Test Run Webhook - API Documentation

Table of Content

1. Revision History

Version	Dates	Comment
1.0	20 jul 2022	API to update test result
2.0	11 oct 2022	Add clone, get status API
3.0	03 nov 2022	Update api urls
3.1	11 nov 2022	Add get test script API Security updates

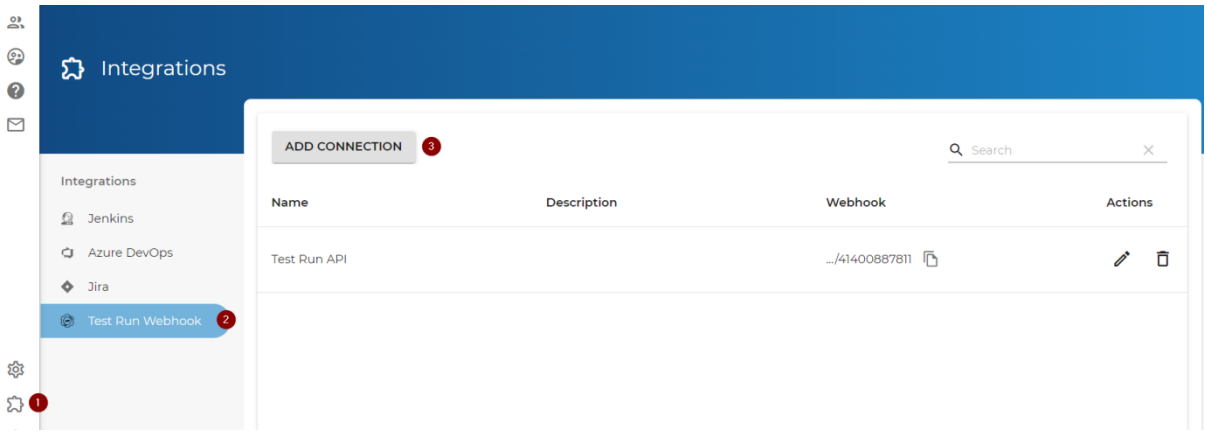
2. Objective

Test Run webhook allows you to configure your account to patch test cases under given test run using implemented endpoints from you application.

If you need support contact us by email: support@qacoverage.com

3. Configure QACoverage

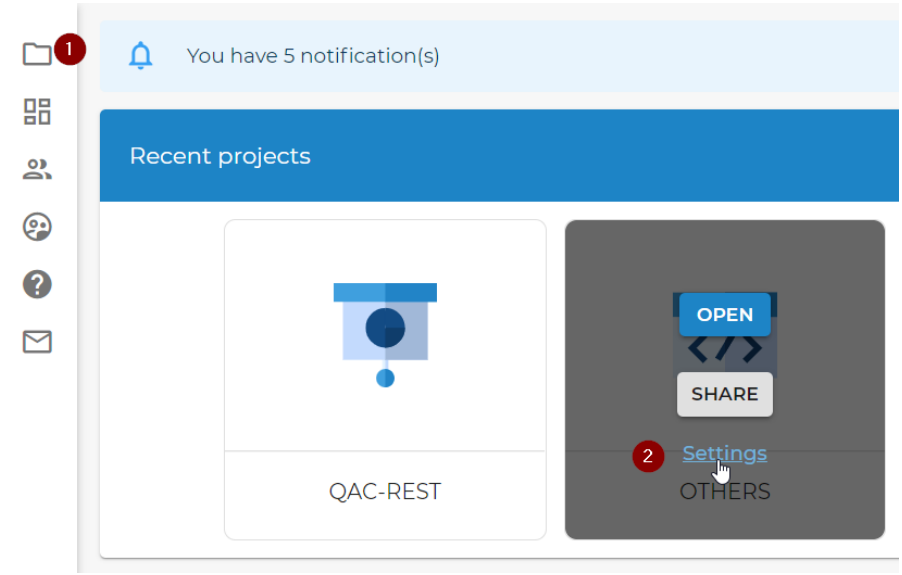
To be able to inject Test cases result on QACoverage, the account should contain at least one connection under menu Integrations > Test Run Webhook

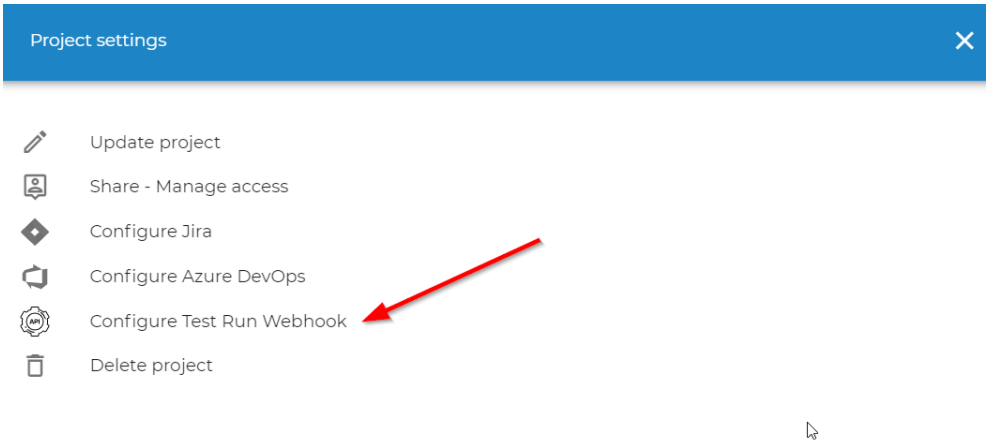


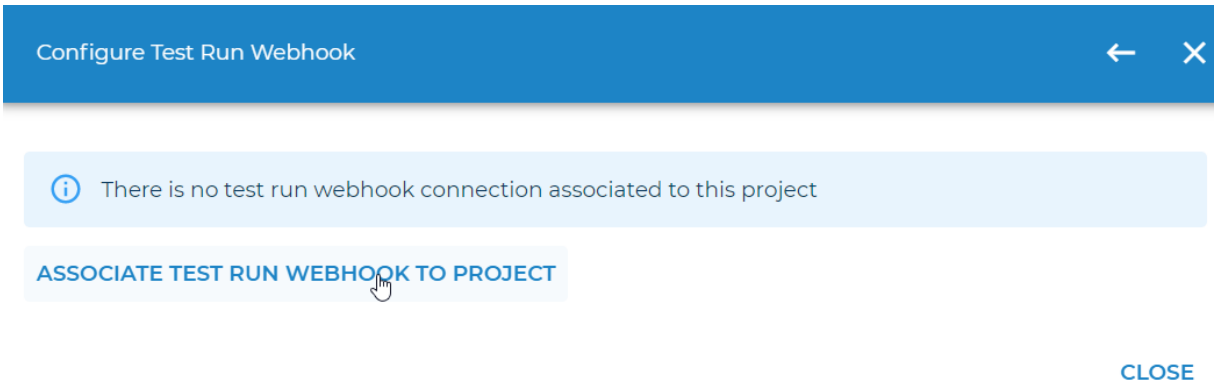
1- Add connection :
- Connection name
- Test result Statuses mapping: between Remote test execution tool statuses and QACoverage statuses. Statuses to be mapped.

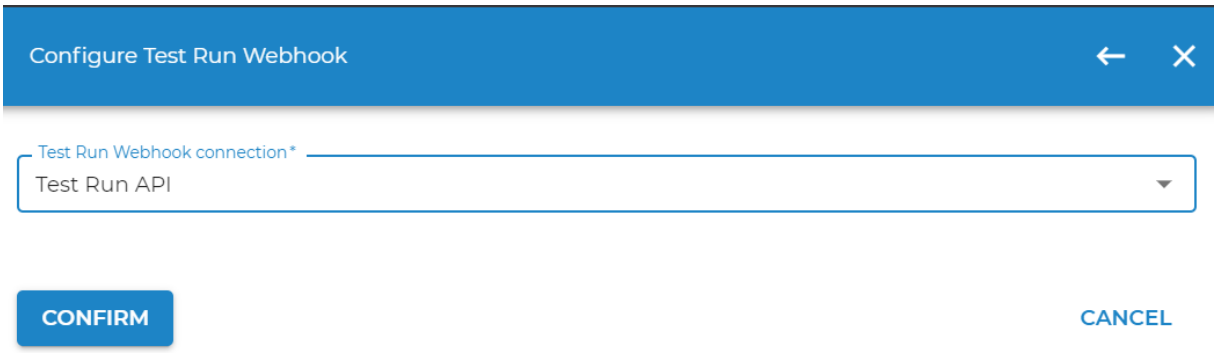
⇨ For each connection, QACoverage generates a unique webhook url to be used for API call

2- Map the used project to the created connection. Go to Projects menu > select in target project “Settings” > Configure Test Run Webhook > Associate test run webhook to project > Select the connection and confirm


1
You have 5 notification(s)
Recent projects
QAC-REST
OPEN
SHARE
2
Settings
OTHERS


Project settings
Update project
Share - Manage access
Configure Jira
Configure Azure DevOps
Configure Test Run Webhook
Delete project

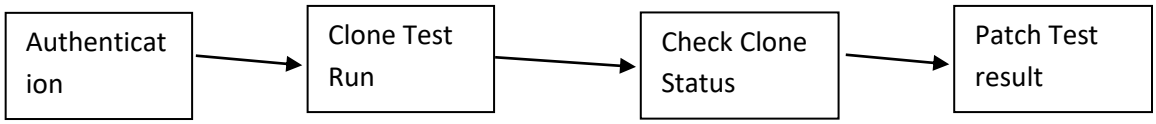

Configure Test Run Webhook
There is no test run webhook connection associated to this project
ASSOCIATE TEST RUN WEBHOOK TO PROJECT
CLOSE



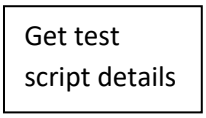
4. Update Test case result in Test Run

After configuration step, you should follow process:

- Authentication
- Clone test run and get TEST RUN ID using dedicated Endpoint
- Check Test Run creation status until the clone process is done successfully (depends on number of associated test cases)
- Update test cases and steps results under specific Test Run using Patch API call



- Endpoint useful for BDD scenarios to get test description from QACoverage and execute them by test automation framework:

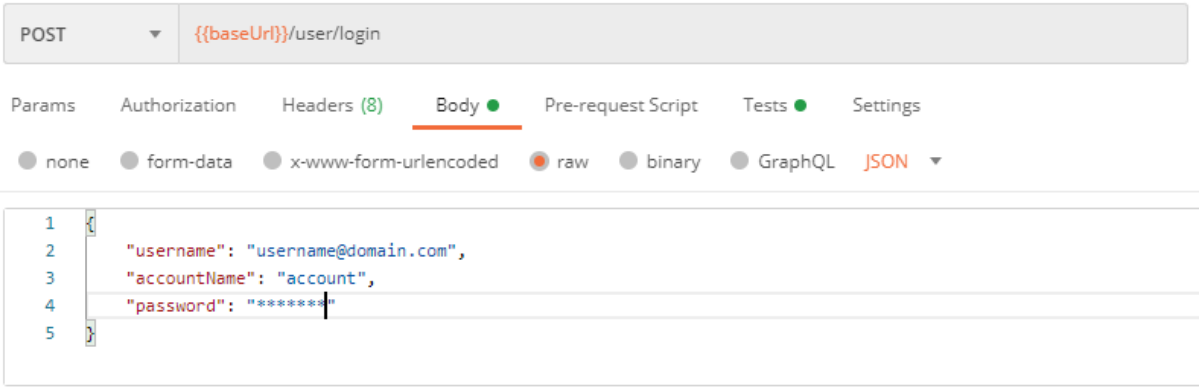


4.1. API Authentication

Authentication protocols, uses OAuth2: a request for logging in QACoverage should be executed to get the access token. Then the next APIs will use this token in their header.

Request Body

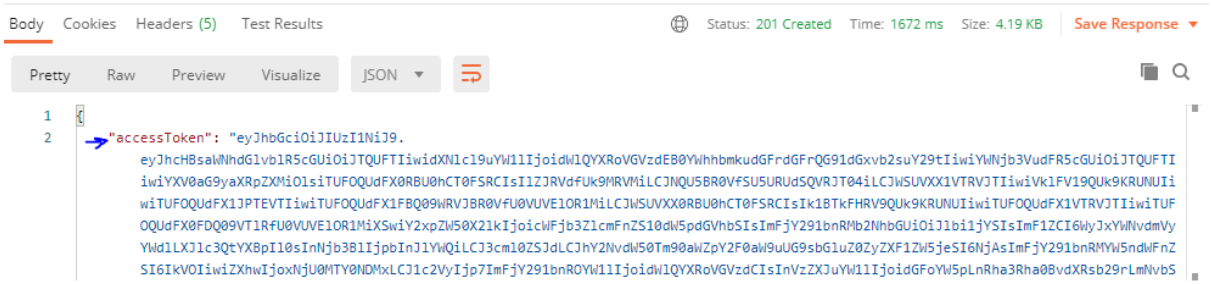
- Endpoint: https://app.QACoverage.com/API/user/login
- The request body is a JSON object, it contains:
  - "**accountName**": the QACoverage account
  - "**username**" and "**password**": user's credentials to access to QACoverage account



Response Body

As response the API returns:

- accessToken: this token will be used in the header of the next APIs

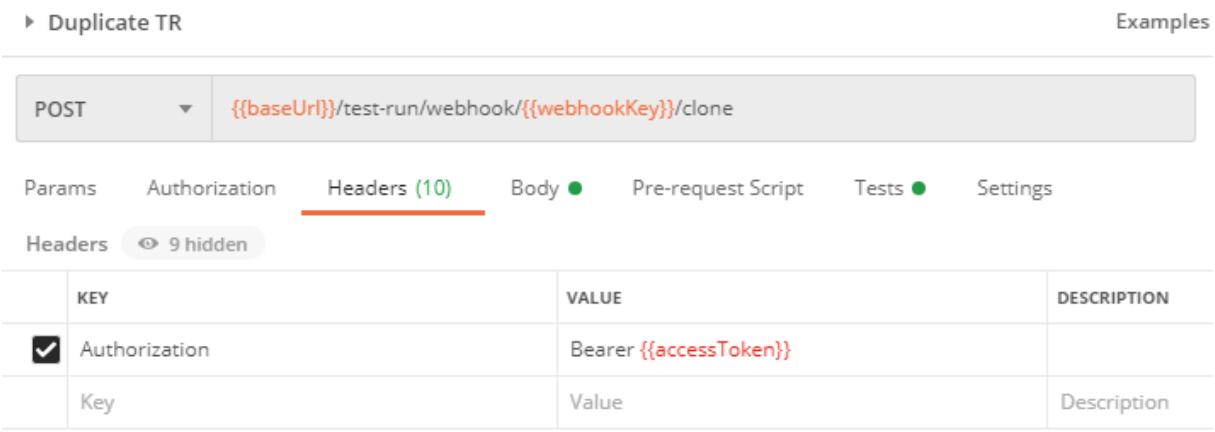


4.2. Clone test run

This API allows you to clone a TEST RUN under your account using OAuth2 authentication.

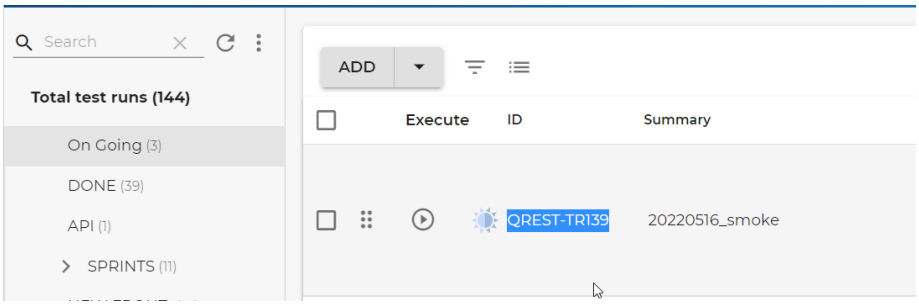
Request Body

- Endpoint: https://app.QACoverage.com/API/test-run/webhook/${webhookKey}/clone
- The webhookKey in the path is the webhook key token from the QACoverage configuration done previously.
- The request body is a JSON object
- This API uses OAuth2 authentication



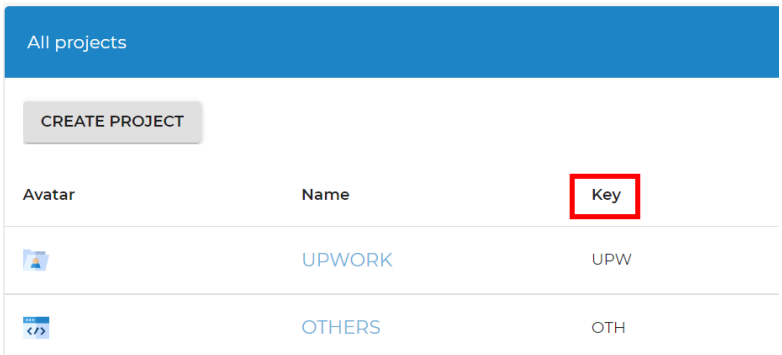
The request body contains:

- "**id**" : Test Run to be cloned
  This field can take two possible values:
  - Test run ID source: Go to QACoverage and copy/paste test run ID from Test Run menu



  - "last" : it will clone the last TEST RUN created under the specified project or under the account if project id is null (based on created At value)

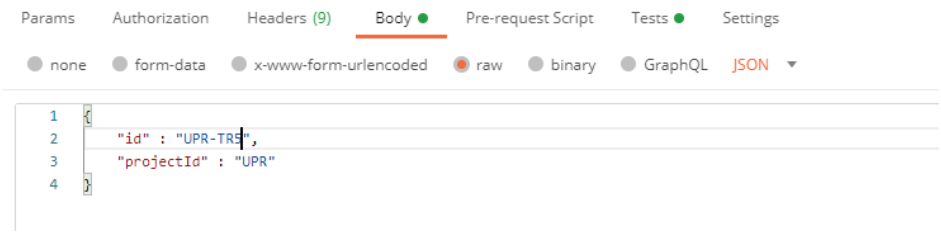
- "**projectId**": this value can be null
  - If value is specified: it's the Project Key from project menu. Clone the last created TEST RUN under the given project



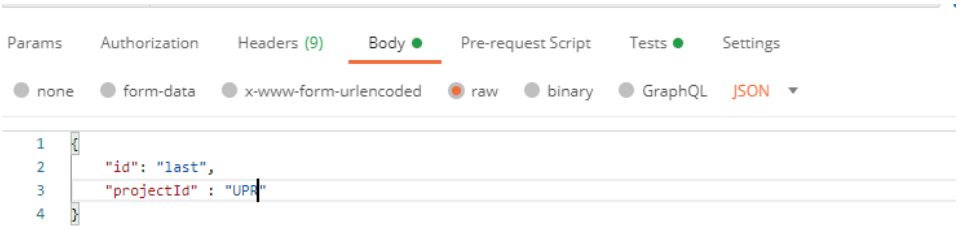
  - If null or empty: clone the last created TEST RUN under the whole account (based on created at value)

Examples:

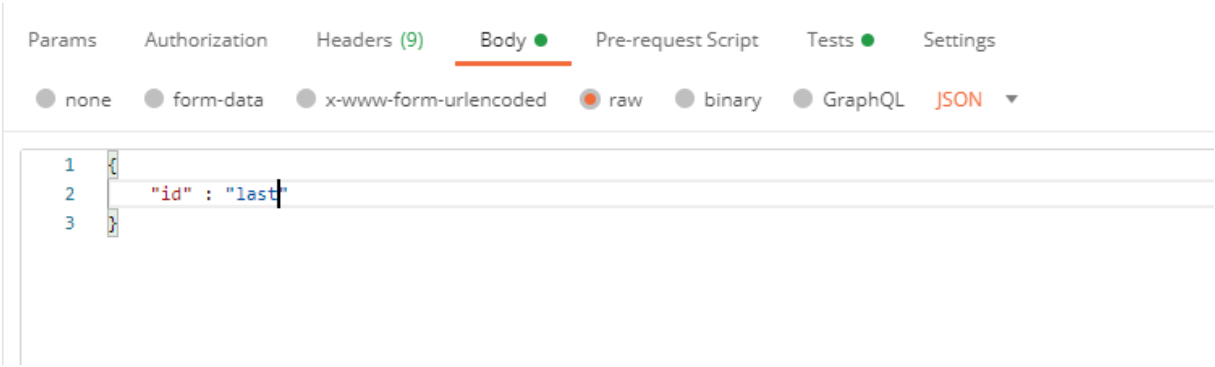
For instance, below API call will clone the TEST RUN defined in id field, under the project with id value.



For instance, below API call will clone the last created TEST RUN under the project with id “UPR”

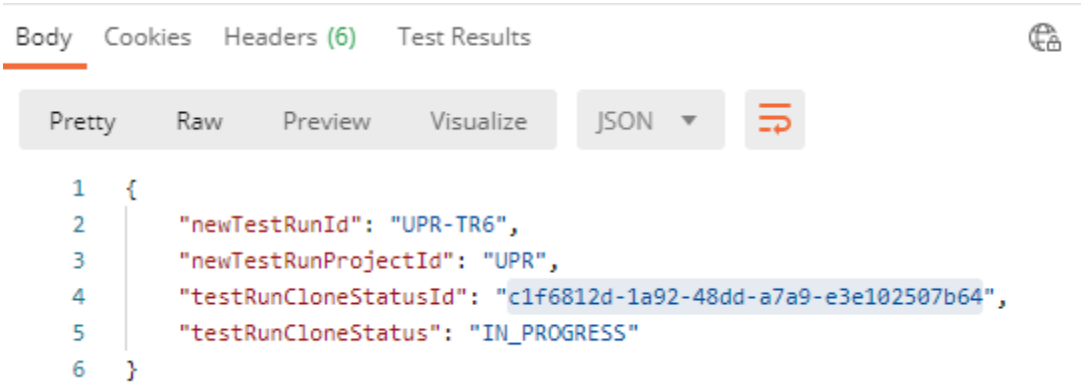


Below API call, will clone the last created TEST RUN under the connected user account



Response Body

As response the API returns:



- newTestRunId: the created TEST RUN Id
- newTestRunProjectId: the project Id where the test run was created
- testRunCloneStatusId: the clone status id, to be used later to check the progression of the process
- testRunCloneStatus: it indicates the status of the clone process.

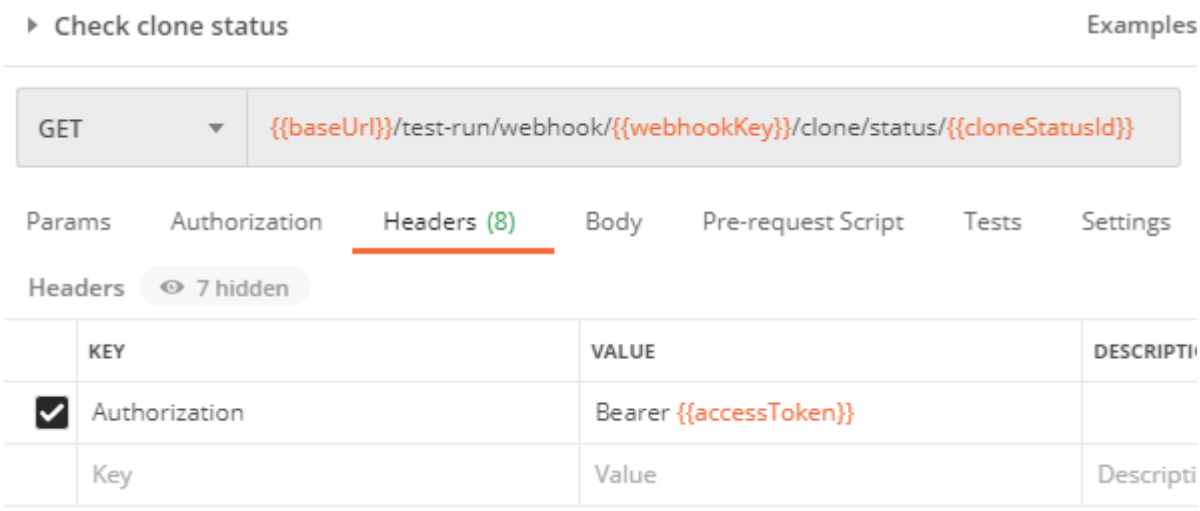
Error cases

Error Code	Error Message	Description
404 Not found	PROJECT_NOT_FOUND	The given project id is not found
404 Not found	TEST_RUN_NOT_FOUND	The given test run id is not found
400 Bad Request	TEST_RUN_PROJECT_MISMATCH	The given test run does not belong to the given project id

This API will only launch the clone process. So you need to follow the progress before starting using the cloned TR.

4.3. Check “Clone Test Run” status

Depends on number of selected test cases, creation of new test run may take several seconds. To follow the progress of the launched clone process, you can use this API:



- Endpoint: https://app.QACoverage.com/API/test-run/webhook/${webhookKey}/clone/status/${ cloneStatusId}
- The webhookKey in the path is the webhook key token from the QACoverage configuration done previously.
- The last id in the path is the clone status id (from the previous step response : “testRunCloneStatusId”)
- This API uses OAuth2 authentication

Response Body

Possible statuses

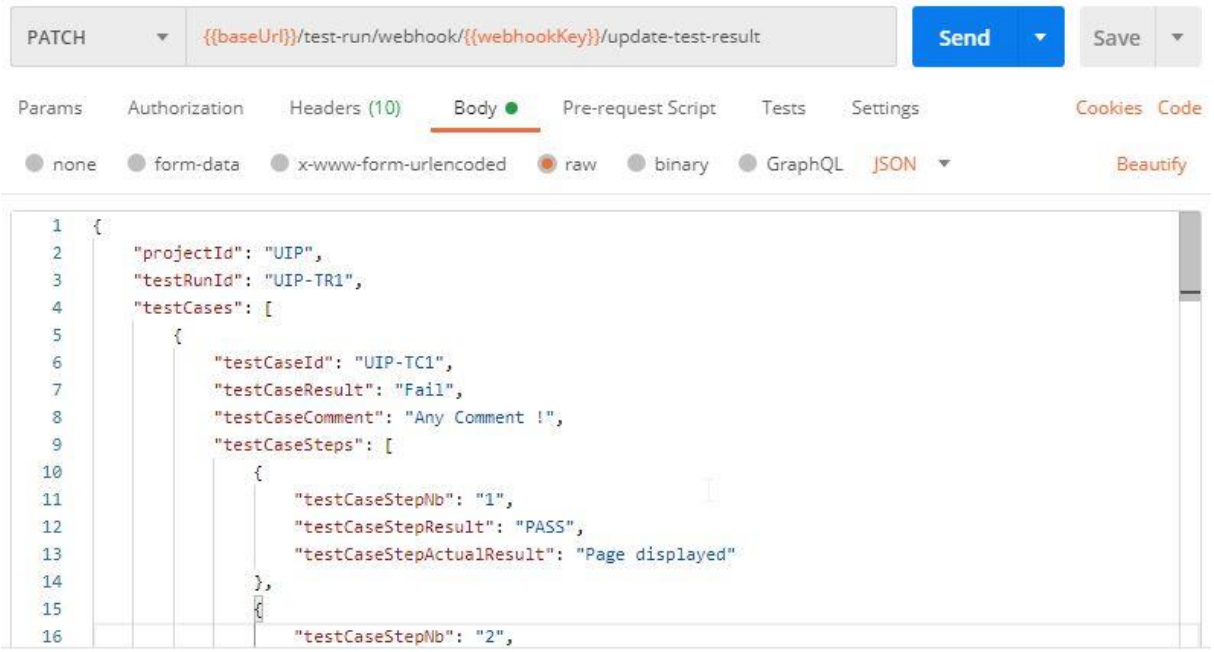
Status	Description
IN_PROGRESS	The clone process still progressing
TERMINATED	The clone process is terminated successfully
FAIL	The clone process fails

Once the clone is TERMINATED, the patch API can be executed.

4.4. Patch / Update Test case results

The API will patch the given TEST RUN test cases results and the test cases steps results also.

Example



- Endpoint: https://app.QACoverage.com/API/test-run/webhook/${webhookKey}/update-test-result
- The id in the path is the webhook key token from the QACoverage configuration done previously.
- This API uses OAuth2 authentication

Request Body

- “projectId” : the project id where the TEST RUN is created
- “testRunId” : the cloned TEST RUN id (already returned by the first API)
- “testCases” : is a list :
    - “testCaseId” : test case id to patch
    - “testCaseResult”: test case result (PASS, FAIL, PENDING)
    - “testCaseComment”: A text comment to be added as actual result for TC
    - “testCaseSteps”: is a list :
        - “testCaseStepNb” : the step order under the TC
        - “testCaseStepResult”: the step result (PASS, FAIL, PENDING)
        - “testCaseStepActualResult” : A text comment as actual result for the step

Example:

Example of Request Body	{ "projectId": "UIP", "testRunId": "UIP-TR1", "testCases": [

```
        {
            "testCaseId": "UIP-TC1",
            "testCaseResult": "PASS",

            "testCaseComment": "Any Comment !",
            "testCaseSteps": [
                {
                    "testCaseStepNb": "1",
                    "testCaseStepResult": "PASS",
                    "testCaseStepActualResult": "Page displayed"
                },
                {
                    "testCaseStepNb": "2",
                    "testCaseStepResult": "PASS",
                    "testCaseStepActualResult": ""
            ]
        },
        {
            "testCaseId": "UIP-TC2",
            "testCaseResult": "PASS",
            "testCaseComment": "Any Comment !",
            "testCaseSteps": [
                {
                    "testCaseStepNb": "1",
                    "testCaseStepResult": "PASS",
                    "testCaseStepActualResult": "Page displayed"
                }
            ]
        }
    ]
}
```


Response Body

Response Status / Body


```
Status 200
{
    "projectId": "UIP",
    "testRunId": "UIP-TR1",
    "errorMessage": null,
    "testCases": [
        {
            "testCaseId": "UIP-TC1",
            "updateStatus": "SUCCESS",
            "errorMessage": null,
            "testCaseSteps": [
                {
                    "testCaseStepNb": "6",
                    "updateStatus": "FAIL",
                    "errorMessage": "TC_STEP_NOT_FOUND"

            ]
```

```
        } ,
        {

            "testCaseId": "UIP-TC16",
            "updateStatus": "FAIL",
            "errorMessage": "[UIP-TC16]_NOT_FOUND",
            "testCaseSteps": null
        }
]
}
```


Status 404


```
{
    "projectId": "UIP",
    "testRunId": "UIP-TR12",
    "errorMessage": "TEST_RUN_NOT_FOUND",
    "testCases": null
}
```


OR


```
{
    "projectId": "UIPs",
    "testRunId": null,
    "errorMessage": "PROJECT_NOT_FOUND",
    "testCases": null
}
```


Status 400


```
{
    "projectId": "",
    "testRunId": null,
    "errorMessage": "INVALID_PROJECT_ID",
    "testCases": null
}
```


OR


```
{
    "projectId": "UIP",
    "testRunId": null,
    "errorMessage": "PROJECT_UIPATH_CONNECTION_MISCONFIGURED",
    "testCases": null
}
```


OR


```
{
    "projectId": "UIP",
    "testRunId": null,
    "errorMessage": "PROJECT_NOT_CONFIGURED_TO_UIPATH",
    "testCases": null
}
```


OR


```
{
    "projectId": "UIP",
    "testRunId": null,
```

	<pre>"errorMessage": "WRONG_WEBHOOK_KEY",
	"testCases": null
	}</pre>

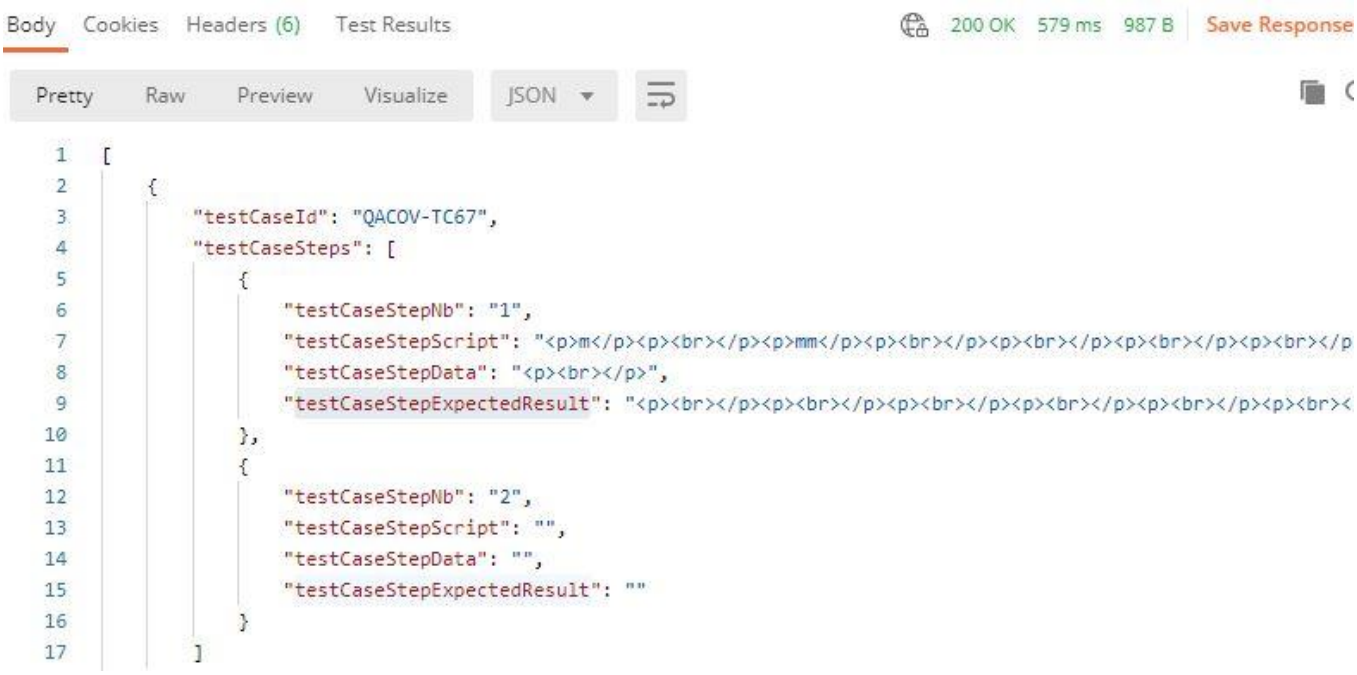
5. Get Test cases details under cloned test run

This API allows you to get the list of test cases associated to test run using OAuth2 authentication executed after login endpoint:

- Endpoint: https://app.QACoverage.com/API/test-run/webhook/${webhookKey}/${testRunId}/test-cases
- The testRunId in the path is the cloned test run id (from the 4.2 step response : "newTestRunId")
- The webhookKey in the path is the webhook key token from the QACoverage configuration done previously.
- This API uses OAuth2 authentication

Response Body

- "testCaseId" : test case id to patch
- "testCaseSteps" : is a list :
  - "testCaseStepNb" : the test case step number to patch
  - "testCaseStepScript": content of "Test Instructions" field
  - "testCaseStepData": content of "Test Data" field
  - "testCaseStepExpectedResult": content of "Expected Results" field



6. Postman Collection

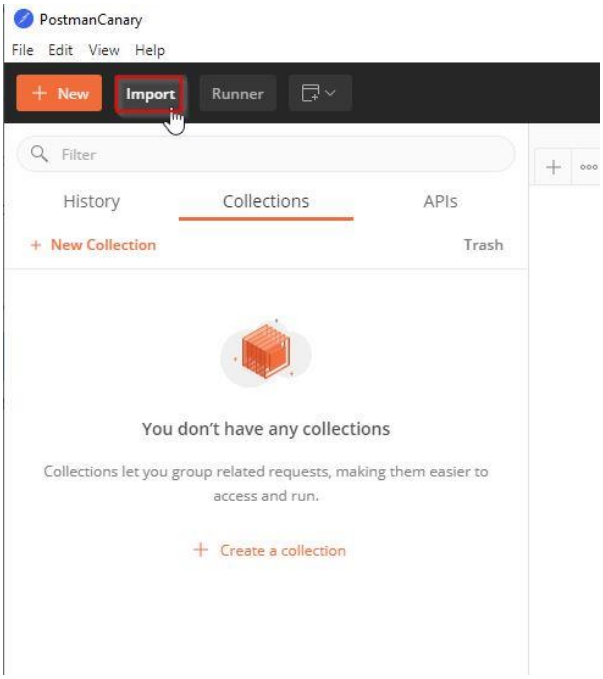
We used Postman to add the endpoints and to test them. As pre requisite, you have to install Postman tool. For more details you can check this tool there https://www.postman.com/

To keep our workspace organized we grouped these requests into a collection and you can import this collection to execute the endpoints from here (download file content)

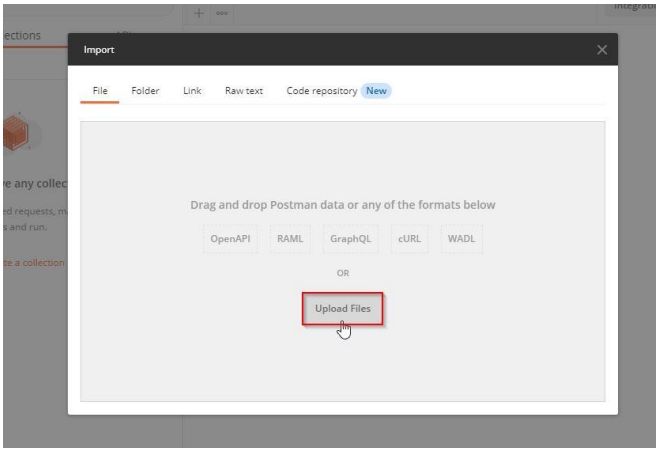
6.1. Import Collection

To import your collection into Postman:

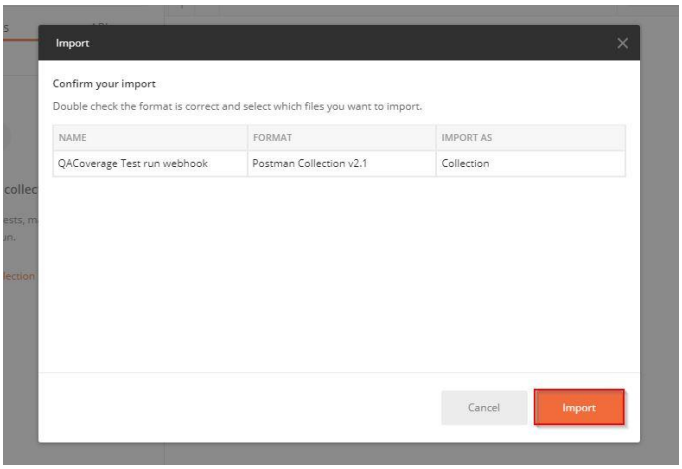
- select Import in upper left:



- Select the file **QACoverage Test run webhook.postman_collection.json** to import.

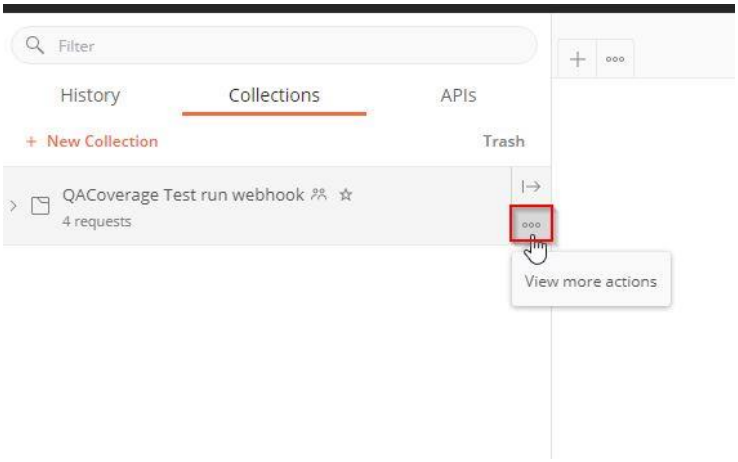


- Select Import to bring your collection into Postman.

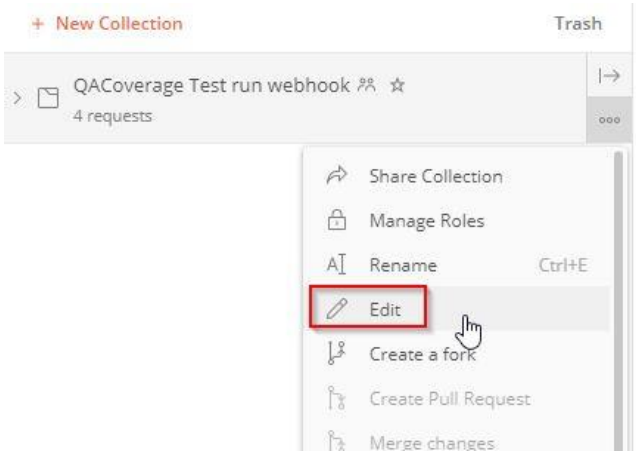


6.2. Define value of variables

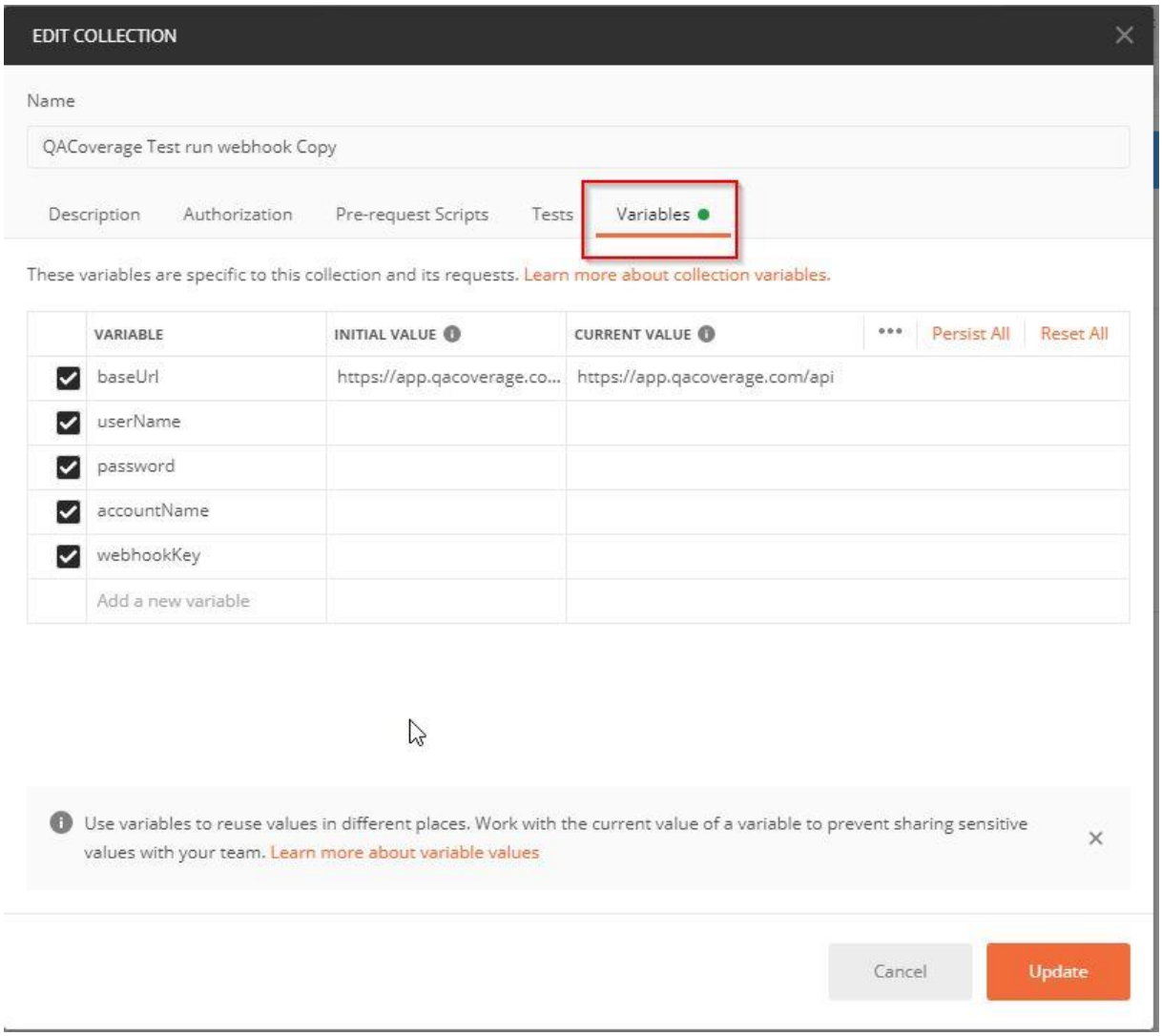
- Click on the option icon



- Click on “Edit”

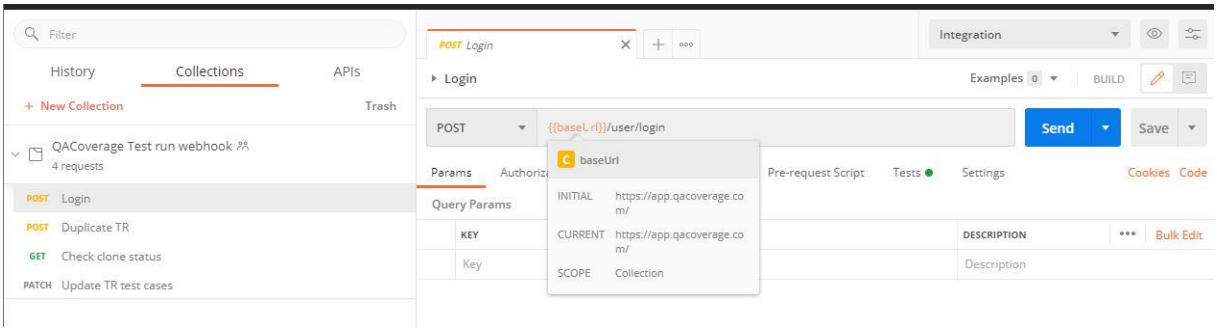


- Click on “variables”

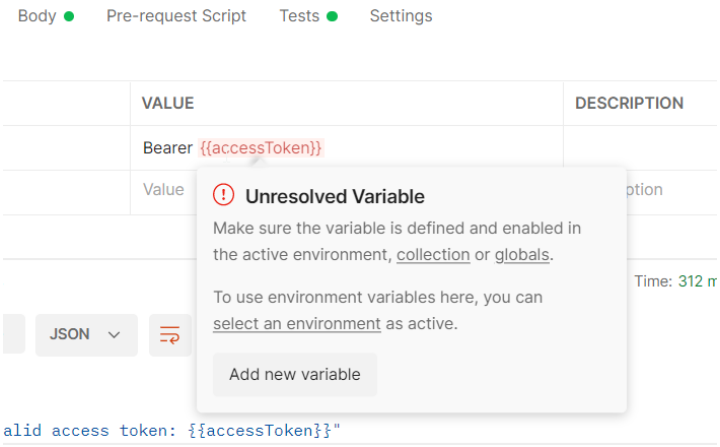


- Add your username, password, accountName and webhook key in current value column
- Click on Update button

When you hover over a variable, Postman shows its current value.



If a variable is unresolved, Postman highlights it in red.

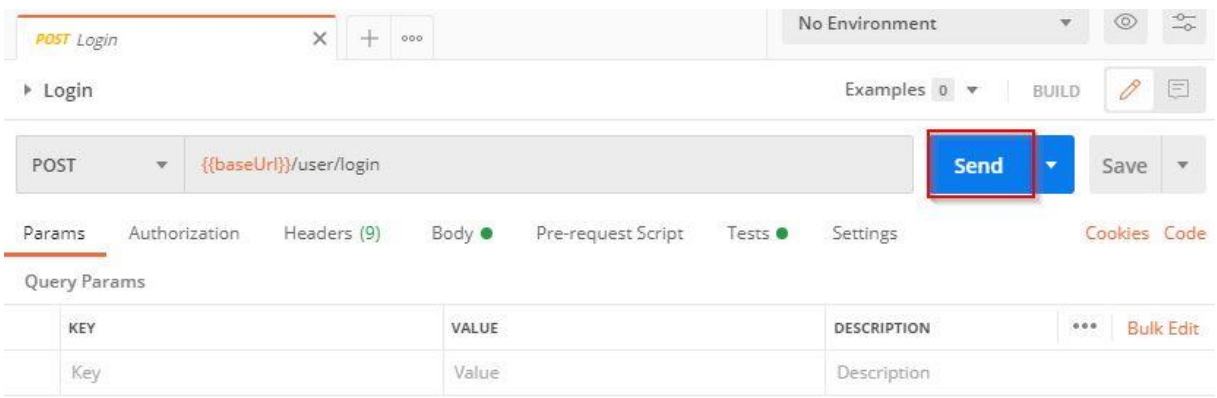


To avoid this error for accessToken variable, you should execute the login endpoint before executing another endpoint.

6.3. Run endpoints

- Run login endpoint

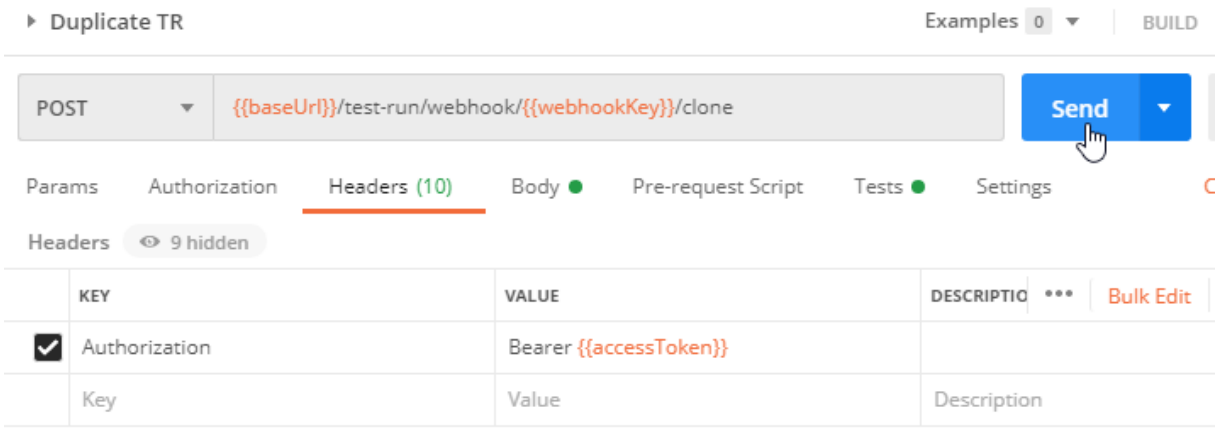
Click on “Send” button



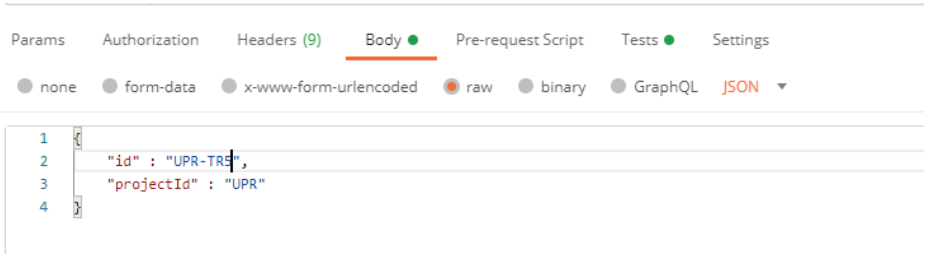
- As response the API returns the accessToken which will be used in the header of the next APIs

- Run Duplicate TR endpoint

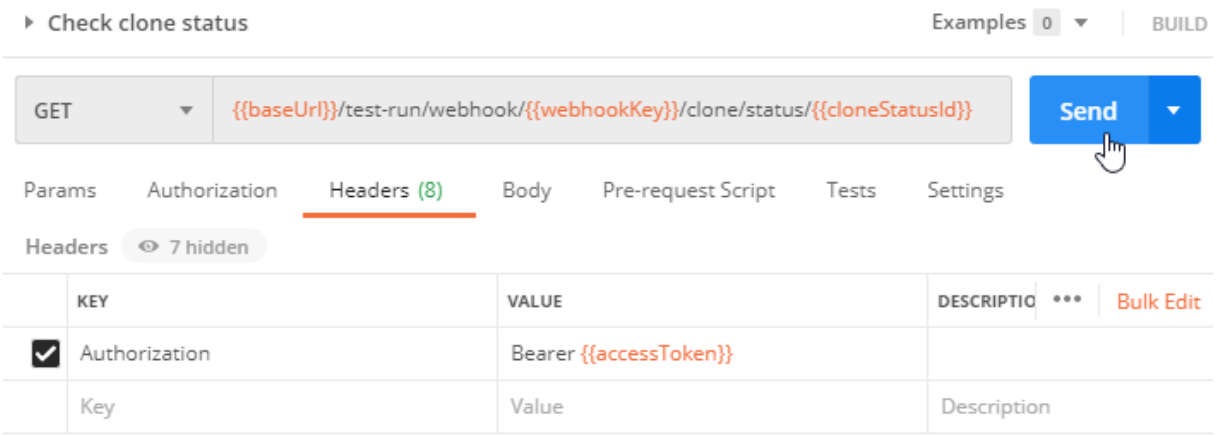
After adding a test run in QACoverage and associate to it test cases, we can run this endpoint



You can use the test run Id and the project Id



- Run check clone status endpoint



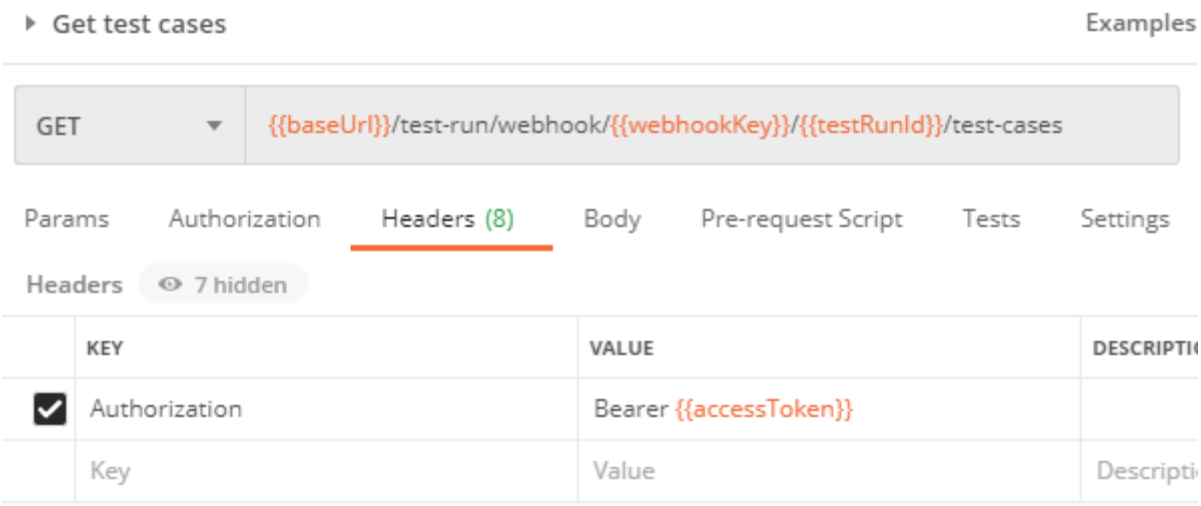
Once the clone status is TERMINATED, the patch API can be executed.

- Run update TR test cases

After adding the body content as defined above we can execute this endpoint.

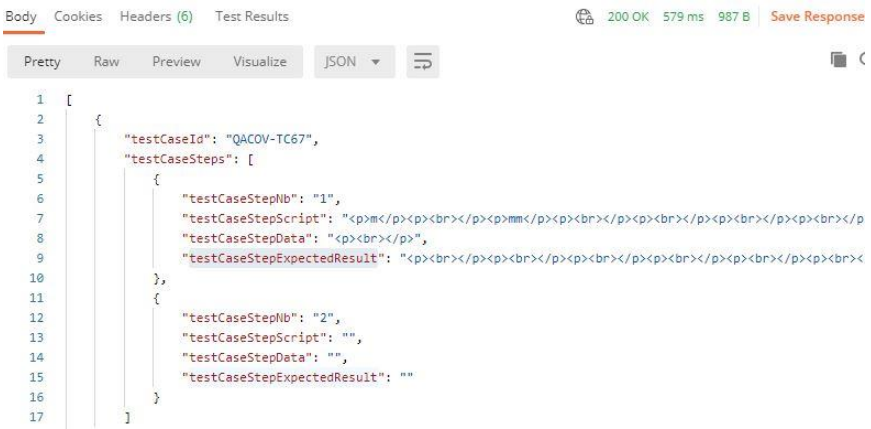
- In case of need, Get Test cases details under cloned test run or existing test run

This API allows you to get the list of test cases associated to test run using OAuth2 authentication executed after login endpoint:



Endpoint: https://app.QACoverage.com/API/test-run/webhook/${webhookKey}/${testRunId}/test-cases

Response Body



7. Conclusion

With test run webhook, you can make changes on test cases results from you application by running a few dedicated endpoints.