IOT EVERYAPI TEST ENGINE – USER MANUAL

Contact:

sales@testonneed.com

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Sairam C.	First Draft version	

Review & Approval

Requirements Document Approval History

Approving Party	Version Approved	Signature	Date

Requirements Document Review History

Reviewer	Version Reviewed	Signature	Date

Table of Contents

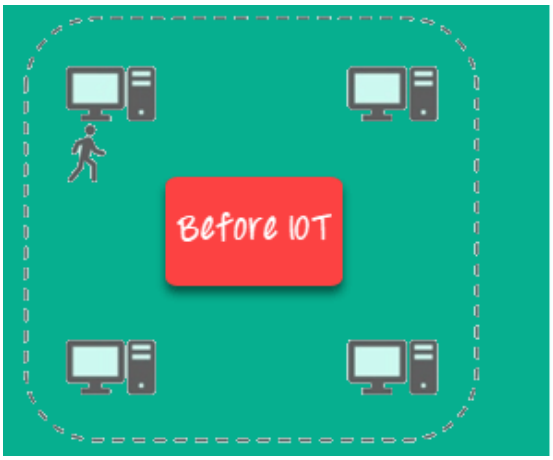
1. INTRODUCTION TO IOT

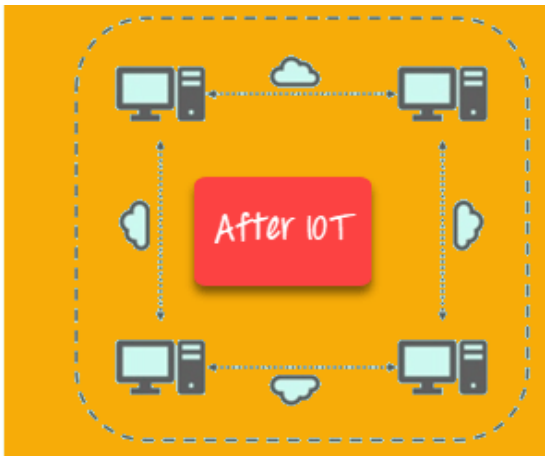
Internet of Things (IoT) is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment.

In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.

The picture below shows the architecture of the systems in action. The picture shows that the systems in the era before IoT came to existence are not inter-connected and the data has to be synced manually by some human interference which brings in errors and cross platform dependencies to be handled. With the invent of IoT, these problems have been solved and these systems are not inter-connected and they can communicate among themselves as they follow the standards mentioned by the standard bodies.

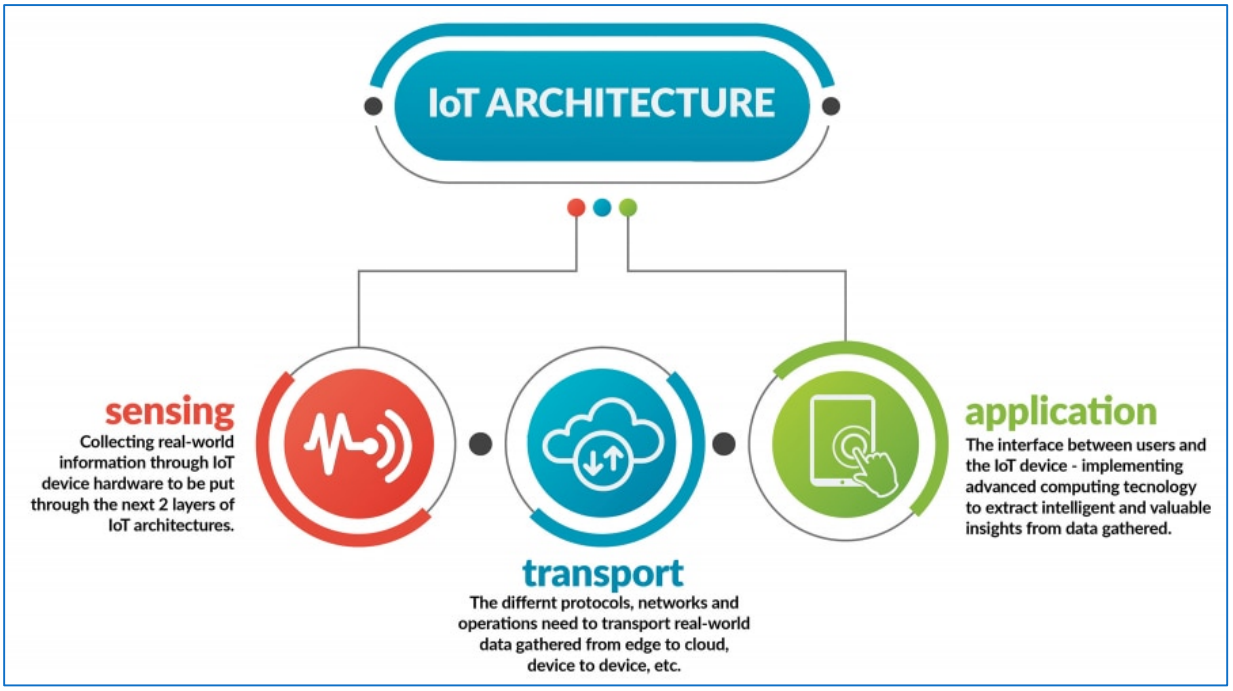
In a nutshell, IoT is a concept that connects all the devices to the internet and let them communicate with each other over the internet. IoT is a giant network of connected devices – all of which gather and share data about how they are used and the environments in which they are operated.





2. ARCHITECTURE OF IOT

The IoT technology is supported by the applications, hardware sensors, cloud servers and the deployments are achieved as defined by oneM2M organization. IoT architecture is the system of numerous elements: sensors, protocols, actuators, cloud services, and layers. Given the complexity of the IoT architecture, a number of elements are chosen to steadily include these various types of components into a sophisticated and unified network.



IoT architecture contains devices and user management components to provide stable and secure functioning of things and control user access issues.

Developing an IoT architecture of a particular solution, it's also important to focus on consistency thus, giving enough attention to every element of IoT architecture and making them work together. We also need to think about flexibility so as to have the opportunity to add new functions and new logic. Finally the integration with enterprise systems like teaming up new IoT solutions.

3. CHALLENGES IN IOT TESTING

In today's world, testing is an integral aspect as the customers are conscious about the quality of the product that is being used. The IoT server and the applications on the IoT server needs to be tested for the applications to behave in the expected way.

The testers today have to go through the specifications documents from the standard bodies and graph out the test cases for testing the applications and the server. For a company which is developing the IoT server, it is a mandatory and a resource consuming task for them to create the test cases reading the standard and deriving the test cases. The greater challenge in testing is the test management which includes the creation, execution and reporting.

This process is a high resource consuming process which needs to be addressed and in the world of automation, machine learning where applications are becoming more intelligent and becoming capable to performing tasks themselves, we need to find a solution and get higher return with minimum investment.

This is what, TestOnNeed specializes in and here we are to offer the Test Engine solution to our customers which is all about the automation of test management.

4. SOLUTION TO THE CHALLENGES

"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 who are ready to test at client side and server side.

We provide a ready to use solution to the customers which help them to achieve quick results for testing the client side. This solution will help to detect problems at an early stage which will reduce the cost of the product development cycle.

This solution is concentrated on the creation of new APIs using the engine which can be used by the customer to create the Client and Server code for any IoT specification using a YAML file of a particular specification of IoT. This then validates the queries sent from the client to the database and checks the expected response to the actual response.

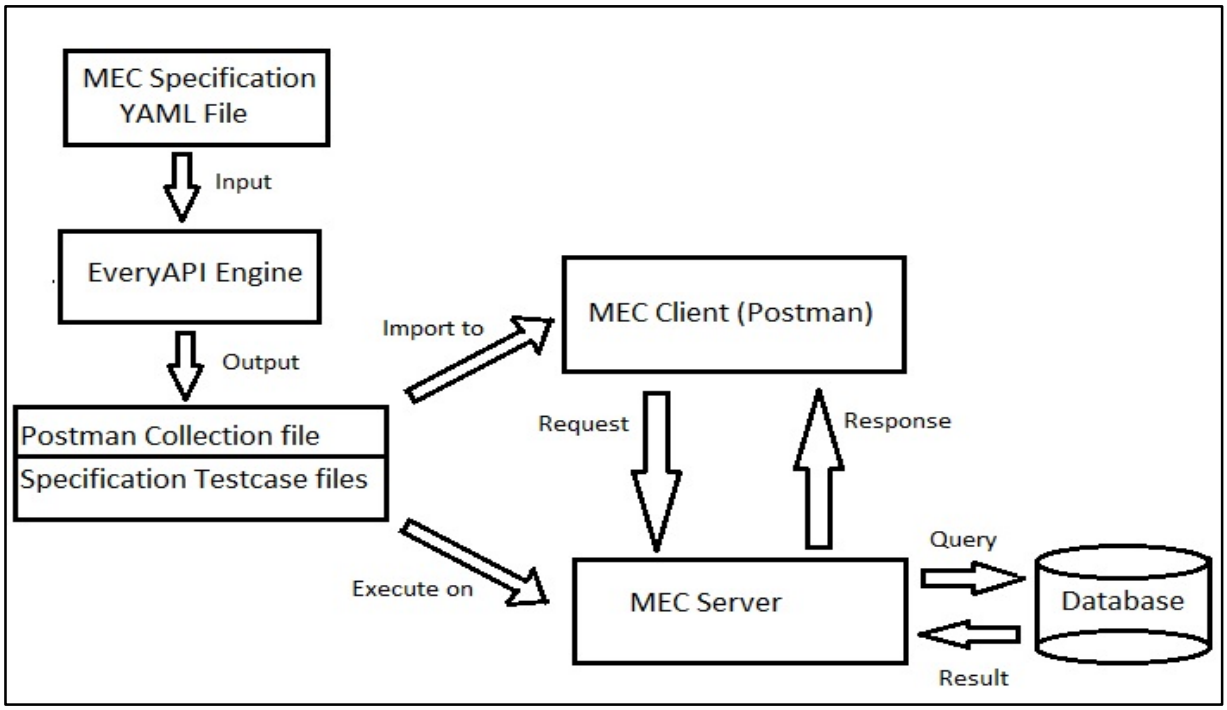
In this document we will describe in details about the Test Engine solution and how it will help the customers to gain from the solution.

You can also link of the following link and check the channel for all the videos which will help get a clear idea on how TestOnNeed can help you achieve the perfect IoT Client or Server application by helping you to detect the shortcomings in the application. The ready to use test cases have been built following the standard solution document created by oneM2M.

Channel for IoT Conformance Testing -
https://www.youtube.com/playlist?list=PLfDSBK5ZoW6SK0wCQnvkZ3hLM9ELjeJX5

5. ARCHITECTURE OF TEST ENGINE SOLUTION

The Test Engine solution is aimed at automating the tasks executed by functional and conformance testers to determine the quality of the IoT server and the applications which are using the IoT server, which are the clients of IoT server.



IoT Specification YAML File – This is the specification document served by standard body for IoT standard which is in YAML format. This is the only file the Test Engine Solution accepts as input.

EveryAPI Engine – This engine accepts the specification YAML file and parses it to create the Specification Test case files which will be used by IoT server to serve requests from IoT client. The test data, test request body and response validation will be formatted and will be formed in Postman collection file.

Postman Collection File – This file will be formed automatically by the EveryAPI server which will take care of the request body, response validation. This file is meant to be used for IoT client.

Specification Test case Files – This is the server code which will be generated automatically by the EveryAPI Test Engine which will be working on the IoT server. This file contains the test cases broken down after parsing the input YAML file based on parameters, request type and response code. This file will be capturing the response for each test case execution.

IoT Client (Postman) – Here Postman Application is being used as client application which will be able to fire the queries to the server based on the test case. Here the response validation, request validation is being validated.

IoT Server – This is the engine which will be responsible to respond to the requests sent from client end. This IoT will be validating the authentication of the requests, the structure of the requests, the parameters being passed and will respond accordingly. The server will be fetching and inserting data in to the database.

Database – This is the container for the test data being generated by the EveryAPI engine to be stored, inserted and to be used by the IoT server. This server will also be recording the test case execution status after each execution.

6. PREREQUISITES FOR TEST ENGINE

The installation of the application and setup which have been used to achieve the automation solution, please refer to the installation videos below.

To install on Windows: https://www.youtube.com/watch?v=gJlOaAMAPWU

To install on Linux: https://www.youtube.com/watch?v=PXRJ4VG4HAk

Listed below are the required software applications. All the applications being used are open source.

- **Postman**

  This is used to record and automate the process of posting requests to server and fetching the corresponding responses back while the user will be executing the actions.

- **NodeJs**

  This will be used by the solution for the generating the requests which will be used by the solution for execution of test cases.

- **Test Link**

  This will be used by the testing professional to keep a track of the test cases and specification document. The test plan will be provided to the customer and can be imported to the application after installation.

- **Sublime Text**

  This will be used as an editor to view the test data by the test professional.

- **Mongo DB**

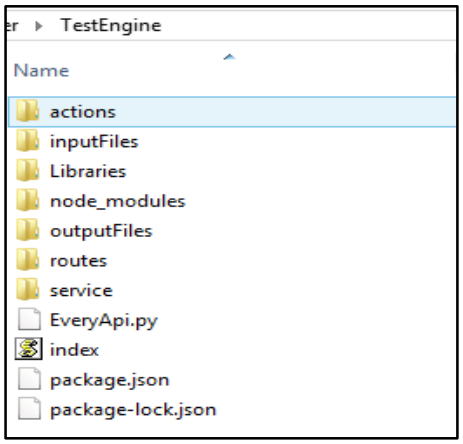
  This is the tool used for the NoSQL Database which is used by the Test Engine to store data into and fetch data for test case execution.

- **Robo 3T**

  This is used as an interface to the Mongo database server which will help the user view the data stored in the database in tabular and JSON formats.

7. DESCRIPTION OF TEST ENGINE SOLUTION

The deliverable package contains a set of a python compiled files which make up the Test Engine. The Test Engine code accepts IoT YAML files as input to it and provide multiple files as output. The whole engine will create a Postman Collection JSON file which will contain the client code and will be imported to Postman application for execution.

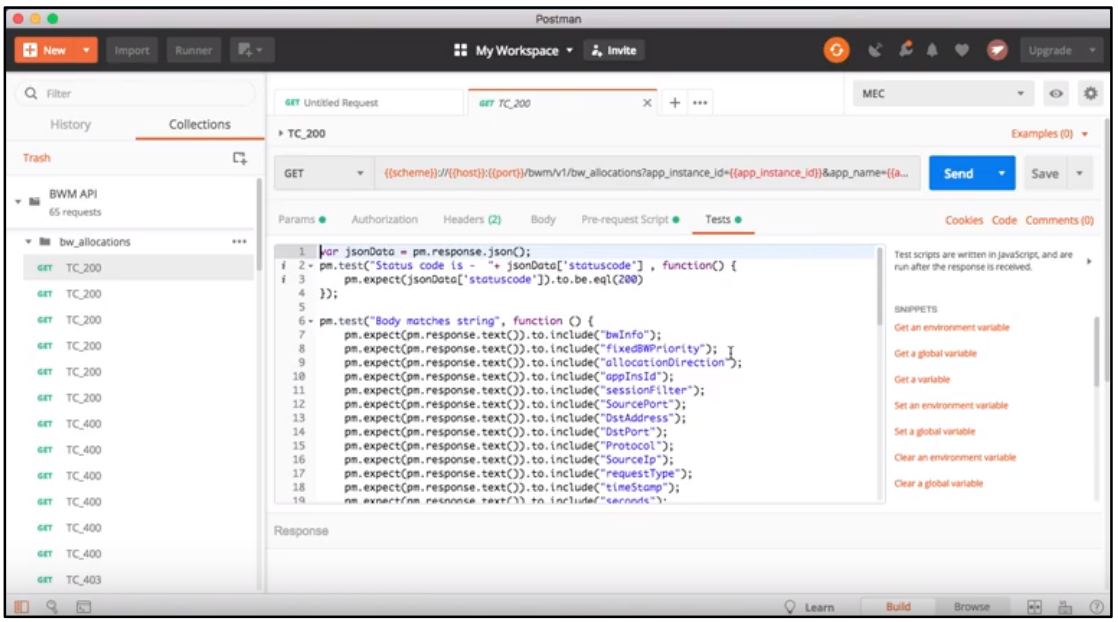


The customer will be provided with the Server end code which will be able to handle the requests sent from the Postman application.
The backend requests will be generated automatically and the API mapping will be performed by parsing the YAML file provided by the user.

Here the user will be able to generate mapping API for any IoT specification as per the current deliverables.

During the execution the requests will be generated from the Postman application and will be executing the test cases one by one as mentioned in the Test Link test case document.

The Postman Application will extract data from the database for each test case execution. The database tables will be provided to the customer as a part of the deliverable package.



The data from the database will be fed to the application for execution of the test cases. The solution will be able to take the requests mentioned in the collection file which will be imported to postman.

The request will be served by the IoT server and will send the response to the client end which will then be validated based on the parameters like data type, cardinality.

The customer will also get a set of links of videos which they can follow and execute the test cases with ease and reach the market quickly with a quality product.

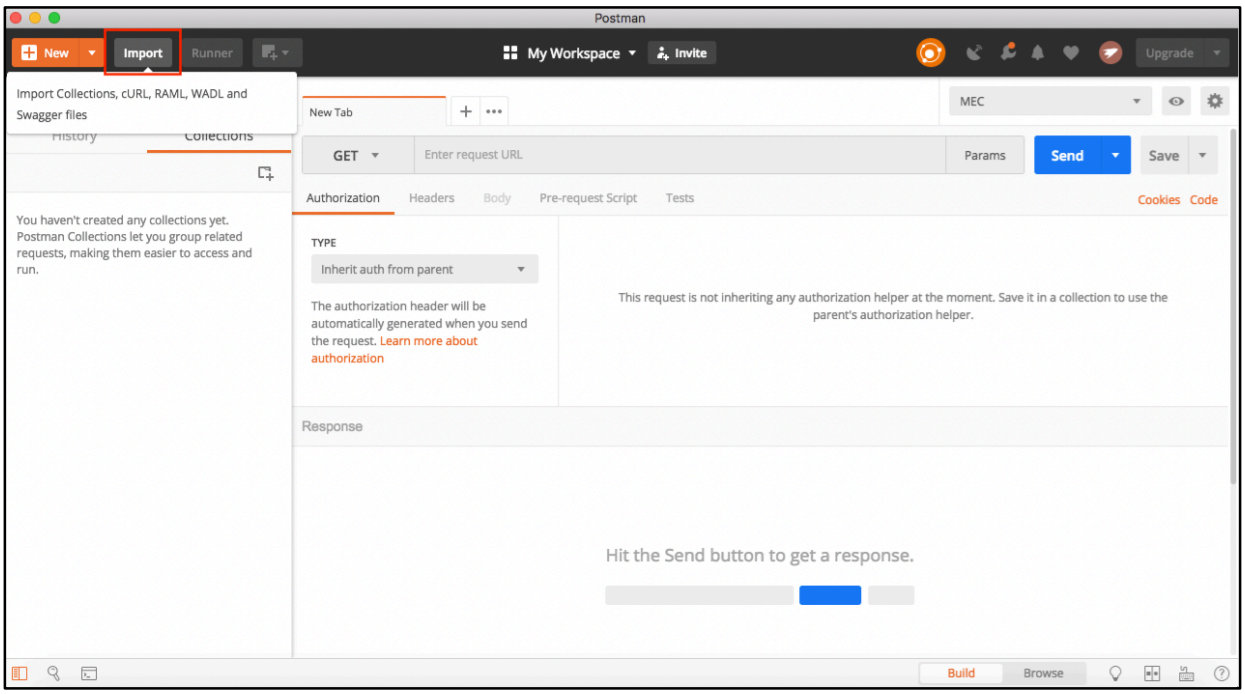
```
---
swagger: "2.0"
info:
  description: "The ETSI MEC ISG MEC011 Application Enablement API described using\
    \ OpenAPI"
  version: "1.1.1"
  title: "Mp1 API"
  license:
    name: "ETSI Forge copyright notice"
    url: "https://forge.etsi.org/etsi-forge-copyright-notice.txt"
host: "127.0.0.1:8081"
basePath: "/exampleAPI/mp1/v1/"
schemes:
- "http"
- "https"
consumes:
- "application/json"
produces:
- "application/json"
paths:
  /applications/{appInstanceId}/dns_rules:
    get:
      description: "This method retrieves information about all the DNS rules associated\
        \ with a mobile edge application instance."
      operationId: "ApplicationsDnsRules.GET"
      produces:
      - "application/json"
      parameters:
      - name: "appInstanceId"
        in: "path"
```

8. PERFORMING CONFORMANCE TESTING

a. The deliverable package will contain a set of Python code which make up the Test Engine. The test engine is capable to creating the client and server end code for any IoT specification when a YAML file in the supported format will be fed as input to the Test Engine.

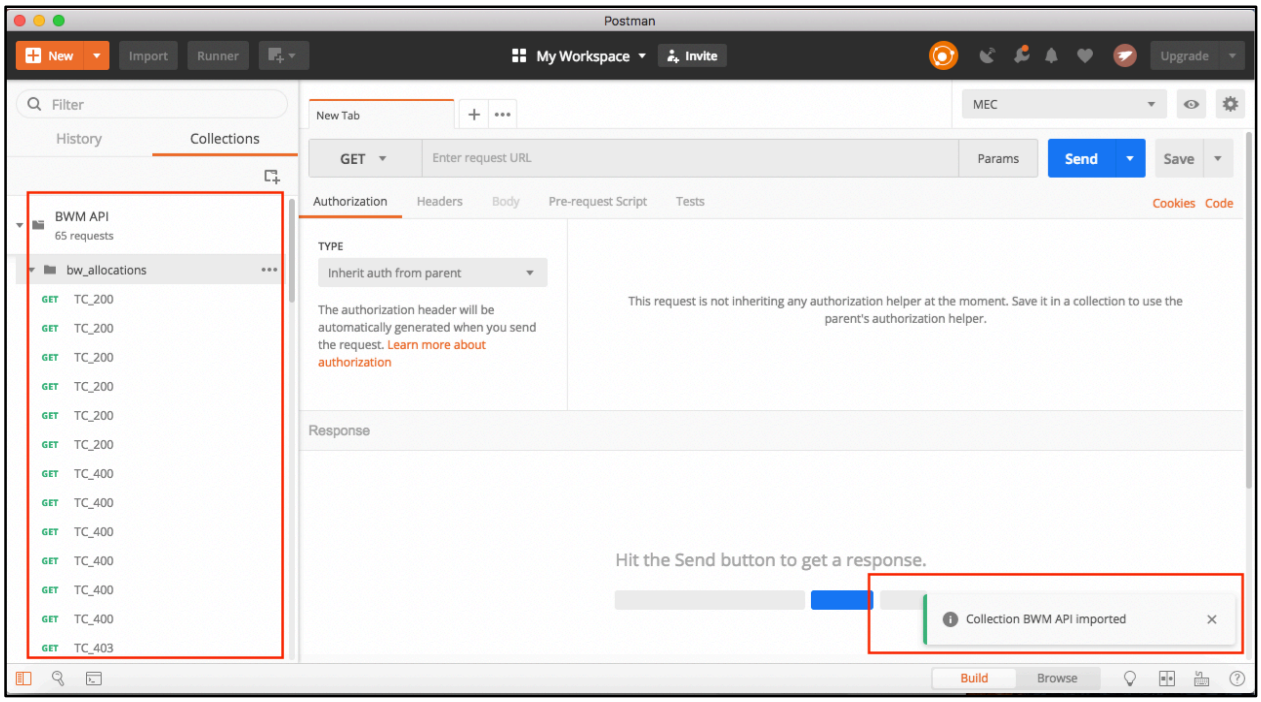
b. The created Postman collection file then can be used by the user without any modifications required to be made to it.

The collection file will then be required to be imported to Postman application.



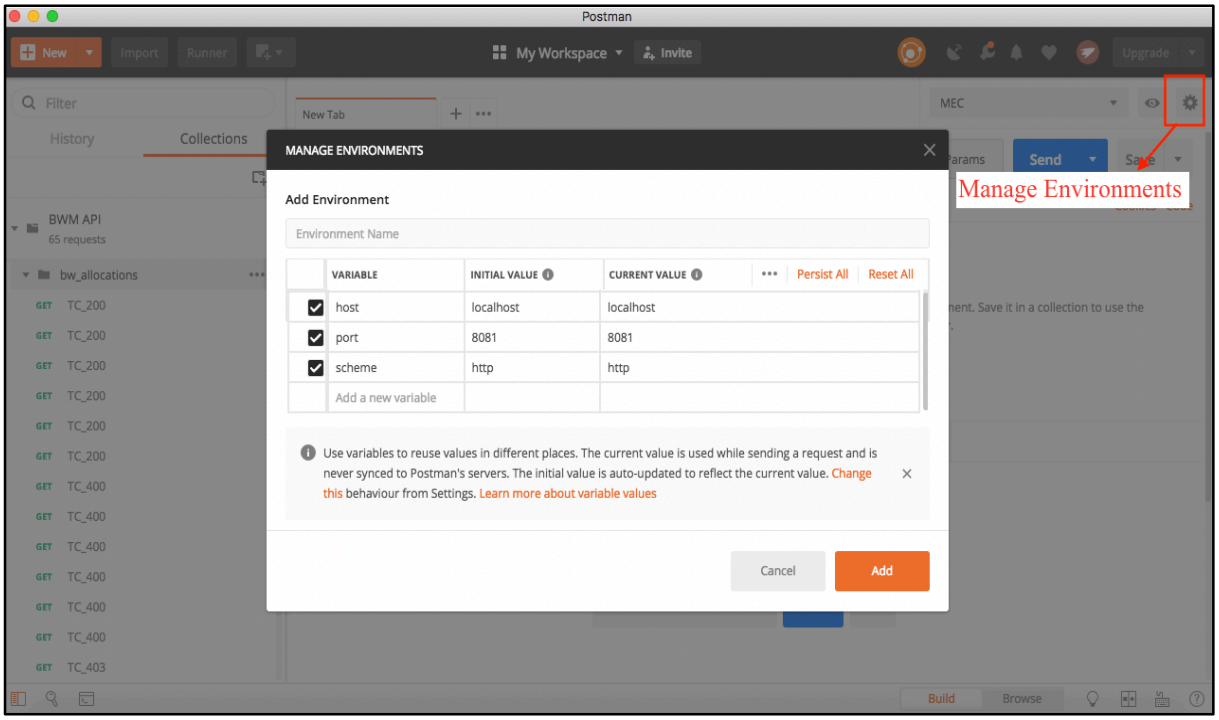
c. After importing the Postman collection JSON file in to the Postman Application then left side section one folder will add automatically and user can see the message as a imported successfully.

On expanding the view folder, the user will be able to view the entire set of test cases which are built based on the input file specifications. The user here will be able to only execute the ready-made test cases and cannot build any test cases from any test specification document.



d. The user will have the option to choose the environment to run the test cases on. On each test cases the user can provide the Environment global variables and the respective values fofr the application environment on the Postman application.

The below image depicts how to set up the Global Param's and values in the Postman Application.



e. 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 the user will be able to start the execution. Postman will use the Mongo database to extract the data to be used for each test case execution. Mongo will be also used to store the data from the test case execution.

f. Postman application will execute the test cases mentioned in serial order. The data for the execution will be taken from the data sheet which will be provided to the user in the deliverable package. The data will be prefilled has a specific order too. The automation solution will be able to extract the data only if the order is not changed.

The user will be able to add to existing set of data as per their requirement and execute the test cases.

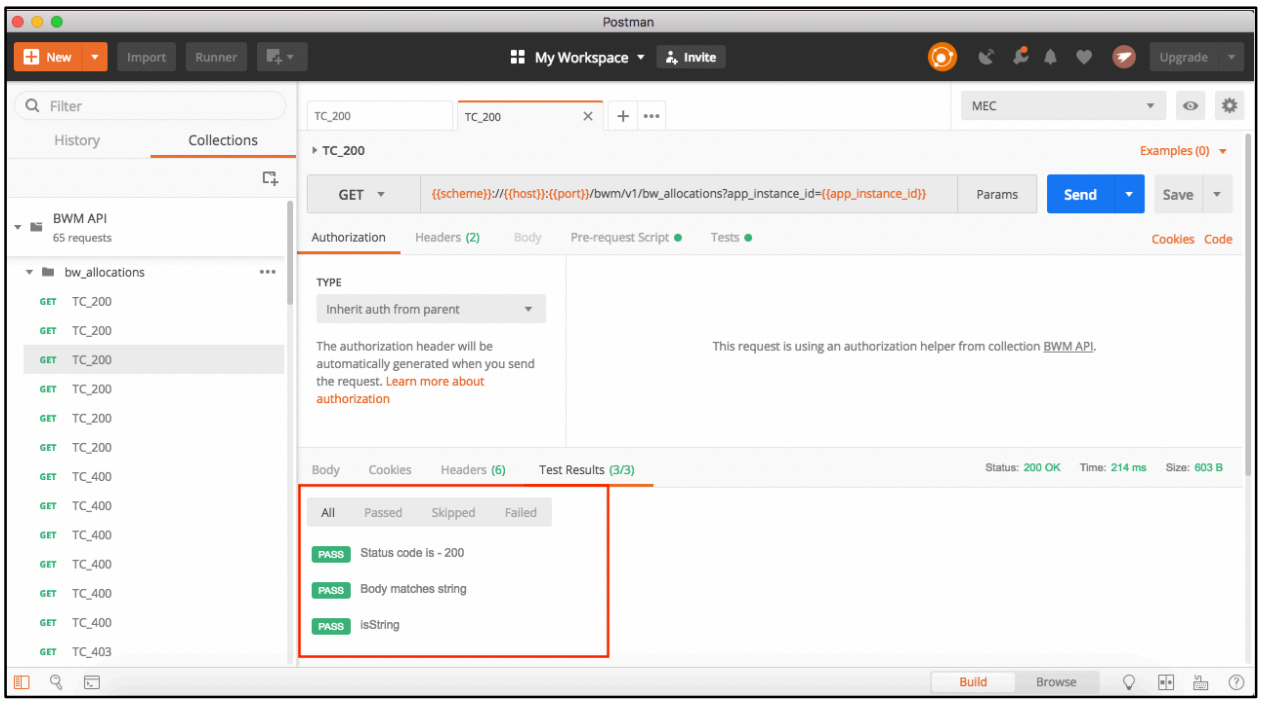
g. The user can execute the test cases in two ways:

1. A single request can be executed by clicking on the “Send” button. The user will be able to view the response in the response section on the Postman application.

2. The user will be able to execute the entire set of test cases by clicking on the “Runner” button and then setting the environment, choosing the collection to execute (in case of multiple collections imported to Postman) and then will be able to view the results in the results section.

h. The user will be able to view the results of the execution as shown below. The execution status will be shown on the Postman application and will also be stored in the Mongo DB for each and every execution.

The picture below shows Test Results in Postman Application.



Please click on the Below YouTube Link for Video: -

https://www.youtube.com/watch?v=3P_LVIdGZNc&feature=youtu.be&list=PLfDSBK5ZoW6SK0wCQnvkZ3hLM9%20ELjeJX5

9. ABOUT US

TestOnNeed is an on-need open source test automation and DevOps solution provider which aims to better the speed, scale, coverage, and quality of software application products. We live only to help businesses 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 (IoT), 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, demo or purchase our solutions, please reach us at mailto: sales@testonned.com.