REST WebServices testing guidelines

Table of Contents

**API**

1. Public api

2. Private / Secure api

○ Authentication

○ Authorization

**Methods**

1. Get

○ With Parameter

○ Without parameter

○ Query parameter

2. Post

○ Positive cases

○ Negative cases

3. Put

○ Positive cases

○ Negative cases

4. Delete

○ Positive cases

○ Negative cases

**Response body**

1. JSON

2. HTML

3. Text

**Response code**

1. 200

2. 201

3. 204

4. 401

5. 403

6. 404

7. 500

**Request header**

1. Content-type

2. Authorization

**Application Programming Interface**.

API is a set of routines, protocols, and tools for building Software Applications. APIs specify how one software program should interact with other software programs.

Public API

Open API is public API.

There is no security mechanism like authentication/authorization provided.

For example, Microsoft makes the Microsoft Windows API public, and Apple releases its APIs Carbon and Cocoa

Example -

Base Url : https://reqres.in/

Method: GET

Description**:** Get list of users

Request URL**:** api/users

Parameters**:** Id (In number format-optional)

Positive case: Response

● Succeeded:

**○** HTTP STATUS**:** 200 OK.

**■** Without Parameter ( /api/users)

**■** With Parameter ( /api/users/2 )

**■** With Query parameter or condition ( /api/users?page=2)

{

“Param\_name” : “Param\_Value”,

}

Private / Secure api

The API is for internal company use only.

Security mechanisms are applied for Private Apis like authentication/authorization.

○ Authentication

verification of the **credentials** of the connection attempt. OAuth, JWT, and Basic Auth all uses headers for transmitting credentials

○ Authorisation

verification that the connection attempt is **allowed**. Authorization occurs after successful authentication

Example :

**Step 1** : CREATE Authentication token/Access token

Base URL for Private Api: https://testmanage.my.imagine.com

Method: POST

Description**:** Create new auth token

Request URL: /Token

Header : Content-Type Text

Fields**:**

● grant\_type

● username

● password

● Client\_id (To identify the application)

● client\_secret : optional (Serves as a password of application)

Text Payload:

grant\_type=password&username=xyz@abc.com&password=abc&client\_id=FTPI mport&client\_secret=abcdxyz

Possible test scenarios :

Positive case **(** Response **:** succeeded HTTP STATUS**:** 200 OK.) 1. Access token retrieved successfully

**○** Example

{

"**Access\_token**": “**eyJhbGciOiJSUzI1NiIsIm**",

"expires\_in": 57600,

"token\_type": "Bearer",

"refresh\_token": "e8ef83db218420759eae7c149836b445cc342",

"id": "0316c8cb-94-a81c01227112",

"username": "xyz@abc.com", }

Negative cases ( HTTP STATUS**:** 400 Bad Request)

1. If fields values are invalid

❖ If username or password values are invalid

❖ If grant type field is not provided

❖ If client id provided with null or invalid value

❖ If client secret is provided for invalid environment

❖ If null/invalid value provided for grant type

❖ If typo mistakes present in the request body for field names.

2. If content type is wrong

❖ If content type is not acceptable like xml or some other encoded format ❖ If no content type provided

❖ If additional not required header added

**○** Examples:

{

"error": "invalid\_grant",

"error\_description": "unknown\_username"

}

**Step 2 :** Retrieve /Get user profile

Description**:** Get/view user profile

Request URL**:** /v2.0/groups

Headers: Content type : application/json

Authorization : bearer <Authentication token from token Api>

Parameters**:** Id

Possible test scenarios :

Positive case ( Response : succeeded HTTP STATUS**:** 200 OK.) Example :

**■** List specific group by id (/v2.0/groups/id)

{

"id": "unique value",

"name": "nameValue",

“Other\_fields” : “Other\_fields’s value”

}

Methods

**Get**

**With Parameter**

Description**:** Get/view user profile

Request URL**:** /v2.0/groups/id

Headers : Content type : application/json

Authorization : bearer <Authentication token from token Api> Parameters**:** Id

Possible test scenarios

Positive case ( Response : succeeded HTTP STATUS**:** 200 OK.) 1. List specific group by id (/v2.0/groups/id)

{

"id": "unique value",

"name": "nameValue",

“Other\_fields” : “Other\_fields’s value”

}

Negative cases

1. Invalid id of group passed in the url

2. If null or zero value provided for id

3. Invalid token provided with less or more length

4. Invalid token provided with same length

5. Same user generated but expired token provided

6. Other-user active token provided

7. No authentication token provided

**Without Parameter**

Description**:** Retrieve /Get list of user profiles

Request URL**:** /v2.0/groups

Headers : Content type : application/json

Authorization : bearer <Authentication token from token Api> Parameters**:** None

Possible test scenarios :

Positive case ( Response : succeeded HTTP STATUS**:** 200 OK.) 1. List of all the groups

List of groups (/v2.0/groups)

[{

"id": "unique value",

"name": "nameValue",

“Other\_fields” : “Other\_fields’s value”

}

{

"id": "unique value",

"name": "nameValue",

“Other\_fields” : “Other\_fields’s value”

}]

Negative case

1. Invalid token provided with less or more length

2. Invalid token provided with same length

3. Same user generated but expired token provided

4. Other-user active token provided

5. No authentication token provided

**With query parameters**

Description**:** Get/view user profile

Request URL**:** v2.0/groups?additionalFields=studentCount&additionalFields=users&desc=false Headers : Content type : application/json

Authorization : bearer <Authentication token from token Api>

Parameters**:** None

Possible test scenarios :

Positive case ( Response : succeeded HTTP STATUS**:** 200 OK.)

1. Valid values for parameter additionalFields only

2. Valid values for parameter desc only

List all the groups

[ {

"id": "someUniqueValue",

"name": "value"

},

{

"id": "someUniqueValue",

"name": "value"

}]

Negative cases :

1. Invalid values for parameter additionalFields only

2. Invalid values for parameter desc only

3. Null or zero values for parameter additionalFields or desc

4. Invalid token provided with less or more length

5. Invalid token provided with same length

6. Same user generated but expired token provided

7. Other-user active token provided

8. No authentication token provided

**○** HTTP STATUS**:** 404 Not Found

**■** In case of invalid parameters. (/v2.0/groups/23434)

**○** If invalid id is provided in url

**○** If null or zero value is provided in the parameter

Response - The resource you are looking for has been removed, had

its name changed, or is temporarily unavailable.

**○** HTTP STATUS: 401 Unauthorized

**■** In case of invalid/ expired/ no/ other-user authentication token

{

"message": "Authorization has been denied for this request."

}

**Post**

Function : Creates new resource.

Description**:** Create/Add new group

Request URL: /v2.0/groups

Headers : Content type : application/json

Authorization : bearer <Authentication token from token Api>

Fields**:**

● Name (String value)

● organizationId (Unique String value)

Json Input Payload:

{

"Name": "SampleGroupName1",

"organizationId" : "uniqueIdValue"

}

Possible test scenarios

Positive case ( Response : succeeded HTTP STATUS**:** 201 Created or 200 Ok ) 1. Valid json input payload and valid token provided

2. Valid optional parameters information provided in input payload

Negative cases

1. Invalid fields length (less than or greater than) provided to name or organization id 2. Null or zero value provided to fields name and/or organization id

3. Either name or organization id is missing in payload

4. Duplicate value for name or organization id present

5. Invalid token provided with less or more length

6. Invalid token provided with same length

7. Same user generated but expired token provided

8. Other-user active token provided

9. No authentication token provided

10. Empty or text payload provided in request

11. Input payload added with numeric values instead of string

12. Organization id of other environment is provided.

**○** HTTP STATUS**:** 400 Bad Request

**■** If length of field name is invalid (less than or greater than ideal

length)

{

"message": "Invalid Group(<group\_id>).\r\n StringLength: Name[KKhgdfsjh]

string length out of range (1-256).\r\n"

}

**■** If content type is invalid (other than application/json)

{ "message": "Request body was empty or invalid" }

**■** If mandatory parameters not provided ( like if organization id not provided here)

{

"message": "Invalid GroupRequest(<group\_id>).\r\n MissingField:

OrganizationId is required.\r\n"

}

**■** If **null/empty** value provided for mandatory parameter

(“name”= null -> in json request body)

{

"message": "Invalid GroupRequest(<group\_id>).\r\n MissingField: Name is required.\r\n"

}

**○** HTTP STATUS**:** 401 Unauthorized

**■** If authentication token is not provided

**■** If expired authentication token provided

**■** If wrong value of authentication token provided

**■** If authentication token of other-user/other-environment is used.

{

"message": "Authorization has been denied for this request."

}

**○** HTTP STATUS: 409 Conflict

**■** If same field value provided for unique parameter while adding group in input payload (same id provided here in input payload)

{

"message": "Invalid Group(<group\_id>).\r\n DuplicateKey: index: \_id\_ dup key: { : \"<group\_id>\" }\r\n"

}

**○** HTTP STATUS: 403 Forbidden

**■** If authorization is not provided properly ( If Admin does not have access rights to add organization)

{

"message": "Administrator does not have rights to the parent of the new organization"

}

**○** HTTP STATUS: 500 Internal Server Error

**● 500 Internal Server Error for any API response is always a Bug**

**Put**

Function : Updates the resource.

Description**:** Update existing group

Request URL: /v2.0/groups/<id of group to update>

Headers : Content type : application/json

Authorization : bearer <Authentication token from token Api>

Parameters**:** Id

Fields:

● name: string

● OrganizationId : unique string

Request/Input Payload:

{

"name":"UpdatingGroupName",

"organizationId":"unique org id value"

}

Possible test scenarios

Positive cases ( Response : succeeded HTTP STATUS**:** 204 No Content or 200 Ok ) 1. Valid values of name and organization id provided

2. Optional parameters values added in the input payload

Negative cases:

● Invalid Fields

1. Invalid fields length (less than or greater than) provided to name or organization id 2. Only name field value or only organization field value provided in input payload 3. Invalid heading (typo) added for field name of organization id.

4. Unauthorized user tries to update name with put request.

5. Updated value is not reflecting - when verified with another get request. 6. Null or zero value provided to fields name and/or organization id 7. Duplicate value for name or organization id present

8. Empty or text payload provided in request

9. Input payload added with numeric values instead of string

10. Organization id of other environment is provided.

● Invalid token

1. Invalid token provided with less or more length

2. Invalid token provided with same length

3. Same user generated but expired token provided

❖ Expired token to be retrieved probably through logout api

❖ Keep the existing token for long time and then use

4. Other-user’s active token (who does not have proper access rights) provided

5. No authentication token provided

6. Null or other environment’s token provided

**○** HTTP STATUS: 404 Not Found

**●** If wrong value added as parameter. (invalid group id value)

Response - The resource you are looking for has been removed, had its name

changed, or is temporarily unavailable.

**○** HTTP STATUS: 401 Unauthorized

**●** In case of invalid/ expired/ no/ Other\_user authentication token

{ "message": "Authorization has been denied for this request." }

**○** HTTP STATUS: 415 Unsupported Media Type

**●** If provided with invalid content type like application/atom+xml

{"message": "The request entity's media type 'application/atom+xml' is not

supported for this resource.", "exceptionMessage": "No MediaTypeFormatter is

available to read an object of type 'GroupRequest' from content with media type

'application/atom+xml'.","exceptionType":

"System.Net.Http.UnsupportedMediaTypeException","stackTrace": " at

System.Net.Http.HttpContentExtensions.ReadAsAsync[T](HttpContent content,

Type type, IEnumerable`1 formatters, IFormatterLogger formatterLogger,

CancellationToken cancellationToken)\r\n at

System.Web.Http.ModelBinding.FormatterParameterBinding.ReadContentAsync(

HttpRequestMessage request, Type type, IEnumerable`1 formatters,

IFormatterLogger formatterLogger, CancellationToken cancellationToken)" }

**Delete**

Description: Delete existing group

Request URL: /v2.0/groups/<Group id to be deleted>

Headers : Content type : application/json

Authorization : bearer <Authentication token from token Api>

Parameters**:** Id

Possible test scenarios

Positive case ( Response : succeeded HTTP STATUS**:** 204 No Content or 200 Ok ) 1. Valid value of parameter id provided

Negative cases:

1. Invalid value for parameter id provided

2. Invalid token provided with less or more length

3. Invalid token provided with same length

4. Same user generated but expired token provided

5. Other-user active token provided

6. No authentication token provided

7. Invalid content type or additional content type added

8. Already deleted id entered in the url for delete request.

**○** HTTP STATUS: 404 Not Found

**■** In case of wrong content type provided (Other than

application/json)

**■** In case of non existing group id provided

{ "message": "Group(<group\_id>) not found" }

**○** HTTP STATUS: 401 Unauthorized

**■** In case of invalid/ expired/ no/ Other\_user authentication token

{ "message": "Authorization has been denied for this request." }

**API Testing Best Practices**

During start of Api Testing Project

1. Retrieve Architectural diagram related with Api flow/structure and layers of api 2. Information about how api changes are reflecting on UI part.

3. Database access

4. Ask for Apiary/ Swagger document (Network tab)

5. Dependency of Apis if present.

6. Need of Input files ( Zip, images or json files if required )

7. Access rights hierarchy for Application users.

8. Server logs for Api queues information if present.

9. Application’s various Environments specific users credentials/information, client ids and client secrets.

Verification points for Data

1. Data accuracy

2. Validity of data format

3. Duplicate/ missing data

4. Mandatory data/ Optional data

5. All possible inputs combinations for complete test coverage of data

Valid values & Invalid values.

( field length, field data type- numeric/char, upper/lower cases, email format etc)

Verification points for Response

1. HTTP status codes

2. Error codes in case API returns any errors

3. Improper messaging present in response

4. Incompatible error handling mechanism

5. Improper errors

6. Test for the expected results

7. Group API test cases by test category

8. Create tests to handle unforeseen problems

9. Automate API testing wherever it is possible

10. Verify whether return value is based on input condition

11. Verify for the response message if any server Jobs queue present

12. Verify whether system is authenticating the outcome when API is updating data structure 13. Verify whether the API triggers some other event or request another API 14. Verify the behavior of the API when there is no return value

15. For post request ids coming in response body/headers

16. Report to the dev team for 500 Internal Server Error code for any API response - ● Mostly could be a Bug

● In some cases if the server/Environment is down.

**Response codes**

When accessing a web server or application, every HTTP request that is received by a server is responded to with an HTTP status code.

HTTP status codes are three-digit codes, and are grouped into five different classes. The class of a status code can be quickly identified by its first digit:

l 1xx: Informational

l 2xx: Success

l 3xx: Redirection

l 4xx: Client Error

l 5xx: Server Error

E.g -

100 Continue

200 OK

201 Created

202 Accepted

204 No Content

301 Moved Permanently

307 Temporary Redirect

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

409 Conflict

415 Unsupported Media Type

500 Internal Server Error

**Response body**

1. JSON

2. HTML

3. Text

4. \_Blank

**Request header**

Components of the header section of request and response messages in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

| **Header**  **field name** | **Description Example** |
| --- | --- |
| Accept | Media type that is(/are) acceptable for the  Accept: text/html  response. |
| Authorization | Authentication credentials for HTTP  Authorization: bearer  authentication.  QWxhZGRpbjpvcGVuIHNlc2FtZ  Q== |
| Content-Type | The Media type of the body of the  Content-Type:  request (used with POST and PUT  application/x-www-form-urlenco  requests).  ded  Application/json |