[API testing using POSTMAN 2](#_Toc469660932)

[What is API testing? 2](#_Toc469660933)

[Why Back-end /Database? 2](#_Toc469660934)

[API Architecture 2](#_Toc469660935)

[Test Cases for API Testing: 3](#_Toc469660936)

[What to test for in API testing 3](#_Toc469660937)

[Best Practices of API Testing: 4](#_Toc469660938)

[Types of Bugs that API testing detects 4](#_Toc469660939)

[Tools for API testing 5](#_Toc469660940)

[Challenges of API Testing 5](#_Toc469660941)

[What should it be doing? 5](#_Toc469660942)

[How to Install POSTMAN? 7](#_Toc469660943)

[How POSTMAN works: 7](#_Toc469660944)

[How to set Environment Variable in POSTMAN? 8](#_Toc469660945)

[Add Collection: 9](#_Toc469660946)

[Ignoring SSL warnings 9](#_Toc469660947)

[Url 10](#_Toc469660948)

[Headers 10](#_Toc469660949)

[Request body 10](#_Toc469660950)

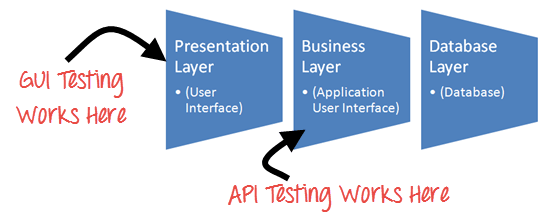
API testing using POSTMAN

What is API testing?

The API Testing is performed for the application, which has a collection of API that must be tested. API calls verify functionality and expose failure of application.

API testing is strongly connected with Back-end/Database testing; you may have brief knowledge of SQL queries. (That would be an advantage)

Why Back-end /Database?

[](http://cdn.guru99.com/images/1-2015/api_testing.png)

For API testing, you may not aware about GUI of application. So DB is only the way to crosscheck, you are doing right or wrong.

(As we crosscheck with database, if you know basic SQL queries (SELECT, UPDATE, ALTER, DELETE) that would be advantage to use POSTMAN, you can refer <http://www.w3schools.com/sql/> it)

API Architecture

API calls Collection include mainly three things:

1. HTTP headers
2. HTTP Request  (POST,GET,PUT,DELETE )
3. Status Code/ Response Code

Apart from your application collection, include many things if it required testing in your application.

1. **HTTP headers**  - HTTP headers are always depended on your application, Mainly two:

|  |  |
| --- | --- |
| **Authorization** | A token included with requests to identify the requester. This header has to be included in all requests other than the login request. |
| **Content-Type** | A standard MIME type describes the format of object data.  Content –type in most of the requests and responses will be application/json. |

1. **HTTP Request  -**There are mainly four request, which we used frequently: DATA =

|  |  |
| --- | --- |
| **POST** | Create Or Update data |
| **PUT** | Update data |
| **GET** | Retrieve data |
| **DELETE** | Delete data |

1. **Status Code/Response Code -**There are many status/response code, from them we can verify the response.

|  |  |
| --- | --- |
| 200 | OK, The request was successful. |
| 201 | Created, The request was successful and data was created. |
| 204 | No Content, The response is empty. |
| 400 | Bad Request, The request could not understood or was missing required parameters. |
| 401 | Unauthorized, Authentication failed or user does not have permissions for the requested operation. |
| 403 | Forbidden, Access denied. |
| 404 | Not Found, Data not found. |
| 405 | Method Not Allowed, Requested method is not supported for the specified resource. |
| 500 | Internal Server Error. |
| 503 | Service Unavailable, The service is temporary unavailable. |

<https://www.getpostman.com/docs/responses>

Test Cases for API Testing:

Test cases of API testing are based on

* **Return value based on input condition:** it is relatively easy to test, as input can be defined and results can be authenticated
* **Does not return anything:**When there is no return value, behavior of API on the system to be checked
* **Trigger some other API/event/interrupt:**If output of an API triggers some event or interrupt, then those events and interrupt listeners should be tracked
* **Update data structure:**Updating data structure will have some outcome or effect on the system, and that should be authenticated
* **Modify certain resources:**If API call modifies some resources then it should be validated by accessing respective resources

What to test for in API testing

API testing should cover at least following testing methods apart from usual SDLC process

* **Discovery testing:** The test group should manually execute the set of calls documented in the API like verifying that a specific resource exposed by the API can be listed, created and deleted as appropriate
* **Usability testing:**This testing verifies whether the API is functional and user-friendly. And does API integrates well with another platform as well
* **Security testing:**This testing includes what type of authentication is required and whether sensitive data is encrypted over HTTP or both
* **Automated testing:**API testing should culminate in the creation of a set of scripts or a tool that can be used to execute the API regularly
* **Documentation:**The test team has to make sure that the documentation is adequate and provides enough information to interact with the API. Documentation should be a part of the final deliverable

Best Practices of API Testing:

* Test cases should be grouped by test category
* On top each test, you should include the declarations of the APIs being called.
* Parameters selection should be explicitly mentioned in the test case itself
* Prioritize API function calls so that it will be easy for testers to test
* Each test case should be as self-contained and independent from dependencies as possible
* Avoid "test chaining" in your development
* Special care must be taken while handling one-time call functions like - Delete, Close Window, etc...
* Call sequencing should be performed and well planned
* To ensure complete test coverage, create test cases for all possible input combinations of the API.

Types of Bugs that API testing detects

* Fails to handle error conditions gracefully
* Unused flags
* Missing or duplicate functionality
* Reliability Issues. Difficulty in connecting and getting response from API.
* Security Issues
* Multi-threading issues
* Performance Issues. API response time is very high.
* Improper errors/warning to caller
* Incorrect handling of valid argument values
* Response Data is not structured correctly (JSON or XML)

Tools for API testing

Since API and unit testing both target source code, similar tools can be used for testing both.

* SOAPUI
* Runscope
* Postman with jetpacks
* Postman with newman
* Curl
* Cfix
* Check
* CTESK
* dotTEST
* Eclipse SDK tool- Automated API testing

Challenges of API Testing

Challenges of API testing includes:

* Main challenges in API testing is **Parameter Combination, Parameter Selection, and Call Sequencing**
* There is no GUI available **to test the application which makes** difficult to give input values
* Validating and Verifying the output in different system is little difficult for testers
* Parameters selection and categorization required to be known to the testers
* Exception handling function **needs to be tested**
* Coding knowledge is necessary for testers

What should it be doing?

* Documentation: API

Like any other kind of software, APIs need to be documented, and the documentation needs to be accurate, complete, and usable. In the case of an API, the target audience for the documentation consists of developers, and the documentation must allow them to make full use of the API; it should be tested as part of the API. Does the documentation allow programmers with access to no other sources of information about the API to make calls to it? Does it cover all features, all inputs and outputs? Does it include the range, type, and format of allowable input data, and the limits of the output data? Does it describe the function of the API, and of each ALI call? Does it list all error codes sent by the API, with their meanings?

* Documentation: The tests

The tests themselves also need to be documented. This includes the test results, of course, **which need to clearly and fully describe the conditions, inputs, expected and observed outputs, and any unexpected effect**. But it also includes test procedures and testing software. You should assume that at some point in the future, someone else my need to perform the same or similar tests, or reuse the testing software. On an immediate level, the development team or another test team may need to verify the tests or examine the test procedures for additional information.

We use POSTMAN as API testing tools.

How to Install POSTMAN?

1. Open a Google chrome
2. Click on : [**https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en**](https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en)
3. Launch app

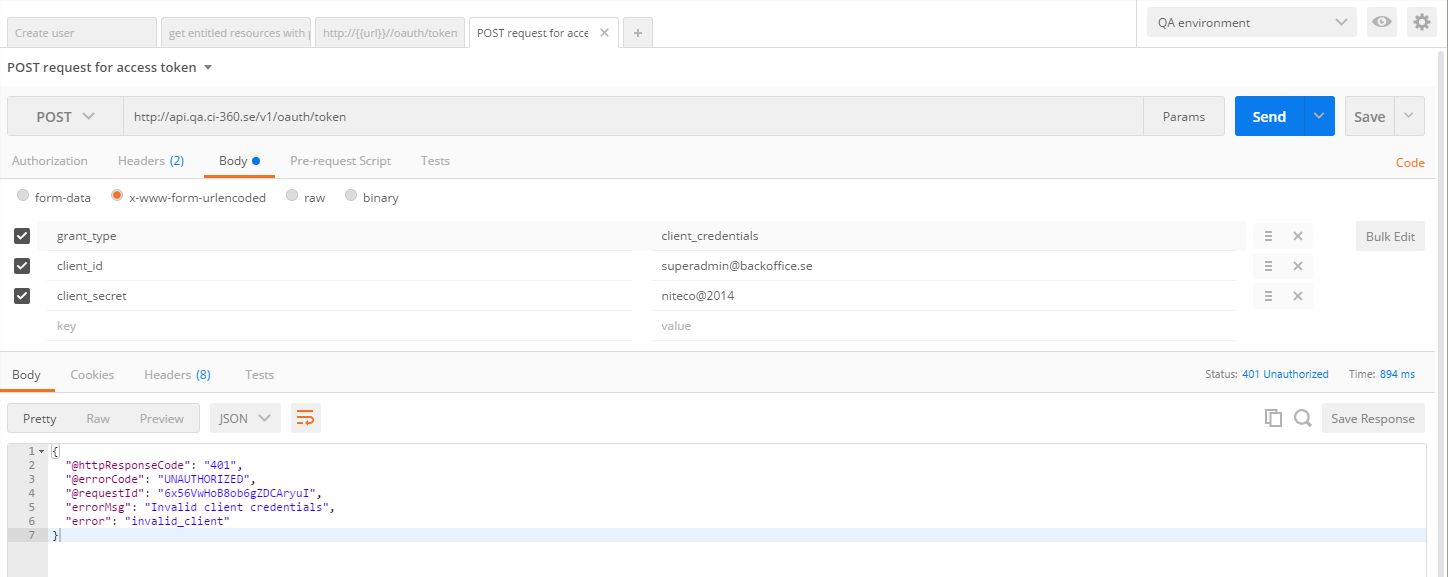
POSTMAN is very easy to use, but API testing is very tricky when your application is complex. Application provide you collection of API calls, you have to follow that collection of API calls for API testing of your application.

How POSTMAN works:

Select API call (GET/PUT/POST/DELETE)

Set Authorization, Header, Body information accordingly your API call:

Then, you can click on **send** to perform your API call.



How to set Environment Variable in POSTMAN?

From Top-Right corner you can set environments variable.

Example: If want to check on local env., Dev env. OR QA env. Even you can set accordingly your various projects as well.

It is very easy to set environment variable.

**Steps to set environment variable.**

1. Click on Manage Environment.
2. Click on ADD
3. Write down the Name of Environment.
4. Fill key & value, you can pass key = variable and value is your host IP address.

Example:  Suppose your URL is something like:

http://api.qa.ci-360.se/v1/oauth/token

Here, I am considering it is a QA environment.

Name Of Environment = QA

Key = urlQA and value = api.qa.ci-360.se/v1/

Whenever, I want to use this environment, I just select QA from top corner. Then, here we have to use …

Https://{{urlQA}}/oauth/token For any API calls

Add Collection:

You can add Each API call in collection and create a collection.

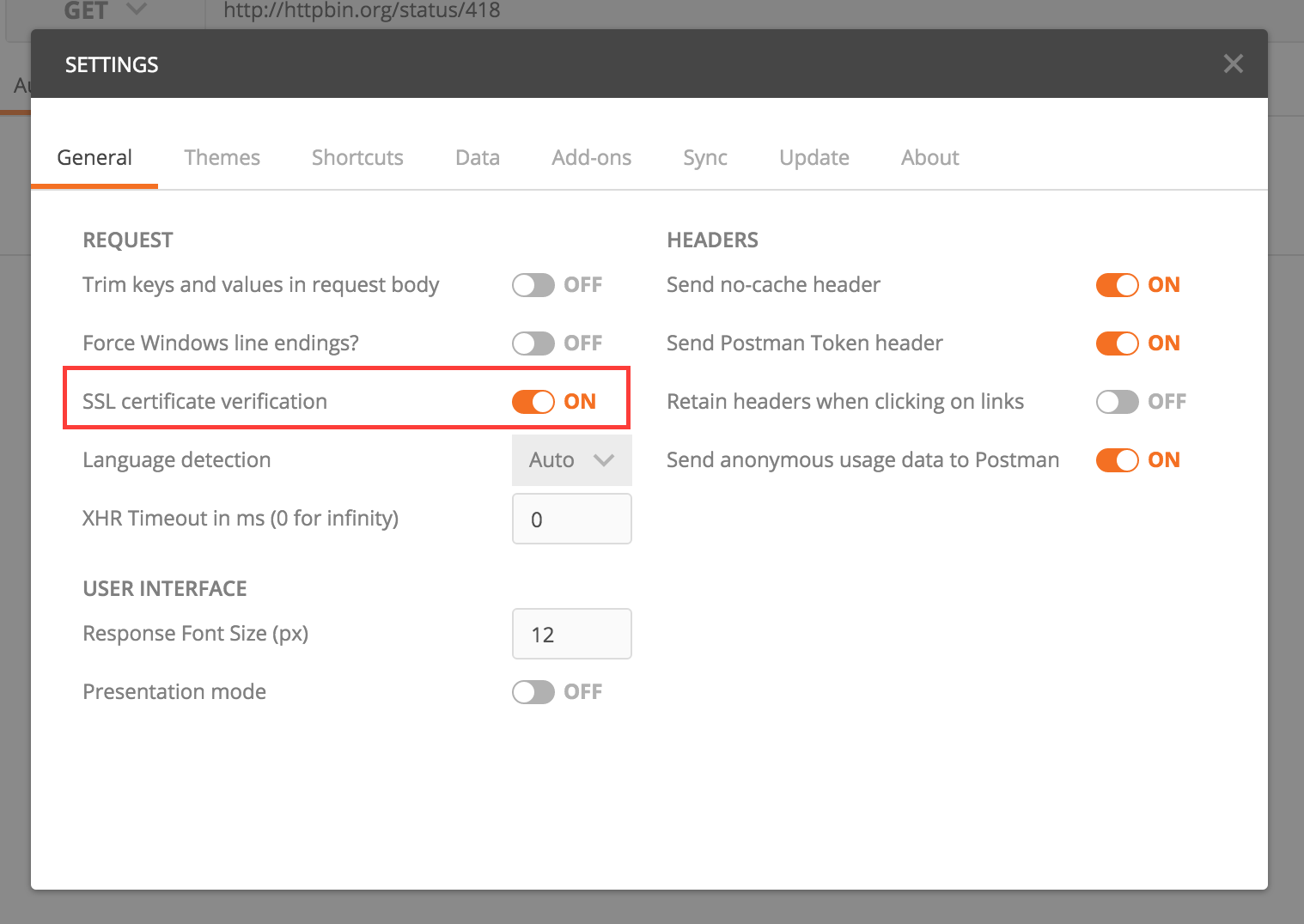
That will be reusable for application.

You can import collection of others.

You can export your collection; others can use it on their machine as well.

Ignoring SSL warnings

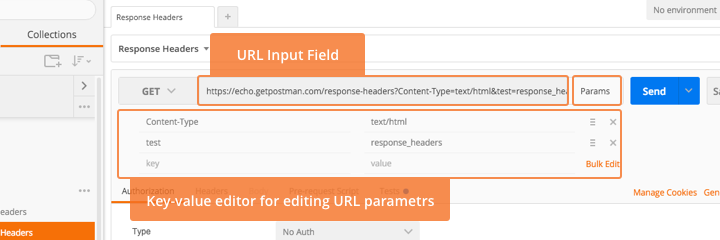
1. Head to the Settings window, and go to the General tab.
2. Change the SSL certificate verification setting to Off.
3. Restart the app.



Url

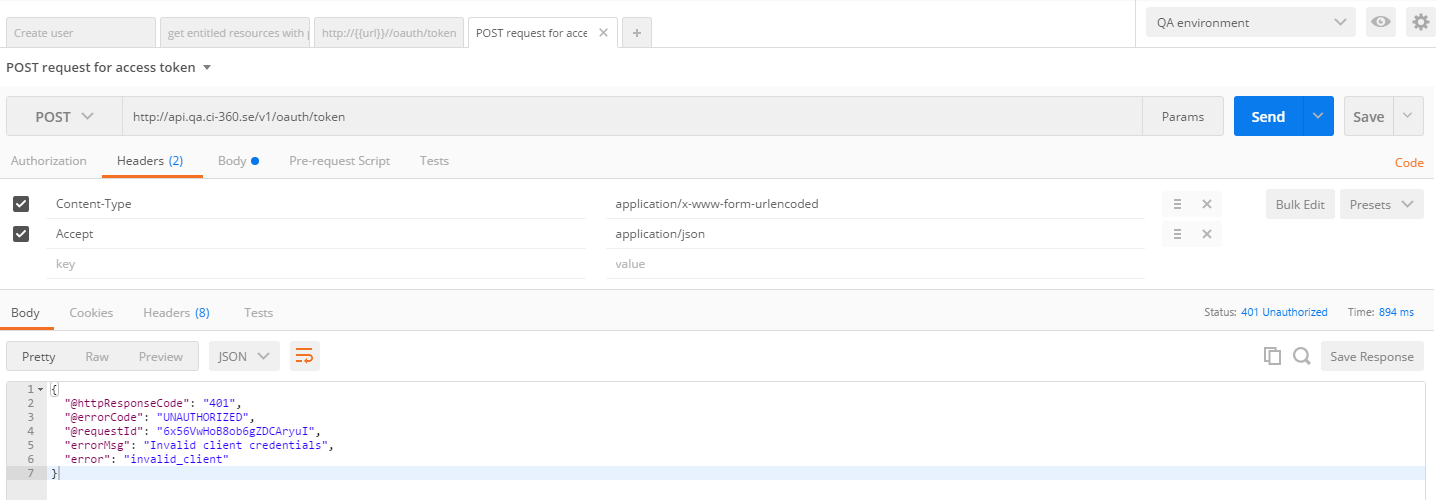
The URL is the first thing that you would be setting for a request. The URL input field stores URLs used previously and will show an auto complete dropdown as you begin entering your URL. Clicking on the URL params button will open up the [key-value editor](https://www.getpostman.com/docs/keyvalue_editor) for entering URL parameters.

Parameters you enter in the URL bar or in the key/value editor will **not** automatically be URL-encoded. Right click on a piece of selected text, and click **EncodeURIComponent** to manually encode the parameter value.



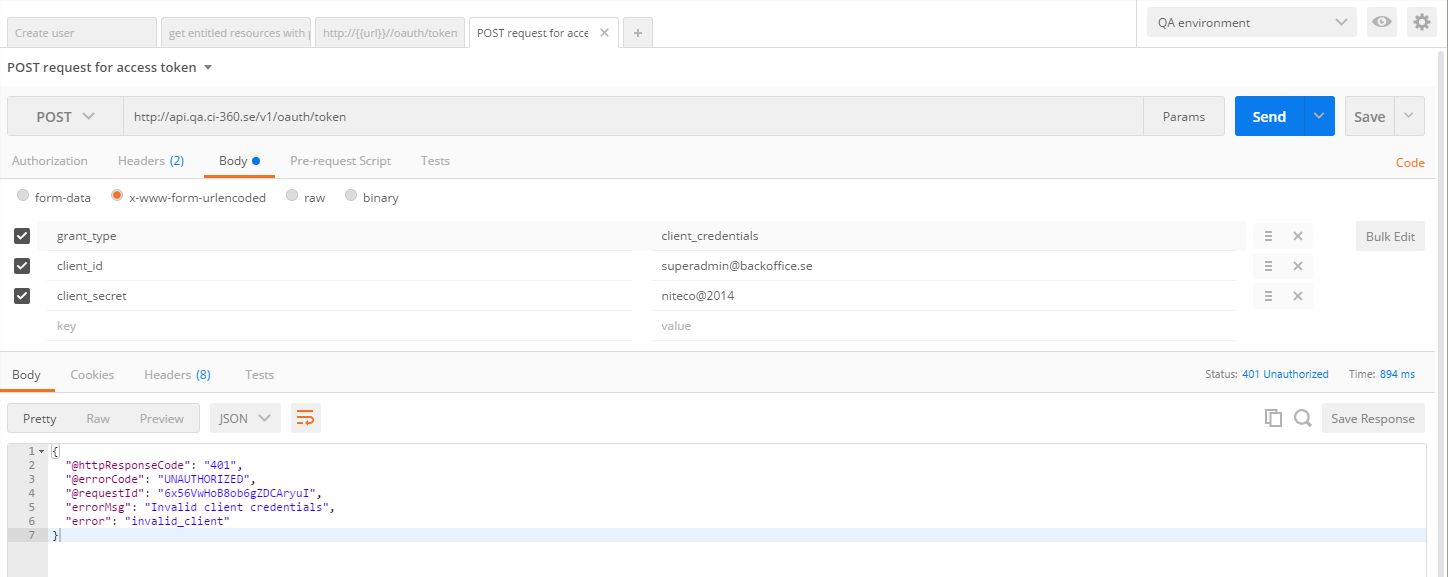
Headers

Clicking on the headers toggle will show the headers [key-value editor](https://www.getpostman.com/docs/keyvalue_editor). You can set any string as the header name. Common headers part of the HTTP spec are available in an auto-complete drop down when you begin typing the header name. Values for the “Content-Type” header are also available in an auto-complete drop down.



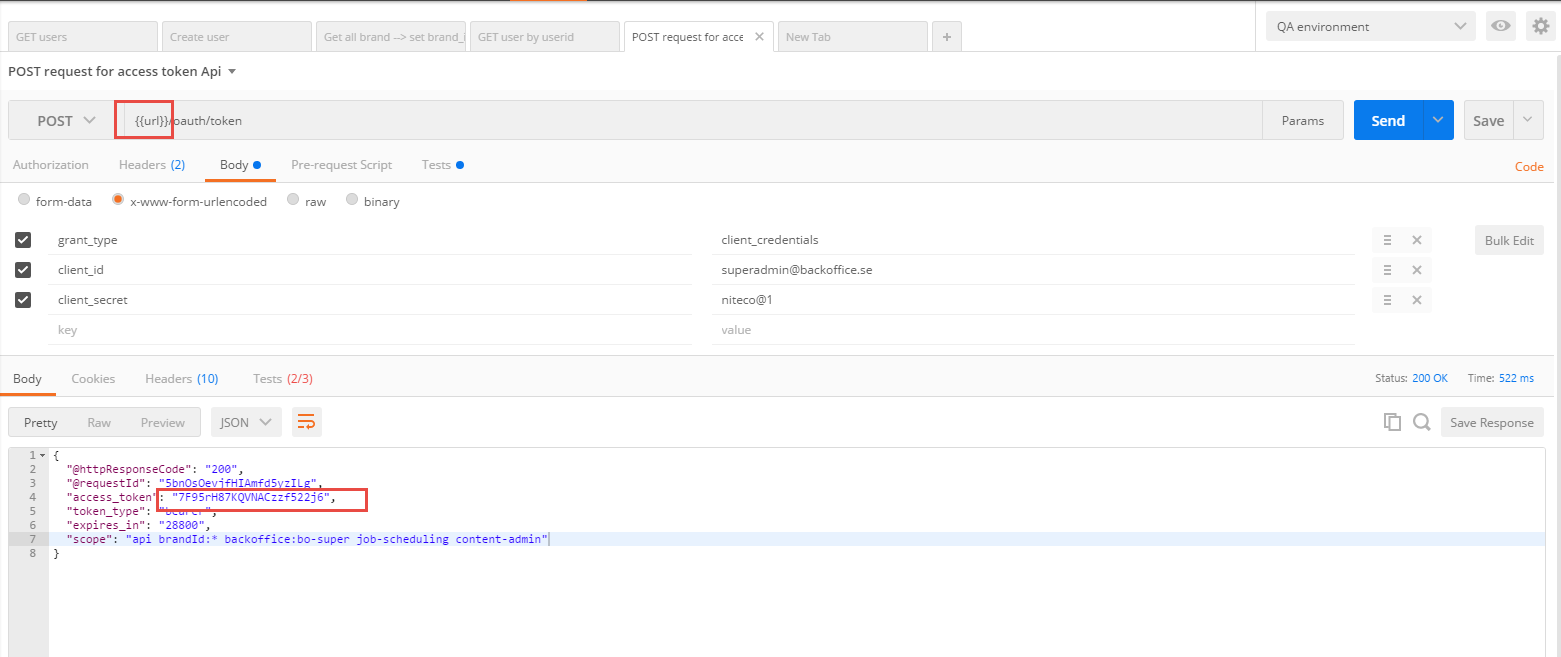
Request body

While constructing requests, you would be dealing with the request body editor a lot. Postman lets you send almost any kind of HTTP request (If you can’t send something, let us know!). The body editor is divided into 4 areas and has different controls depending on the body type.



How to dynamic parameter and improve api test collection more effective.

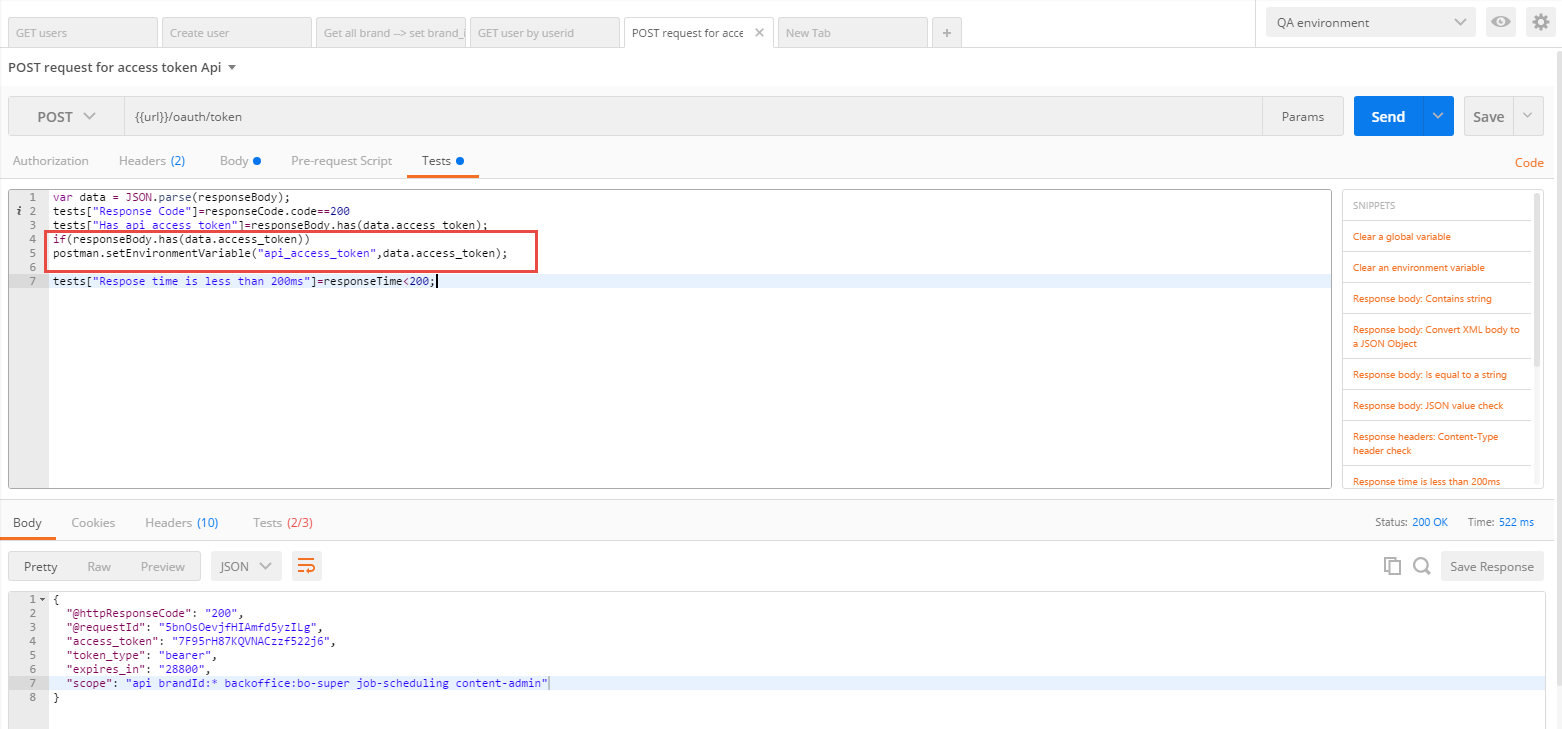
1. Dynamic parameter
   * Dynamic parameter for url of QA/ Stage Environment.

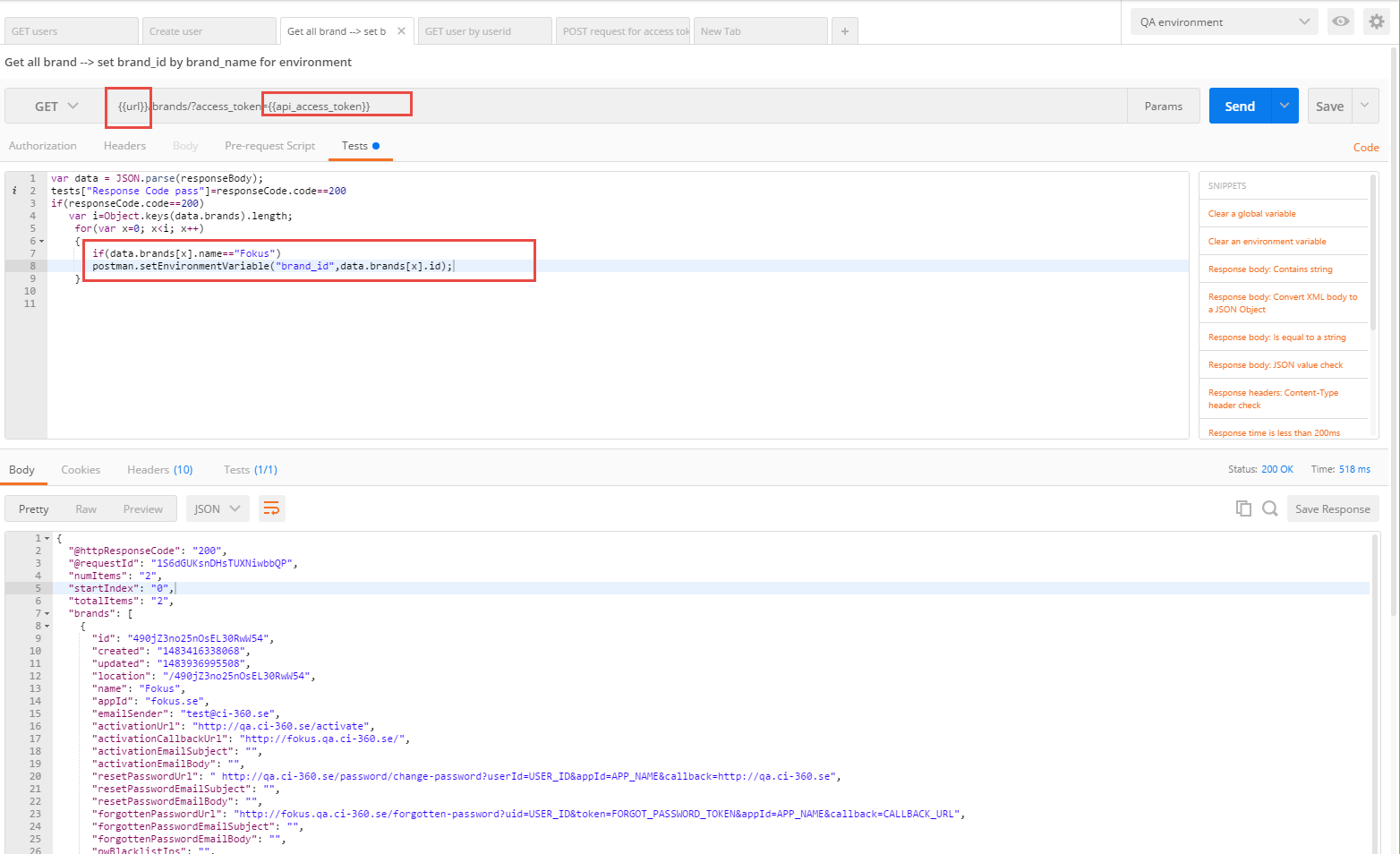


* + What are the parameters can be using multiple time.
  + What are the parameters can be get by response body of period query.

1. Write test script to set/get Global variable; Environment variable;

<https://www.getpostman.com/docs/sandbox>





Go Manager Environment to verify parameter variable after call bellow api.

* + - The parameter and value will be added in list Key

