GlobalLogic®

Simplifying and Reshaping API Testing Using the Postman Platform: View From the Trenches

by
Nishant Bansal, Senior Software Tester
and Sudhanshu Gupta, Senior QA Lead

Contents

Executive Summary

> "The pandemic has changed the world, but it didn't stop APIs
>
> Nearly 90% of industry members surveyed stated that their organizations offered remote work options as a result of COVID-19. Furthermore, nearly a third (30.6%) said that APIs played a role in their organization's ability to respond to COVID-19 - many utilized APIs for customer communications, powering remote work options, and quickly responding to regulartory changes and government initiatives"
>
> ***State of the API report, Sep 2020***

As the API economy continues to expand, more challenges are created for developers. The old methods of manually creating and testing APIs no longer scale as today's software and services can interface with hundreds of APIs within a single application.

Development, testing, and delivery teams must work together to make sure that applications work seamlessly with APIs to provide a business advantage, rather than cause a business obstacle. Collaboration and operational efficiency are the keys to delivering modern API-powered applications.

Postman has emerged as a superstar in this arena. With a $2B valuation and a 5M developer community across the world, Postman is a force to reckon with in the API development lifecycle space. In one of our projects, we had opportunity to use Postman and Newman as an API testing platform and gained a good amount of insights and experience.

This white paper is an attempt to document these learnings for the benefit of new adopters of Postman and it's CI/CD helper – Newman.

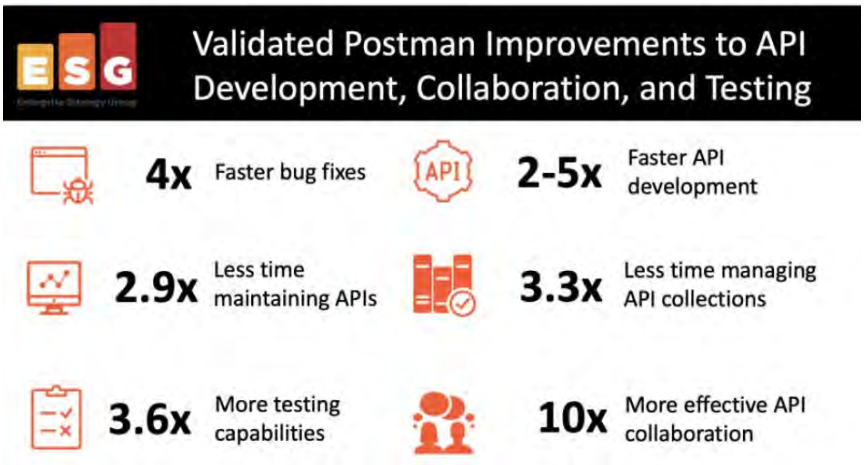
This is a technical report and assumes a basic knowledge of JavaScript as well as initial concepts of API creation and usage. It also assumes that the reader is familiar with the basics of the Postman platform and instead focuses on the nuances and operational learnings gained around API testing using Postman.

Introduction to the Postman API Platform

> The four most important factors individuals consider before intergrating with an API are reliability (71.6%), security (71.0%), performance (70.9%), and documentation (70.3%).
>
> ***2020 State of the API Report***

Postman is one of the best tools in the market today on API automation and documentation. Initially starting its life as a mere Chrome browser plugin, Postman is now a complete API testing solution used by 5 million developers and more than 100,000 companies around the world. It's a unicorn in its own right with a $2B valuation and has emerged as the go-to platform for building enterprise APIs

A recent report from ESG (Enterprise Strategy Group) opines that the Postman API platform provides an essential toolkit for organizations that helps to streamline and accelerate the development, collaboration, delivery, and maintenance of APIs. Two core benefits include faster time to value, which creates earlier revenue streams, and quicker resolution of issues, resulting in lower risk, fewer bugs, and increased customer satisfaction. ESG predicted that a modeled organization could reduce the cost of API testing by 3.6x and API development by 1.9x by using Postman. By using Postman and also implementing an API-first development strategy, organizations could save even more by reducing development time and cost by 4.7x. With Postman, organizations can develop new APIs, monitor existing ones, and perform ongoing testing to eliminate any possible downtimes or slowdowns that could lead to a loss of revenue or customer dissatisfaction. A strong confidence factor was also witnessed in individual developers and development teams using Postman (see inset Figure 1).



As the API economy continues to expand, more challenges are created for developers. The old methods of manually creating and testing APIs no longer scale as today's software and services can interface with hundreds of APIs within a single application. Development, testing, and delivery teams must work together to make sure that applications work seamlessly with APIs to provide a business advantage, rather than cause a business obstacle. Collaboration and operational efficiency are the keys to delivering modern API-powered applications. And this is the space that Postman plays in.

In one of our GlobalLogic projects related to a Banking and Financial Services Customer Communications Management platform, we used Postman and its CI/CD add-on Newman to create, customize and automate Web API tests. This white paper and technical report describes our journey through this automation and showcases some learnings and best practices realized along the way. This is our view from the trenches on the capabilities and potential of the Postman platform for API testing.

Our Foray into Postman and Learnings

> APIs are the nucleus of digital transformation
> For those working on digital transformation initiatives, 84.5% state that APIs are playing a significant role in those initiatives.

At first glance, Postman looks like an interface for sending HTTP requests and viewing responses. But deep down, it is built on an extensive set of powerful tools that are very easy to use. The Postman tool allows you to organize your API requests into collections and folders share common values across requests with environment variables, script tests with the built-in node.js based runtime, and automate them with Newman.

It is a complete API development platform with various built-in tools that support every stage of the API lifecycle. Postman tool allows you to design, mock, debug, automated testing, document, monitor and publish the APIs - everything from one place. Postman can be accessed through native apps for MacOS, Linux and Windows.

One of the most used feature of Postman is **Postman Collections**. As the name suggest this feature lets you group individual requests together and organize these request into folders. A postman Collection can hold sub collections, this helps segregating APIs and is easy to maintain.

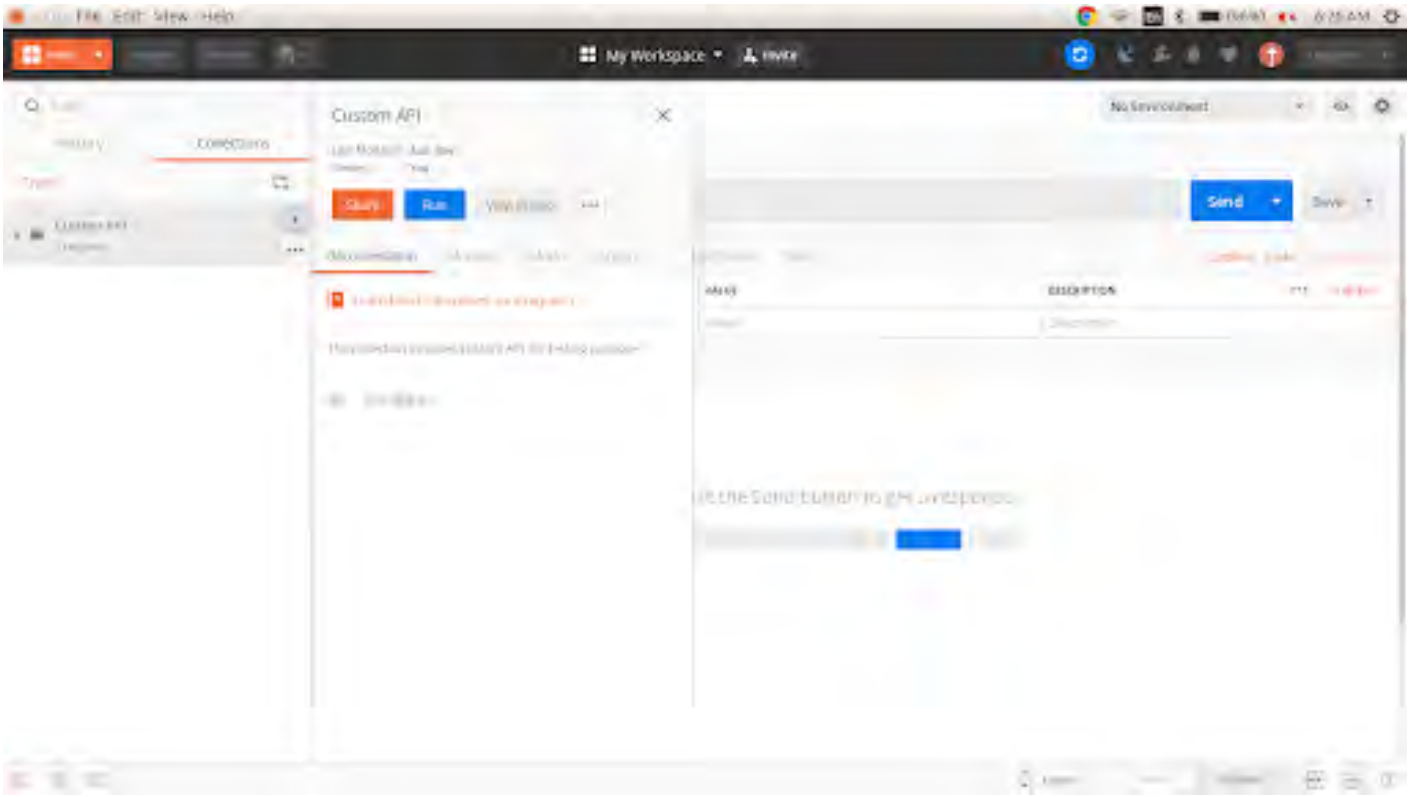


Figure 2: Postman UI showing The Collections feature

Postman also empowers users with a **host of variables which have different scopes** this helps sharing of data within the requests and also within different collection and set of APIs. There are 4 types of variable scope which offers immense power to mix and match your test cases. They are detailed in the table below.

Category	Usage	Javascript Access Functions
Global	Global variables has the supreme scope its used to share variables and functions across various environments	• pm.globals.set() • pm.globals.get
Environment	Environment variables has the scope limited to various collection	• 1, 2, 3, or 6 months. pm.environment.set() • pm.environment.get()
Collection	Collection variables as the name suggests can be shared within different requests of one particular collection	• 1, 2, 3, or 6 months. pm.collectionVariables.set() • pm.collectionVariables.get()
Local	Local variables are local to that request and does not hold any scope outside of that request	• pm.variables.set() • pm.variables.get()

Postman is case sensitive. So pm, variables are considered different from Pm and Variables. Keep this in mind while adding your Javascript code

PRO TIP

Beyond just the data, variables can also store functions and this feature makes it extremely easy to call the functions and use them. Functions can be stored in the variables as function expressions

```
var NameOfTheFunction = () => {

Function Definition

}
pm.environment.set('NameBywhichYouwantedtostoreavariable', NameOfTheFunction.toString());
```

To invoke a function set in an environment variable use the eval() routine

```
eval(pm.environment.get('NameBywhichYouwantedtostoreavariable'))();
```

Variables together with JavaScript functions makes Postman a very efficient tool for executing API, sending HTTP requests and viewing their response. They provide an elegant solution to send requests via any section of the Postman be it the body, pre-request script or post-execution test.

Using the sendRequest function, one can even send API requests directly from the Pre-Requisite Script or post-execution test section. The response of this request gets stored in the function's return response which can then be extracted or saved in variables inside that same request. See the usage below

Function to send the request

```
pm.sendRequest({
    url: url,
    method: 'POST',
    header: header,
    body: {
      mode: 'raw',
      raw: JSON.stringify(body)
    }
  }, function (err, res) {}
```

Function to send the request

```
function (err, res) {
 pm.collectionVariables.set('xxx',res)
}
```

Now the response becomes available in the collection and is available for use to all other requests in the collection. This feature also helps in chaining of the requests to produce desired result. In case there is a need to create some data or run some other requests before running the main request then to make that request independent one can call other request in the pre-requisite script section and create the desired data for the main request

When request chaining strategy is employed, you need to take care of synchronization, i.e waiting for completion of the previous request. Since JavaScript is asynchronous, one has to apply waits in order to guide Postman to wait for the first request to complete. This is shown in the code snippet below

How to handle synchronization

```
setTimeout(function () {

// Operations to be performed on the completion of request if any

},Timeout )
```

Postman also provides a function to jump from one request to another bypassing the order of the requests in the collection and allowing customized flow. Below is the code snippet for the same

Custom request flow (jump)

```
postman.setNextRequest("Request name");
```

This function comes in handy deciding the next request to be executed after the completion of the current request. In our case, we used this function as an authorization tool and upon unsuccessful authorization we set this to null so that no other requests get executed in the collection runner if the user does not have the required authorization.

Custom request flow (stop)

```
postman.setNextRequest(null);
```

The overall code snippet for our use case is provided below:

Pre-Requisite Script (code)

```
setTimeout(function () {
var securedType=pm.collectionVariables.get('securedType')

if (!securedType){
    var securedType=['Secured','Unsecured','Mortgage Secured','Cash Secured']

}

var currentSecuredType=securedType.shift();
pm.variables.set('currentSecuredType',currentSecuredType)
pm.collectionVariables.set('securedType',securedType);
```

Test Code

```
if (securedType!=="" && securedType.length>0){
    postman.setNextRequest("Name of the same request");
}
```

API Body code

```
{
"SecuredType":"{{currentSecuredType}}"

}
```

So the same request gets executed until there is a data in the 'securedType' variable

A few other observations on Postman usage were made as part of the project and are listed below

- Postman also provides a lot of pre-defined assertions to be performed on the results of the API. However one can always amend them and make their own assertions using various JavaScript libraries

- Postman supports a lot of libraries such as Math and Moment and many more. We used moment library in our case to work around and assert dates and the math library to perform various calculations

Example usage of moment library

```
setTimeout(function () {
    var today = moment();
    var PastDate = moment(today).subtract(1, 'M').format('YYYY-MM-DD');
    pm.collectionVariables.set("PastDate", PastDate);
```

Our Foray into Newman and Learnings

Twitter serves 15 billion API calls per day. Twitter serves more APIs than Facebook and Google combined. And they use Postman

With the Postman UI platform, a very robust mechanism of API testing was built. This was extremely easy for developers and SDETs to create their own unit, component and integration tests as well as test APIs as they were being developed. The flexibility and power of collections allowed for this level of coverage. However, Postman's foray into the Agile Continuous Integration, Continuous Development (CI/CD) realm was facilitated by another solution called Newman.

Newman is basically a Command Line Interface tool which will allow you to run Postman collections directly from the command line. Their official website describes Newman as follows: ***"Newman is a command line Collection Runner for Postman. Newman allows you to run collections in the same way they are executed inside a Postman collection runner"***. Since a CI/CD environment helps us to combine different software systems and execute different tests continuously, the applicability of Neuman to facilitate CI/CD continuous testing is crucial.

Newman is built to integrate easily with your build systems and continuous integration server. It allows developers to get a quick feedback about the performance of the APIs after changes in their code. With the help of Newman, it gets integrated with the CI and after that if any changes are pushed, CI will run the postman collections with Newman.

Newman is a node package and so can be easily installed via npm. Newman also provides various extinct reports which describes the health of postman collection and becomes a key matrix for evaluation.

Newman and its reporting engine can easily be installed after the successful installation of node.js by running the following on the command line

Install Newman and its reporter engine using npm
npm install –g newman
npm install –g newman-reporter-htmlextra

The overall code snippet for our use case is provided below:

Run Postman collections in headless fashion using Newman
npm install –g newman newman run collectionName.json –e EnvironmentName.json –r cli,htmlextra -k
Here, -r stands for report and cli stands for command line report

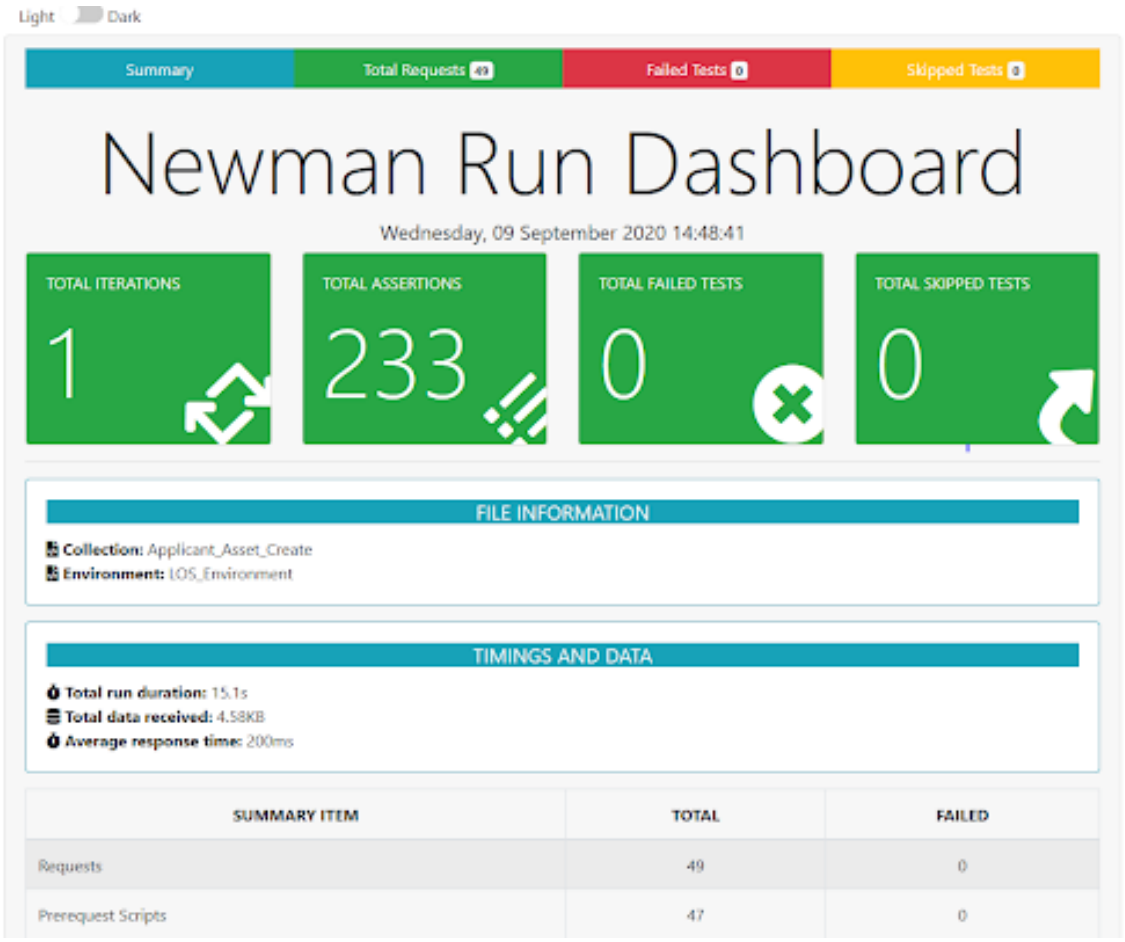
JSON and Junit reports can also be obtained as shown in the snippet below

Run Postman collections in headless fashion using Newman
newman run collectionName.json –e EnvironmentName.json –r cli,json,junit,html –k
This command will result in 4 reports created. One on the command line and three in the newman folder of type json, junit and html.

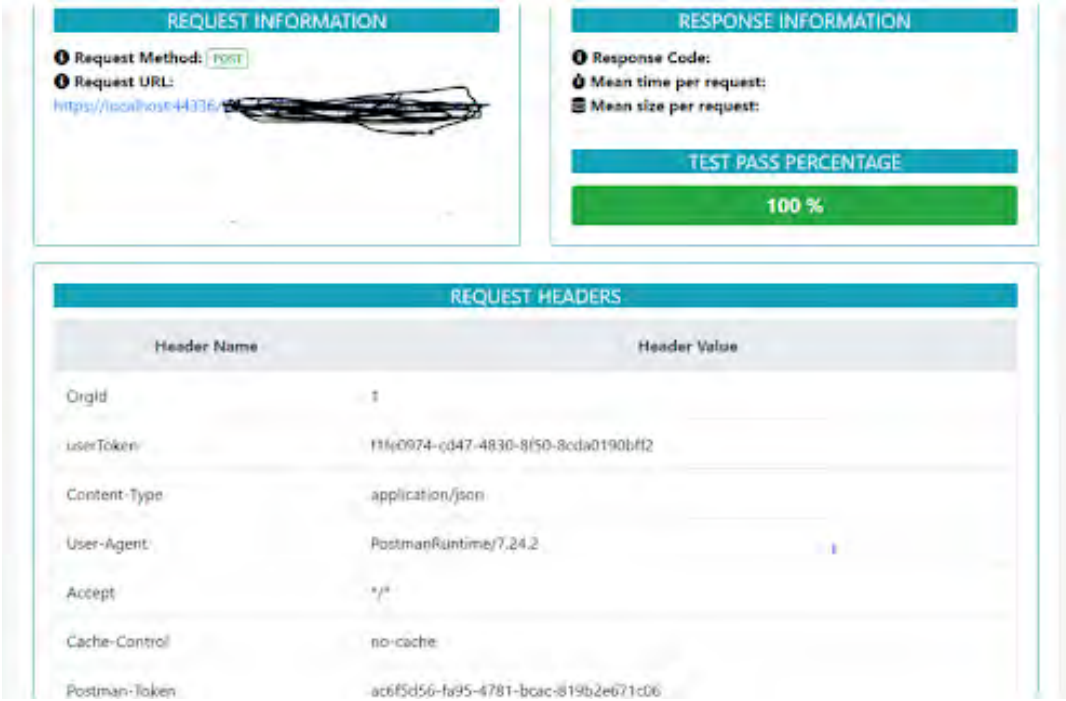
Environment and collection must be placed at the same location to execute the collection or exact path of the same is to be provided while running npm commands

PRO TIP

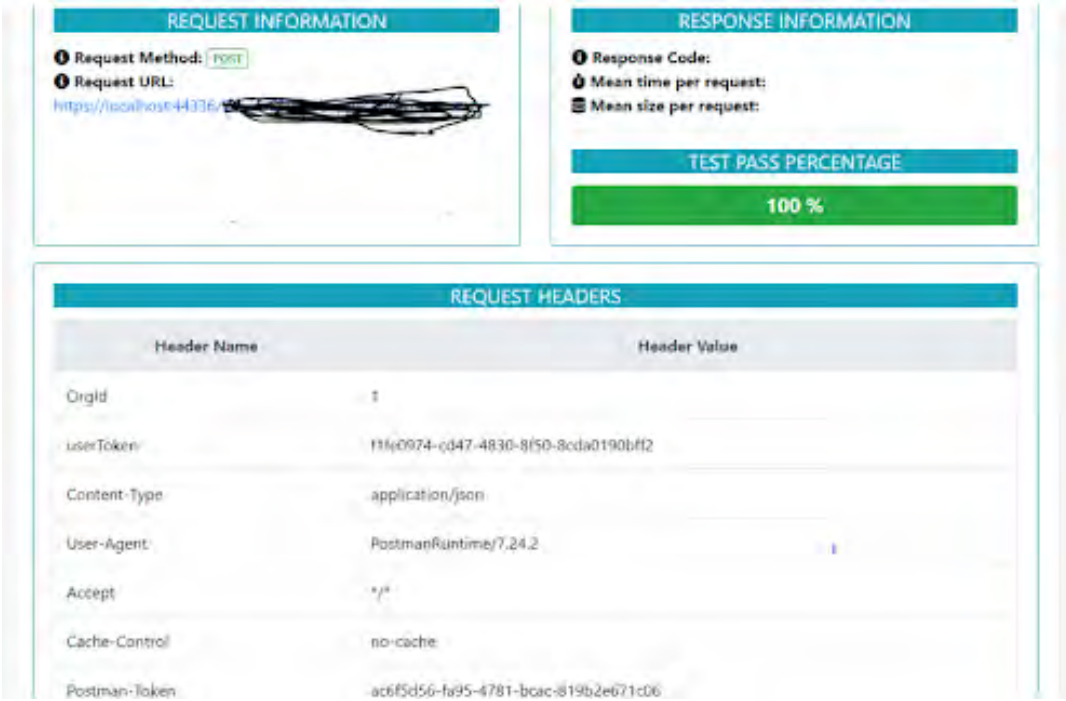
A few screenshots of a Newman run dashboard is shown below:




Light
Dark
Summary
Total Requests 49
Failed Tests 0
Skipped Tests 0
1 ITERATION AVAILABLE TO VIEW
Expand Folders
Expand Requests
1
ITERATION 1 SELECTED
PRE-TESTS - 1 REQUEST IN THE FOLDER
POSITIVE TEST CASES - 6 REQUESTS IN THE FOLDER
NEGATIVE TEST CASES - 40 REQUESTS IN THE FOLDER


REQUEST INFORMATION
Request Method: POST
Request URL:
RESPONSE INFORMATION
Response Code:
Mean time per request:
Mean size per request:
TEST PASS PERCENTAGE
100 %
REQUEST HEADERS
Header Name
Header Value
OrgId
userToken
Content-Type
application/json
User-Agent
PostmanRuntime/7.24.2
Accept
/
Cache-Control
no-cache
Postman-Token


Light
Dark
Summary
Total Requests 49
Failed Tests 0
Skipped Tests 0
1 ITERATION AVAILABLE TO VIEW
Expand Folders
Expand Requests
1
ITERATION 1 SELECTED
PRE-TESTS - 1 REQUEST IN THE FOLDER
POSITIVE TEST CASES - 6 REQUESTS IN THE FOLDER
NEGATIVE TEST CASES - 40 REQUESTS IN THE FOLDER


REQUEST INFORMATION
Request Method: POST
Request URL:
RESPONSE INFORMATION
Response Code:
Mean time per request:
Mean size per request:
TEST PASS PERCENTAGE
100 %
REQUEST HEADERS
Header Name
Header Value
OrgId
userToken
Content-Type
application/json
User-Agent
PostmanRuntime/7.24.2
Accept
/
Cache-Control
no-cache
Postman-Token

Conclusion & Call to Action

> "If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas."
>
> ***G.B. Shaw***

Postman as an API testing platform is very robust, powerful and flexible. With Newman as an add-on, having a fully configurable continuous testing platform for APIs is very real. The learnings gained in our project leads us to believe that most development projects will benefit with the usage of Postman toolchain for their continuous testing of APIs.

We have included below a short recap of the salient learnings in our project as a quick reference

Learnings from Postman	
1.	Postman Collection and its importance in organizing related requests and segregating APIs
2.	Variable and their scope Binding Function to variables and the art of invoking them thereafter
3.	Sending requests from any section of Postman and fetching response in collection variable
4.	Skipping Request or hoping from one request to another using SetNextRequest() Recursion using setNextRequest()
Learnings from Postma	
5.	Newman as Command Line runner for Postman
6.	Newman and its Supported reports
7.	Detailed Extent Report by htmlExtra and Newman

About the Authors

Nishant Bansal is a senior software tester at GlobalLogic. His core area of expertise is in Automation Testing and he has successfully executed and delivered various API scripts using Postman. In addition, he also has experience in automating UI using selenium and Robot framework

Sudhanshu Gupta is a Senior QA Lead at GlobalLogic with 11+ years of IT experience. He has experience in creating automation solutions and frameworks for large scale complex digital programs. He is an expert in automation across tech stacks (Java, Javascript and Python) and on different channels (Web, API & Mobile Apps) and also has experience in designing and implementing the CI/CD pipelines. Sudhanshu is recognized across the company for bringing in best industry practices for coding standards, extensible framework design, and low maintenance automation solutions.

References

The Economic Benefits of the Postman API platform - ESG report published in May 2020

2020 State of the API report commissioned by Postman

GlobalLogic®

GlobalLogic is a leader in digital product engineering. We help our clients design and build innovative products, platforms, and digital experiences for the modern world. By integrating strategic design, complex engineering, and vertical industry expertise,— we help our clients imagine what's possible and accelerate their transition into tomorrow's digital businesses. Headquartered in Silicon Valley, GlobalLogic operates design studios and engineering centers around the world, extending our deep expertise to customers in the communications, automotive, healthcare, technology, media and entertainment, manufacturing, and semiconductor industries.



www.globallogic.com