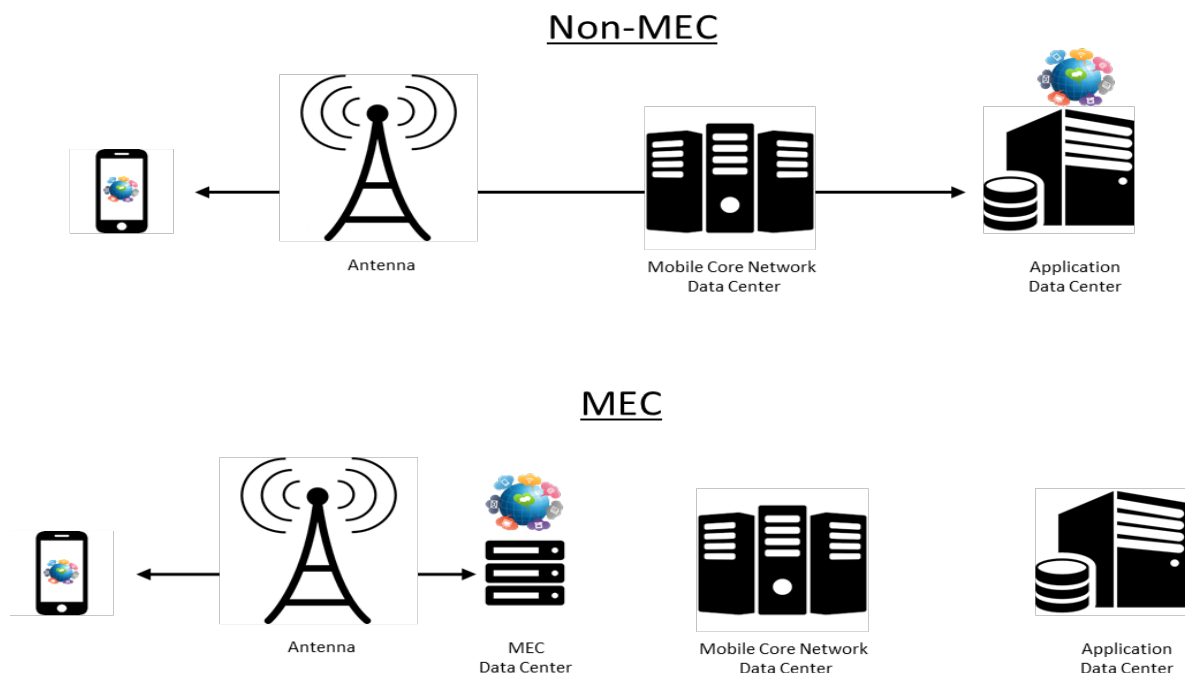


MEC

(MULTI-ACCESS / MOBILE EDGE COMPUTING)

Introduction:

Multi-access Edge Computing (MEC), also described as a Mobile Edge Computing is an industry innovation that improves the performance and intelligence of the software application that deliver services such as IoT (Internet-of-Things) to end customers. The improved ultra-low latency and high bandwidth performance are made possible by the deploying applications in the data center at the edge of the network closer to the customer. The improved intelligence by real-time access to radio network information that can be leveraged by applications.



To achieve the promises made by MEC technology the edge data center support MEC framework and deployments as defined by European Telecommunications Standards Institute (ETSI) in standard specifications ETSI GS MEC 003 V1.1.1 (2016-03). In Summary, the implementation minimum requires the MEC platform that hosts and manages MEC applications on NFV infrastructure. Also, the MEC and

NFV management manages its respective domains as shown in the simplified Figure 02.

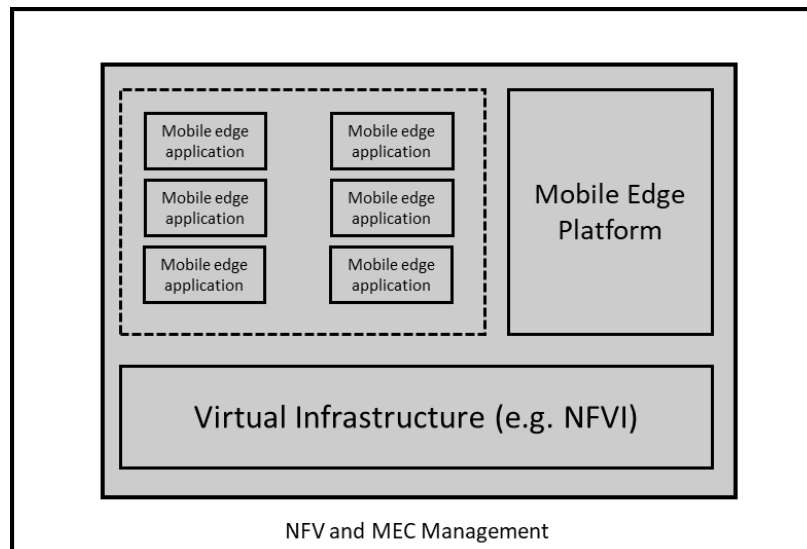


Figure 02: MEC Framework

The next generation technology such as MEC and NFV requires protocols to communicate between antenna (eNodeB), MEC platform, MEC applications, NFVI and management entities. The connective between these entities is called "Reference Points". ETSI defines the API, data model and data format in the specification for these reference points.

*"What If testers have a solution that provides lot of test cases to ready-to-go **client and server side** API conformance test cases?"*

TestOnNeed offers precisely such a solution to our customers ready to go to test at client side and server side.

Client side Solutions are:

1. MEC EveryAPI Conformance Test Suite.
2. MEC EveryAPI Test Engine.

2. Installation and Configuration of MEC EveryAPI Test Engine: -

For the installation of the application and setup of the applications and software's which we have used to achieve the automation solution, please refer to the installation videos. We will require Apps / Software's:

1. Node (Ex: Local Apache Server)
2. Postman
3. Test Link (User for management of the Test cases)
4. Editor/ MS Excel to view and edit Datasheet.

check the below link to install the Node, Postman: -

<https://www.youtube.com/watch?v=gJlOaAMAPWU> (Windows)

<https://www.youtube.com/watch?v=PXRJ4VG4HAK> (Ubuntu)

After Purchase the MEC EveryAPI Test Engine, User can use different types of Yaml files to get the Postman Collection file. While executing, MEC EveryAPI Test Engine will take the Input as a yaml file and automatically it creates the business logic and test cases based on the input file specification. During the execution, the Postman Application will use a CSV file to extract the data to be used for each test case execution.

Sample YAML Example:

```
---
swagger: "2.0"
info:
  description: "API described using OpenAPI"
  version: "1.1.1"
  title: "Sample API"
  license:
    name: "xxx copyright notice"
    url: "https://copyright-notice.txt"
host: "127.0.0.1:8081"
basePath: "/sample/v1"
schemes:
- "http"
- "https"
consumes:
```

```

- "application/json"
produces:
- "application/json"
security:
- OAuthSecurity:
  - "all"
paths:
  /APIName:
    get:
      description:
      operationId:
      produces:
      parameters:
      - name:
        in:
        description:
        required:
        type:
        items:
          type:
        collectionFormat:
      responses:
        200:
          description:
          schema:
            $ref:
        400:
          description:
          schema:
            $ref:
      x-swagger-router-controller: "Default"
  /APIName/{arguemnt1}:
    get:
      description:
      operationId:
      produces:
      parameters:
      - name:
        in:
        description:
        required:
        type:
      responses:
        200:
          description:
          schema:
            $ref:
        400:
          description:
          schema:
            $ref:
      x-swagger-router-controller: "Default"
securityDefinitions:
definitions:
  ProblemDetails:
parameters:
externalDocs:
  description:
  url:

```

1. MEC EveryAPI Test Engine executes It will create the test cases file automatically based on the Input file which can be executed by importing to the Postman Application.

I) Once we run the engine with Input standard yaml file will get the Postman Collection. Take that Output Postman collection file import into Postman Application.

Please follow the below points and image how to execute the yaml files using commands in the Engine.

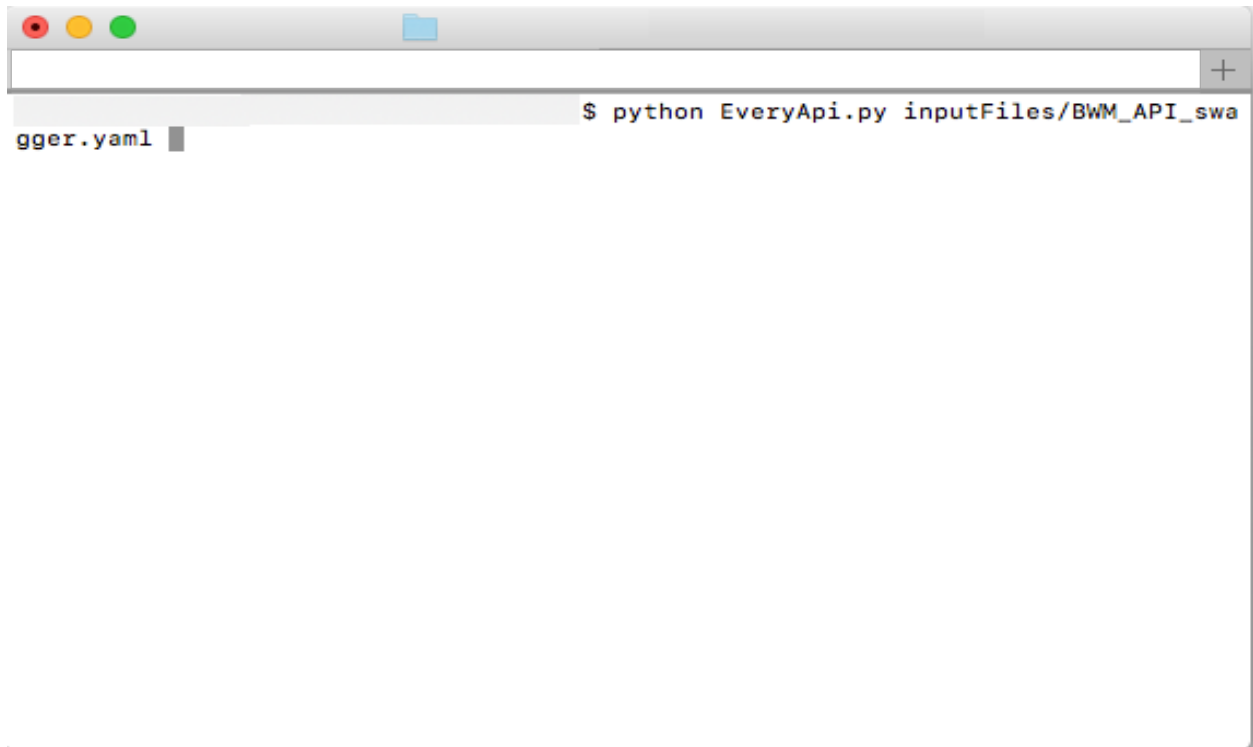
- ➔ Once get the Test Engine have to unzip Folder
- ➔ Next, Open Project Folder
- ➔ Next, Need to open Command Prompt (or) Terminal with the Test Engine Location.
- ➔ Once Open the Terminal, The Below Command have to enter
Mac: -

```
$ python EveryApi.py inputFiles/BWM_API_swagger.yaml
```

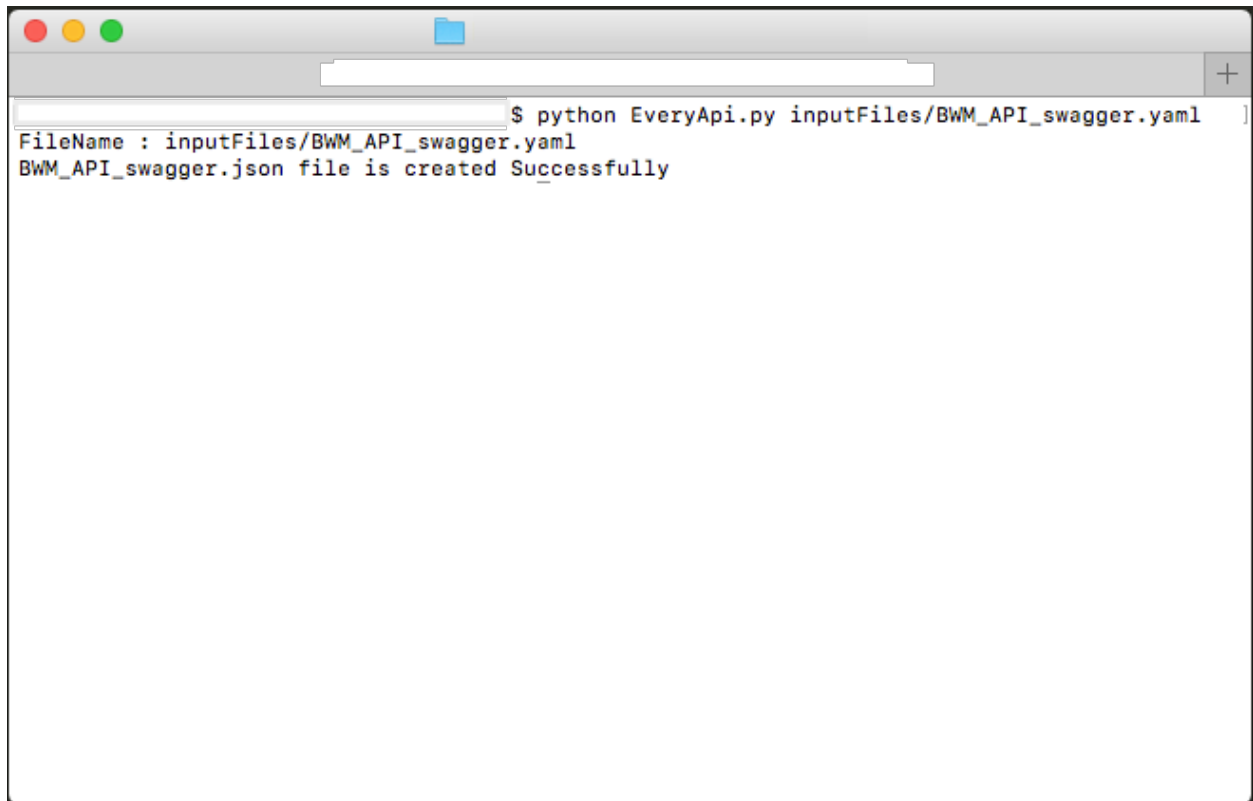
Windows: -

```
C:\MECTestEngine> python EveryApi.py inputFiles/BWM_API_swagger.yaml
```

- ➔ Test Engine Requires Python to run the Engine. 'EveryApi.py' is index Execution file. 'InputFiles/BWM_API_swagger.yaml' is the Location for Input Yaml file.

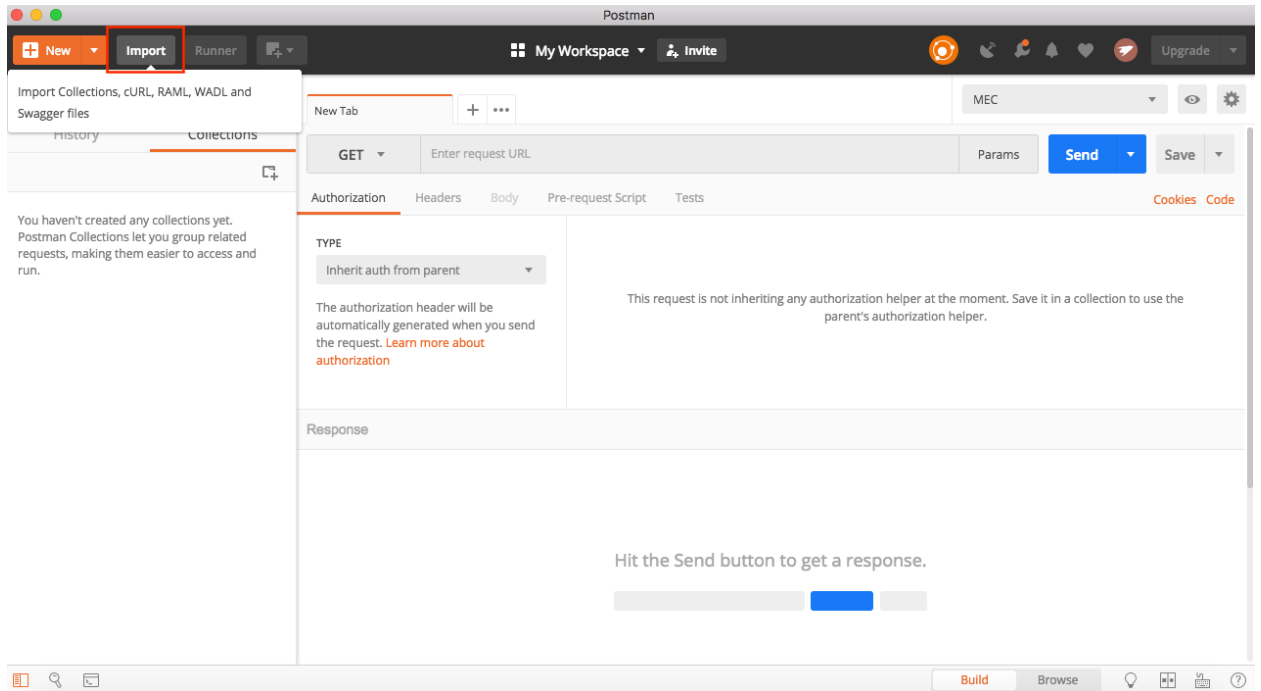


- II) After Entering the Command in terminal click on Enter Button. It will execute and display the Log of execution and Output file Location in the terminal. Refer the below image

A screenshot of a macOS-style terminal window. The window has a title bar with red, yellow, and green window control buttons on the left and a blue folder icon in the center. The terminal text shows a command being executed: '\$ python EveryApi.py inputFiles/BWM_API_swagger.yaml'. Below the command, the output is displayed: 'FileName : inputFiles/BWM_API_swagger.yaml' followed by 'BWM_API_swagger.json file is created Successfully' on the next line. The terminal has a light gray background and a white text area.

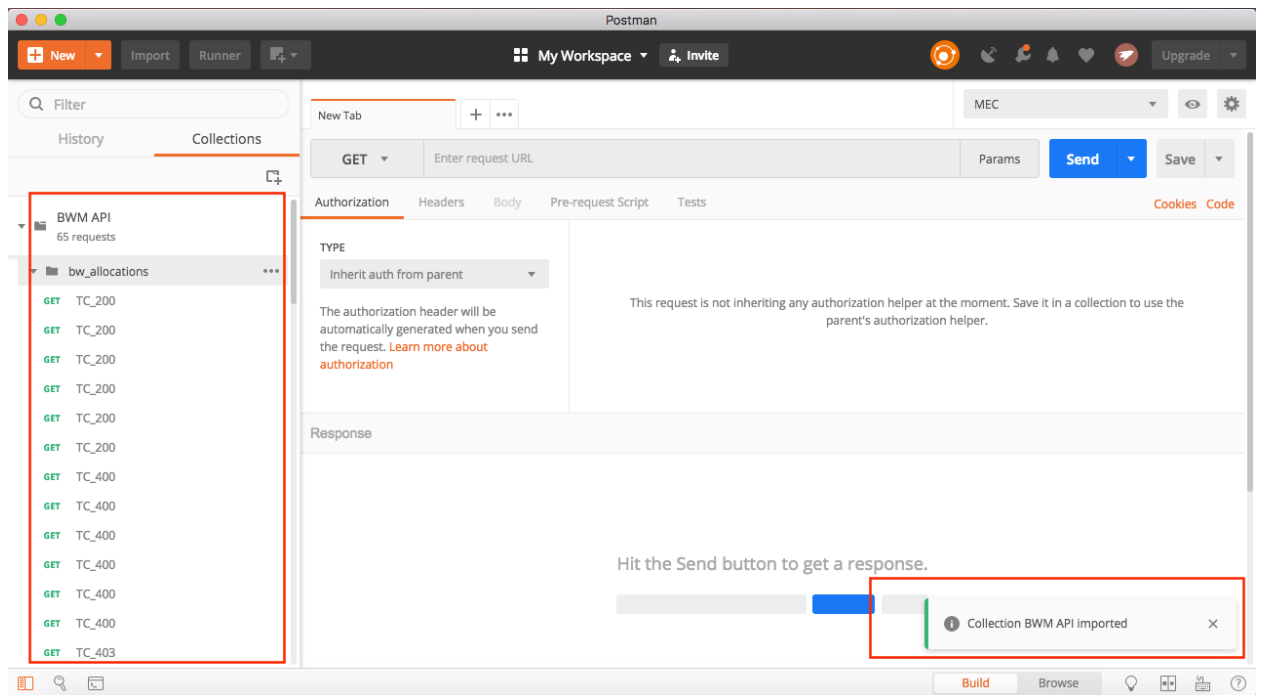
```
$ python EveryApi.py inputFiles/BWM_API_swagger.yaml  
FileName : inputFiles/BWM_API_swagger.yaml  
BWM_API_swagger.json file is created Successfully
```

2. I). Take that Postman collection JSON file from the Output Folder Location and import into Postman Application and see the test cases on the left. Please refer the below image.

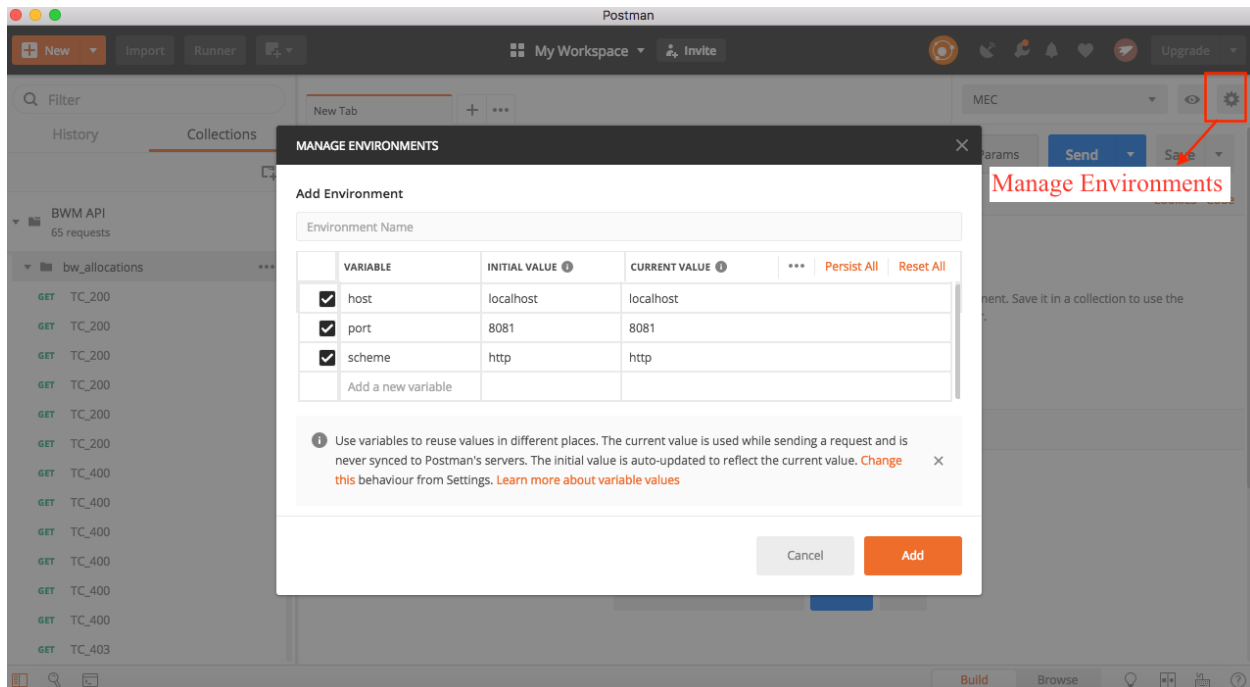


II) Once the User import the Postman collection JSON file in to the Postman Application then left side section one folder will add automatically and user can see the message as a Imported Successfully. If we extract the folder user can see the lots of test cases which are built based on the Input file specifications.

Please refer below image After import the JSON file into Postman Application:



3. I). Before Execute each test cases user have to provide the Environment global variables and values (local / Production / Development server IP configuration) in the Postman Application. The below image depicts how to set up the Global Param's and values in the Postman Application.



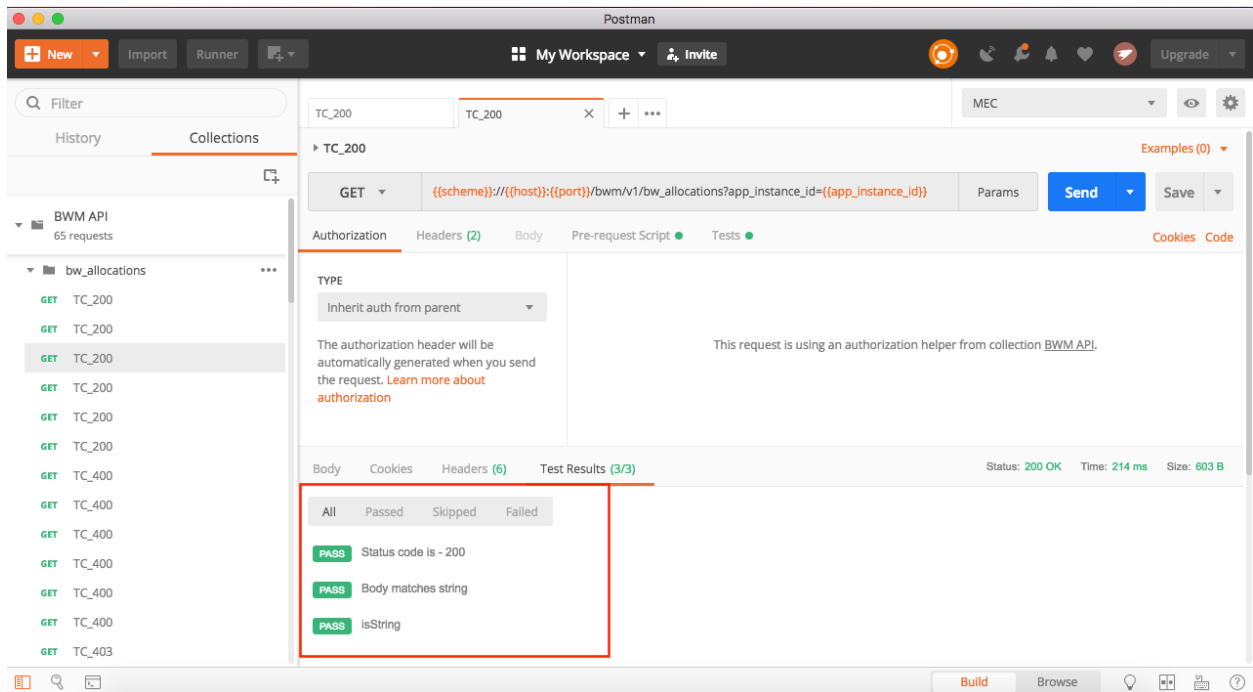
II). After Adding the Global Variables and Values, user have to select the Environment Name in the Dropdown from the right side of Postman Application.

After setup Global Values in the Postman Application, During the execution, the Postman will use a csv file to extract the data to be used for each test case execution.

4. How the CSV will work while Execution, In the Postman Application for 1st test case (Ex: GET 200) from the CSV file it will take 3rd row of values based on the parameters required. Another example: For the 4th from the CSV file it will take 6th row of values based on the parameters required, Like for rest of all test cases.

In the CSV file 1st row occupied with Request Object tag names in the Input File mentioned and 2nd row occupied with the data type of tag names.

5. After Executing test case clicking on the send Button in the Postman Applications user can see the Test Results. Please refer the Below Image for Test Results in Postman Application.



Please click on the Below YouTube Link for Video: -

<https://youtu.be/OoZE2T98CDc?list=PLfDSBK5ZoW6SK0wCQnvkZ3hLM9ELjeJX5>

Thank You,
TestOnNeed,
sales@testonneed.com

Contact Information:

TestOnNeed is an on-need open source test, automation, and DevOps solution provider to better the speed, scale, coverage, and quality of software application products. We live only to help Business seeking the winning formula to bring their products, and services to the market fast to beat the slow. We do this by helping our customers to create flawless, high quality, high-performance Block chain, Artificial Intelligence (AI), Internet of Thing (IoT), Multi-Edge (Mobile) Edge Computing (MEC), 5G, Cloud, Micro services, Augmented Reality (AR), and Virtual Reality (VR) software application products. Moreover, we make it all happen with open sources using an open source testing ecosystem.

At TestOnNeed, we don't just test products; we make products better. Being fast is all about transformation and success is all about welcoming it. To discover more, visit us at <https://testonneed.com/> If you need additional information or have questions about our solutions or demo and Purchase, please reach us at <mailto:sales@testonneed.com>