and verify in another test time.

**Common API test cases that can be used to test various APIs:**  
  
1.    Verify the API response code for a valid request.  
2.    Verify the API response code for an invalid request.  
3.    Verify the API response time for a valid request.  
4.    Verify the API response time for an invalid request.  
5.    Verify the API response message for a valid request.  
6.    Verify the API response message for an invalid request.  
7.    Verify the API response content for a valid request.  
8.    Verify the API response content for an invalid request.  
9.    Verify the API response format for a valid request.  
10.  Verify the API response format for an invalid request.  
11.  Verify the API response size for a valid request.  
12.  Verify the API response size for an invalid request.  
13.  Verify the API response headers for a valid request.  
14.  Verify the API response headers for an invalid request.  
15.  Verify the API request method for a valid request.  
16.  Verify the API request method for an invalid request.  
17.  Verify the API request parameters for a valid request.  
18.  Verify the API request parameters for an invalid request.  
19.  Verify the API request payload for a valid request.  
20.  Verify the API request payload for an invalid request.  
21.  Verify the API authentication for a valid request.  
22.  Verify the API authentication for an invalid request.  
23.  Verify the API authorization for a valid request.  
24.  Verify the API authorization for an invalid request.  
25.  Verify the API rate limiting for a valid request.  
26.  Verify the API rate limiting for an invalid request.  
27.  Verify the API error handling for a valid request.  
28.  Verify the API error handling for an invalid request.  
29.  Verify the API security for a valid request.  
30.  Verify the API security for an invalid request.  
31.  Verify the API response time under high load.  
32.  Verify the API response time under low load.  
33.  Verify the API response for edge cases (empty, null, negative values, etc.).  
34.  Verify the API response when input is out of range.  
35.  Verify the API response when input is invalid.  
36.  Verify the API response when required fields are missing.  
37.  Verify the API response when optional fields are missing.  
38.  Verify the API response when the input is too long.  
39.  Verify the API response when input is too short.  
40.  Verify the API response when input contains special characters.  
41.  Verify the API response when input contains non-ASCII characters.  
42.  Verify the API response when input is case-sensitive.  
43.  Verify the API response when input is case-insensitive.  
44.  Verify the API response when input contains HTML or JavaScript tags.  
45.  Verify the API response when input contains SQL injection.  
46.  Verify the API response when input contains a cross-site scripting (XSS).  
47.  Verify the API response when input contains malware or viruses.  
48.  Verify the API response when input contains sensitive information  
(C&P)