Table of Contents

[1. Background on Semaphore API 2](#_Toc164283438)

[2. Objective 2](#_Toc164283439)

[3. Test Approach 2](#_Toc164283440)

[4. Test Case Design 3](#_Toc164283441)

[5. Test Case Naming Convention. 3](#_Toc164283442)

[6. Sample Test Case 3](#_Toc164283443)

[7. Test Environment: 5](#_Toc164283444)

[8. Test Execution: 10](#_Toc164283445)

[9. Test Monitoring and Reporting 14](#_Toc164283446)

[10. Performance Tool Integration with CI/CD 22](#_Toc164283447)

[11. Performance Test Tool as JMeter. 23](#_Toc164283448)

**This Document provides the details about how to Test Semaphore API Performance Automation Using JMeter.**

API Performance testing is Performed based on set of documents starting from Test strategy/Test Plan Test Cases, Integration approach.

# Background on Semaphore API

The Semaphore API is locked to apply on the process running in Front End Turbine, there will be concurrent process running in Front End Turbine, to set a lock to the concurrent process in turbine front end using API Calls so Semaphore API is used, this API operation allows users to explicitly lock a resource to prevent concurrent processing. There is API Methods which allow for acquiring and releasing a semaphore.

# Objective

Performance test will evaluate Performance and scalability:

|  |  |  |
| --- | --- | --- |
| SI | Scenarios | How to test |
| 1 | With various Loads for application under test | To Verify the load we need to have different test with changing the no of virtual users/Thread |
| 2 | measure performance of the system | Response time and latency |
| 3 | ensure the system can handle the expected no of users/Transaction. | Assuming that max 100 user will be running the application in parallel |
| 4 | Validate the Bottlenecks | Verify the performcane with respect to System parameter /configurations |

**Note**: 100 users is an assumption. It can be validated with advised no of users/threads

# Test Approach

1. Test cases will be created and updated as new versions whenever there are changes related to semaphore API.
2. Tests will be performed to validate the performance whenever there are changes that relate to semaphore components, this includes changes in semaphore API and semaphore database.
3. Tests will be performed as part of regression test to ensure semaphore is functional.
4. The Performance test for Semaphore API will be Automated and JMeter is recommended as Automation tool.

# Test Case Design

1. In this stage Test cases are Written in Azure test plans for Semaphore API Features.
2. All the test cases are Mapped to Work items in azure test plan.
3. All Test Scenarios identified as of now.
   * GET-Semaphore Scenarios
     1. Fetch the semaphores.
   * POST – Semaphore scenarios.
     1. acquire a lock if no blocking semaphores exist.
     2. reject when acquiring and another exclusive semaphore exist for different process key.
   * PUT-Release – Semaphore scenario to release semaphore.
   * PUT-Renew -Should renew semaphore.

# Test Case Naming Convention.

Below are the naming conventions followed by the QA team in azure test plan.

TCXXX\_Product\_Feature\_Functionality\_TestcaseName

Eg : TC001\_Metadata\_SemaphoreAPI\_AquireLock\_to Validate that lock is acquired when Semaphore is not Blocked.

Testcase Naming convention should follow as-

Testcase Id followed by Product Name, Feature Name, Functionality, Test Case Objective.

-------------------------------------------------

# Sample Test Case

**TC#1**

TC001\_Metadata\_SemaphoreAPI\_AquireLock\_to Validate that lock is acquired when Semaphore is not Blocked.

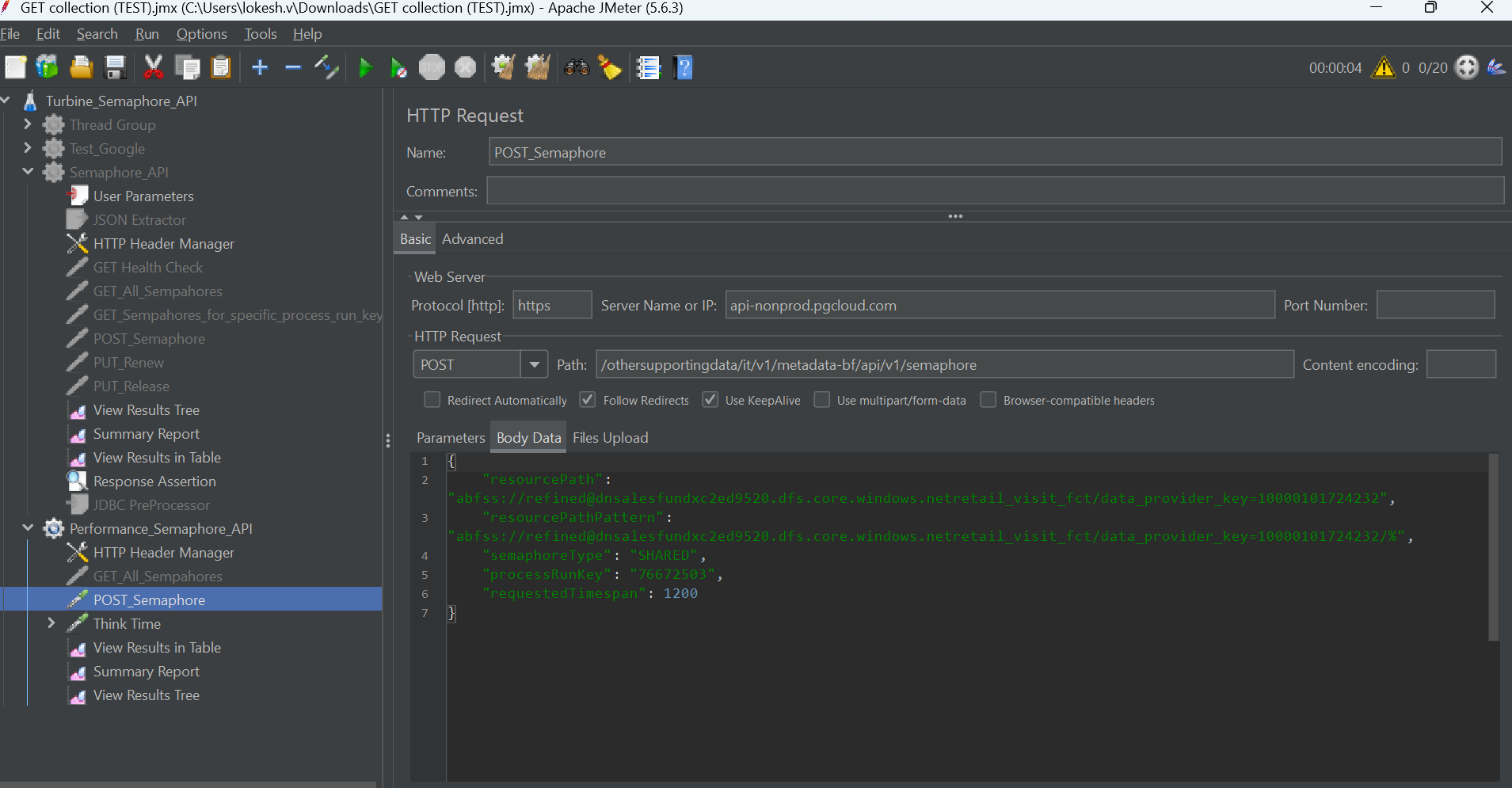
**Objective:** to Validate that lock is acquired when Semaphore is not locked.

**Precondition:**

1- All access, Bearer token and authentication should be available   
2- there should not be any exiting lock on semaphores in the system for the process to acquire lock.

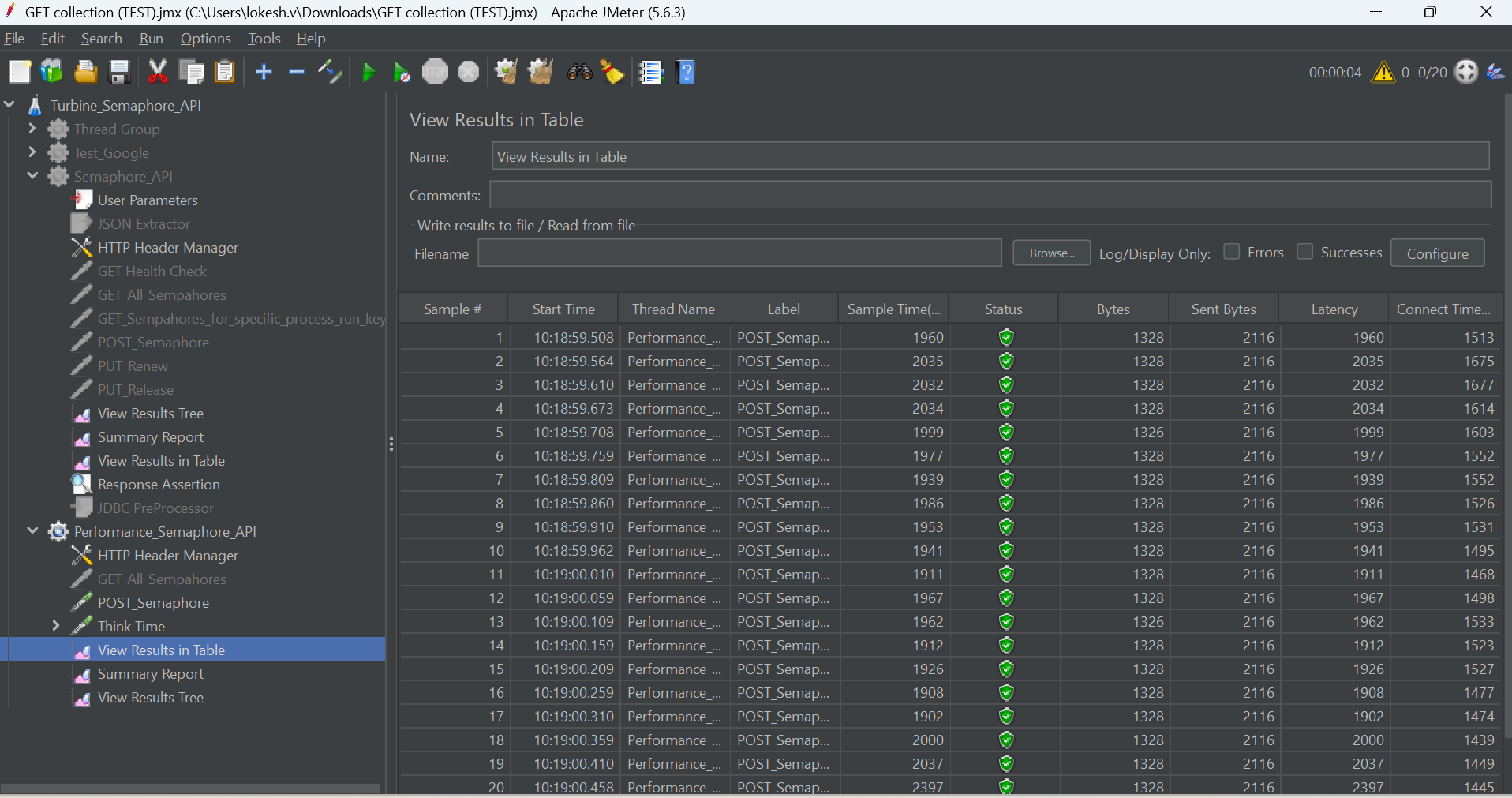
**Steps:**

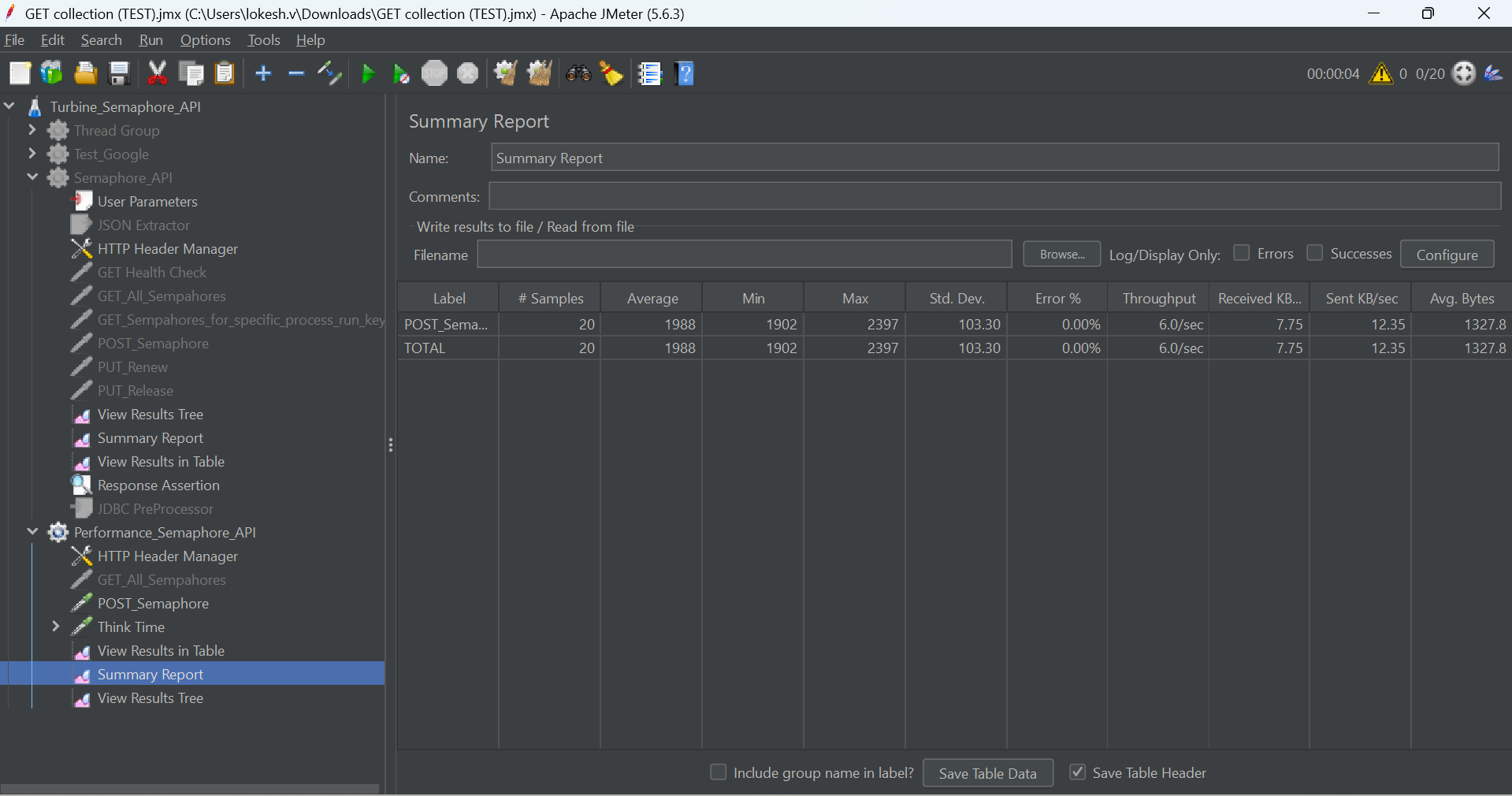
1.With API Post " POST URL" request the lock

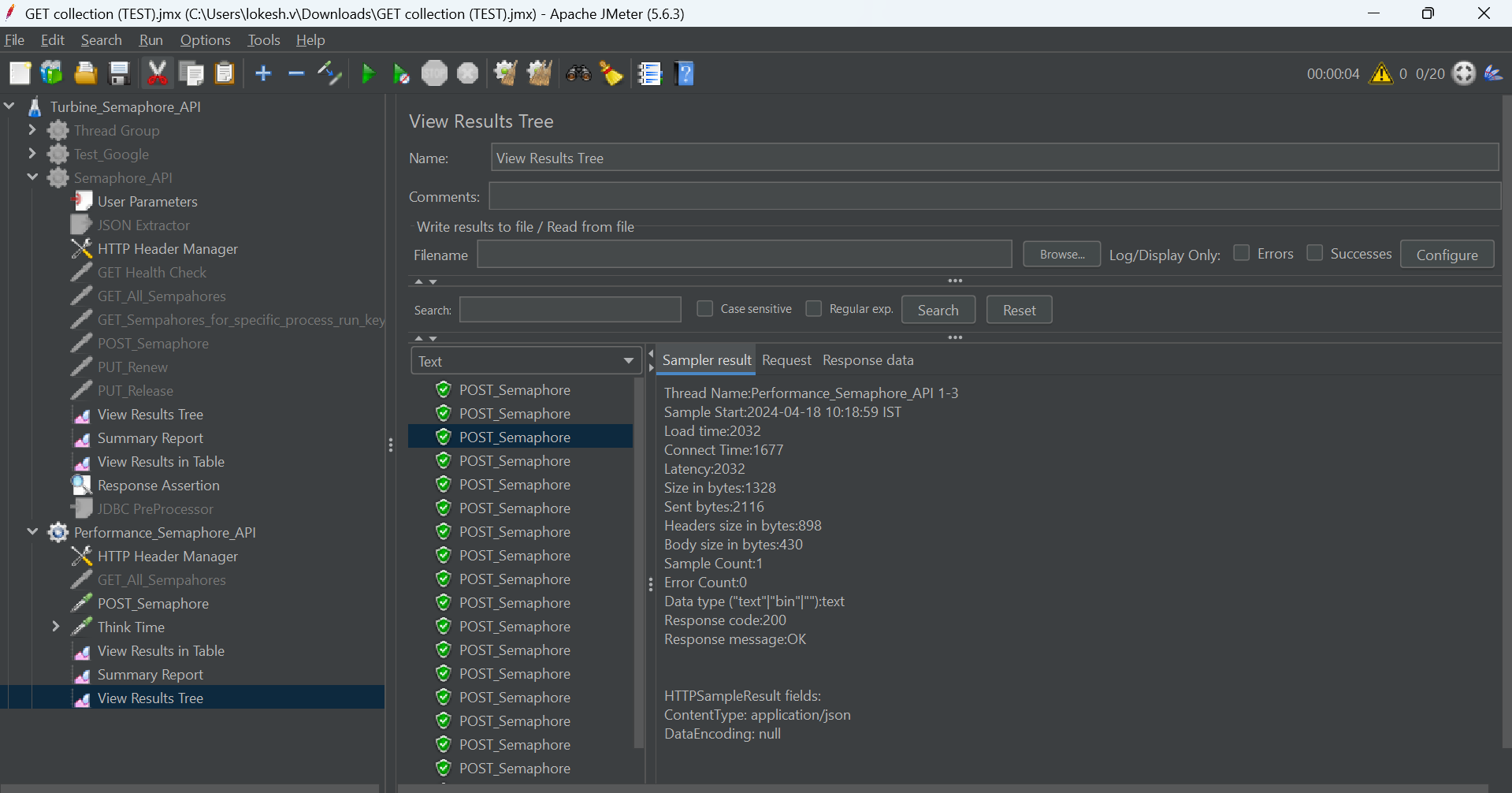
****

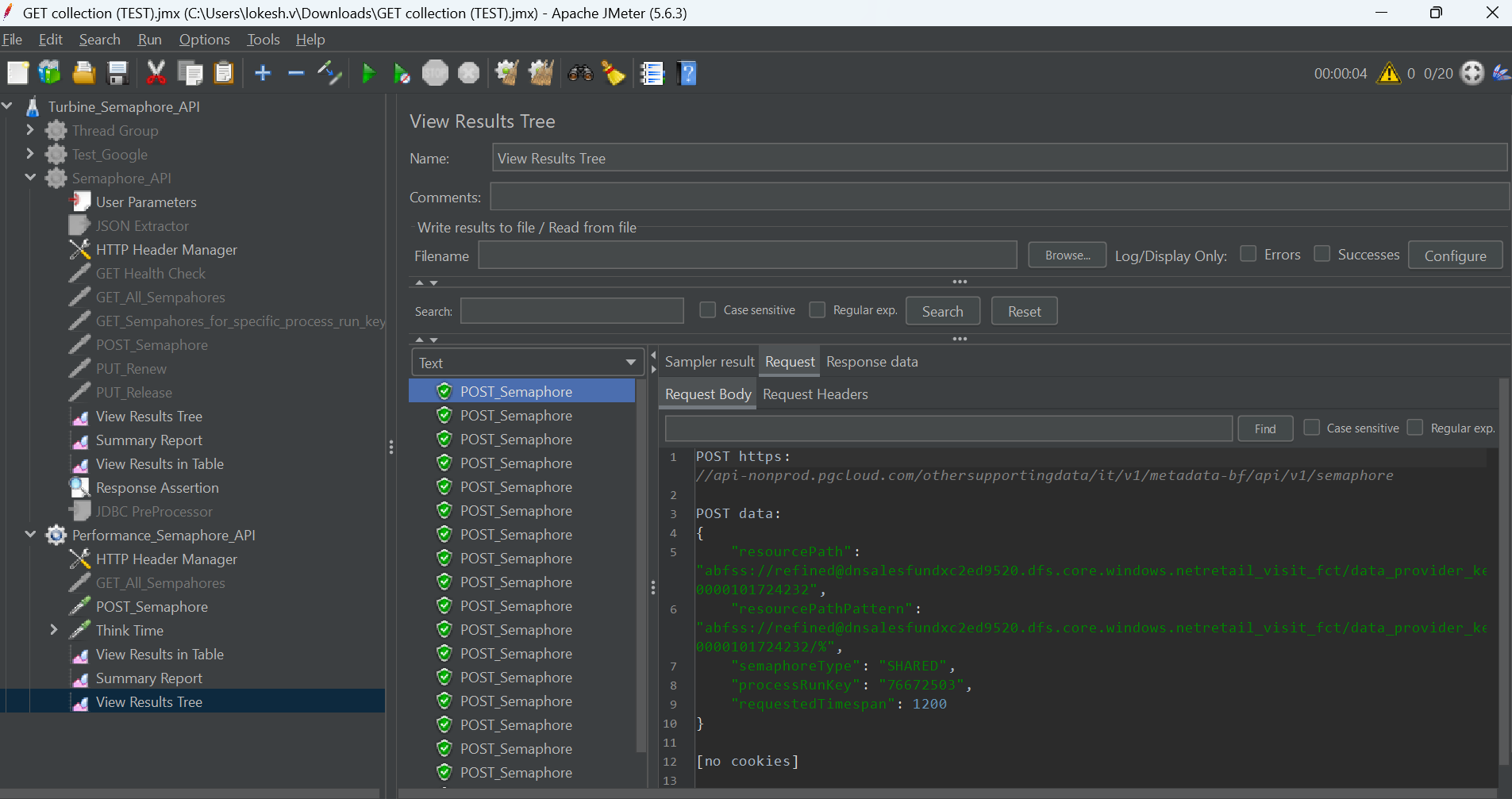
2.Validate the DB table Expected.

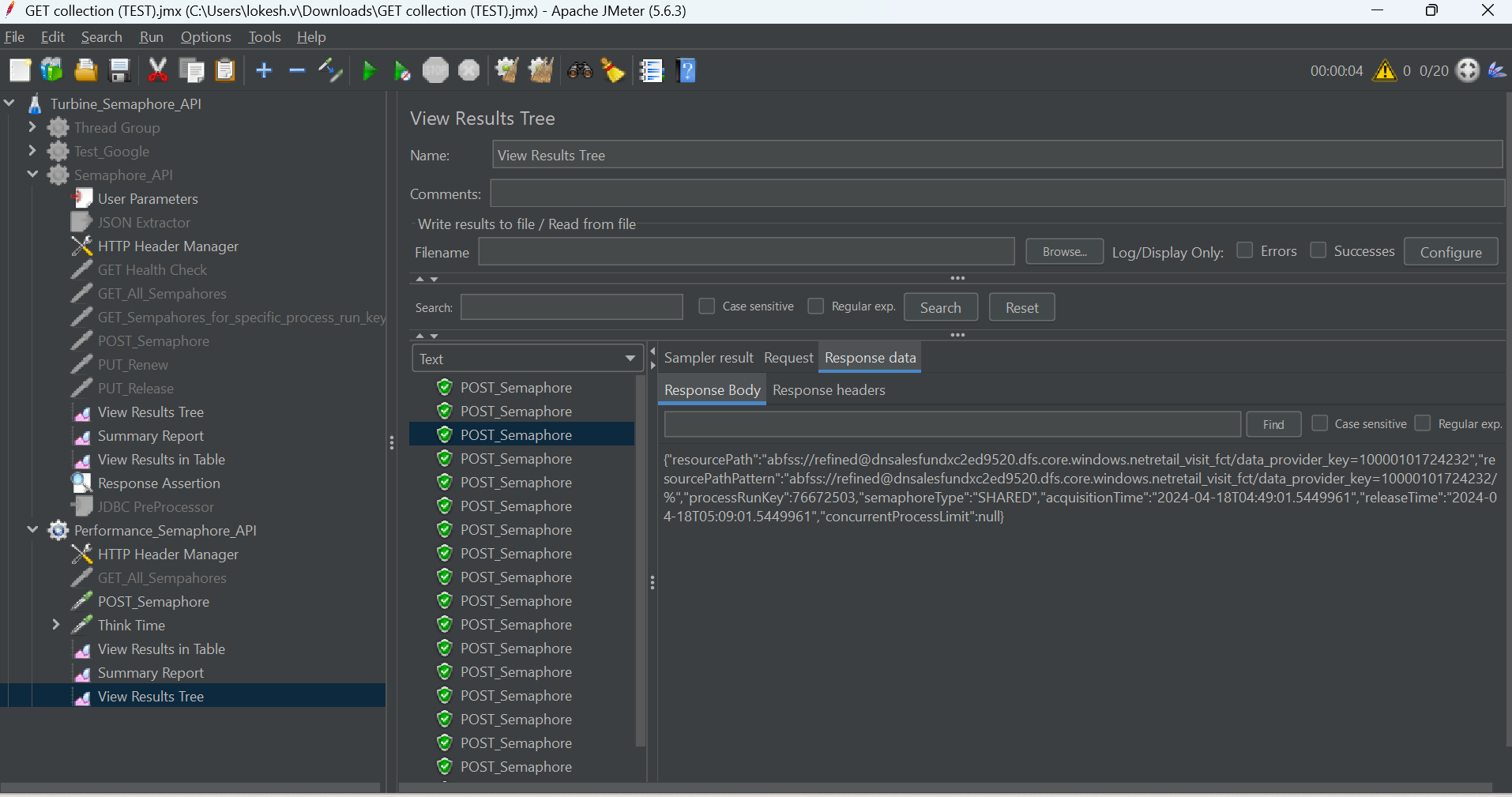
**expected Result:**

****

****

****

****

****

**TC#2**

TC002\_Metadata\_SemaphoreAPI\_RejectLock\_to Validate that lock is Rejected when Semaphore is Blocked.

**Objective:** to Validate that lock is rejected when Semaphore is locked

**Precondition:**

1- All access, Bearer token and authentication should be available   
2- there should not be exiting lock on semaphores in the system for the process to reject lock.

**Steps:**

1.With API Post " POST URL" request the access to acquire lock

2.Validate the DB table Expected.

**expected result:**

A diagram of a computer

Description automatically generated

The above screen shows the test plan structure.

The same structure is maintained in azure test plan.

The above Screenshot shows the Test plan structure maintained in Azure Test Plan.

A screenshot of a computer

Description automatically generated

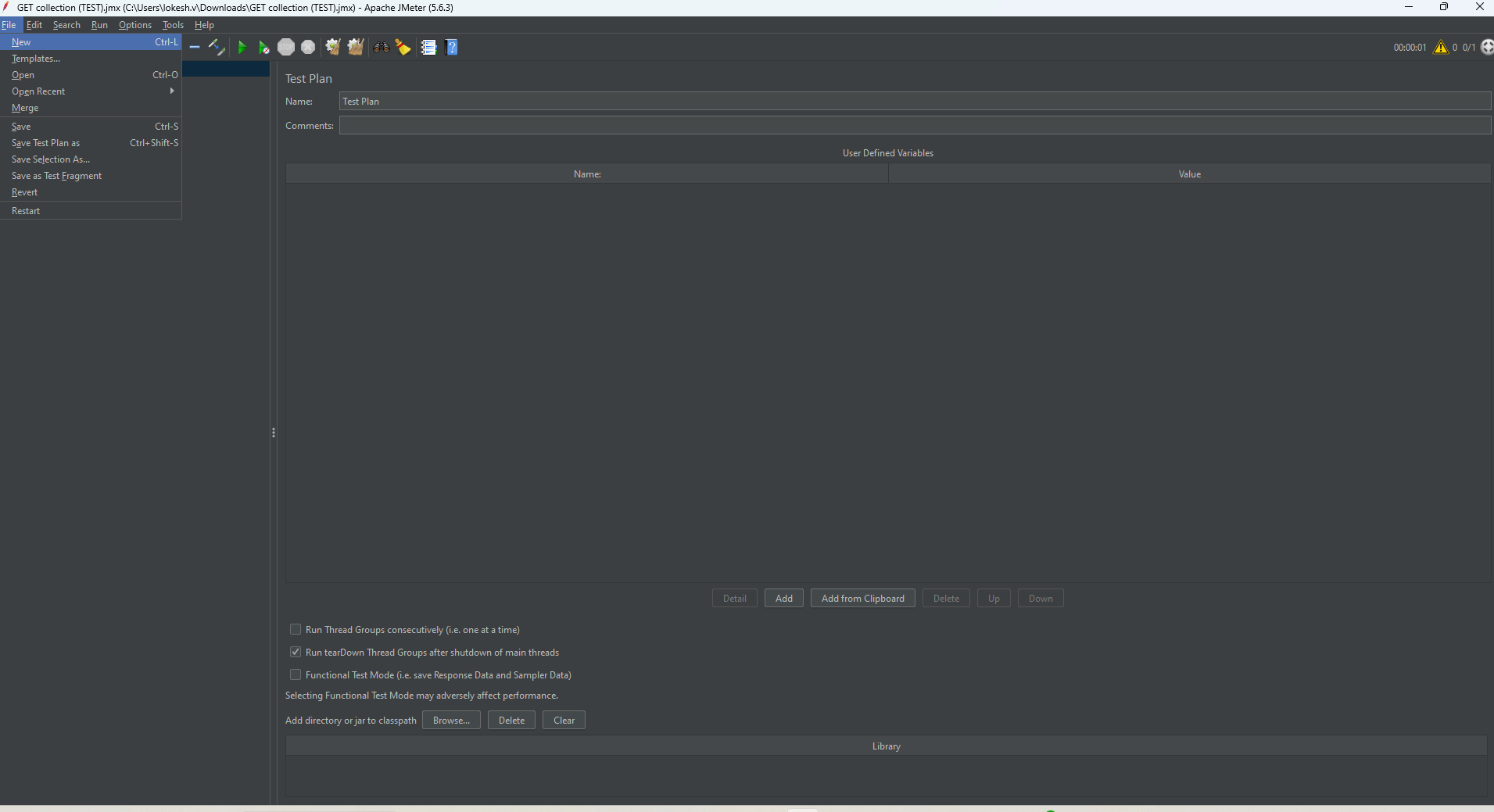
The above screenshot shows the linking of the test plan to the PBI or work item and ensures traceability is maintained.

# Test Environment:

In this stage we will prepare the test environment using the JMeter tool

Below are the screen shots step by step to set up the JMeter tool for Test case execution.

Belo snapshot shows how to add the Test plan, click on File 🡪 Select New

 How to configure JMeter Test Plan for Semaphore API Performance Testing.

1. Go to JMETER\_HOME/bin and start JMeter with jmeterw.cmd on

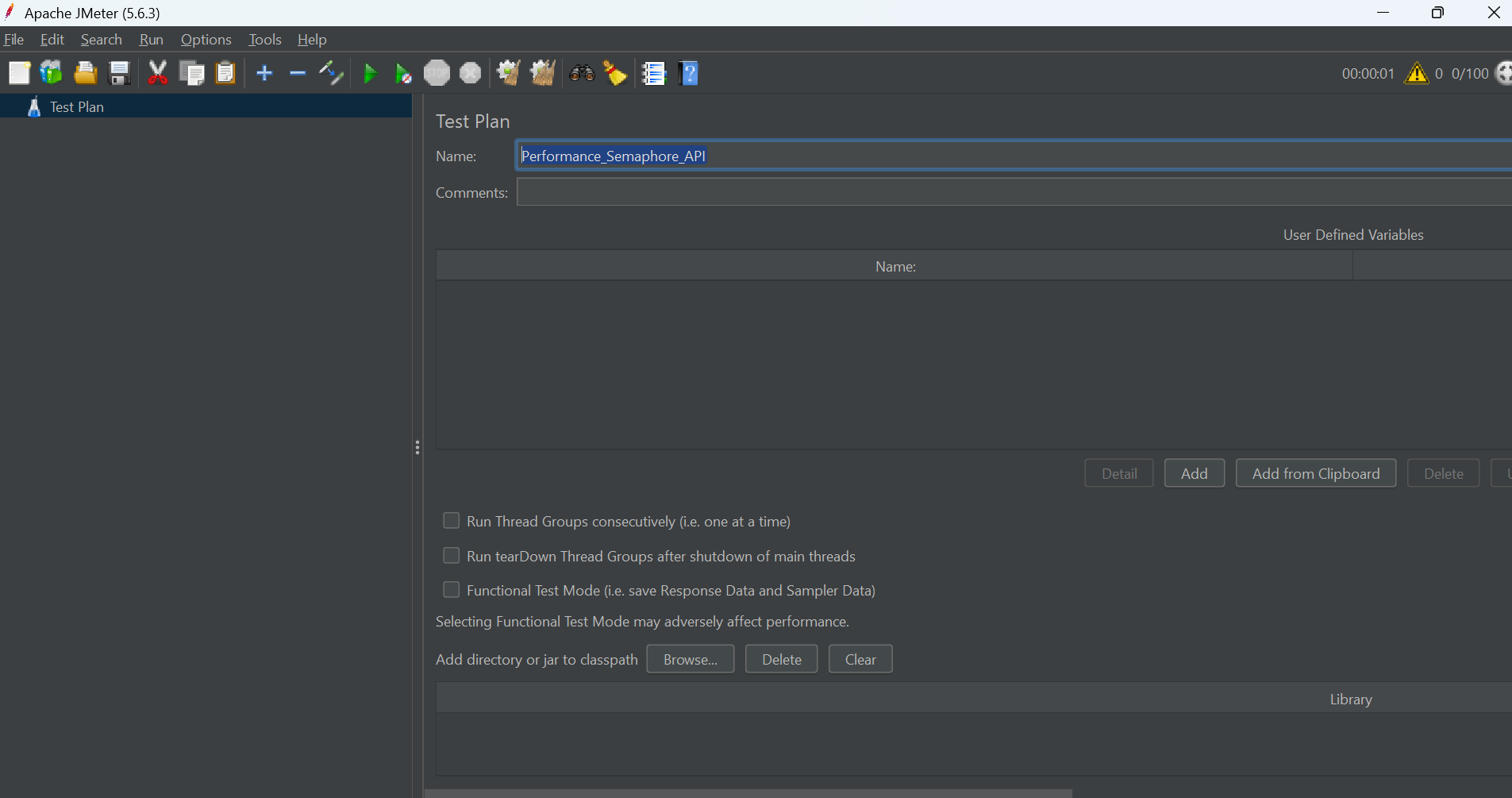
Select File 🡪New 🡪 Test Plan on JMeter

A screenshot of a computer

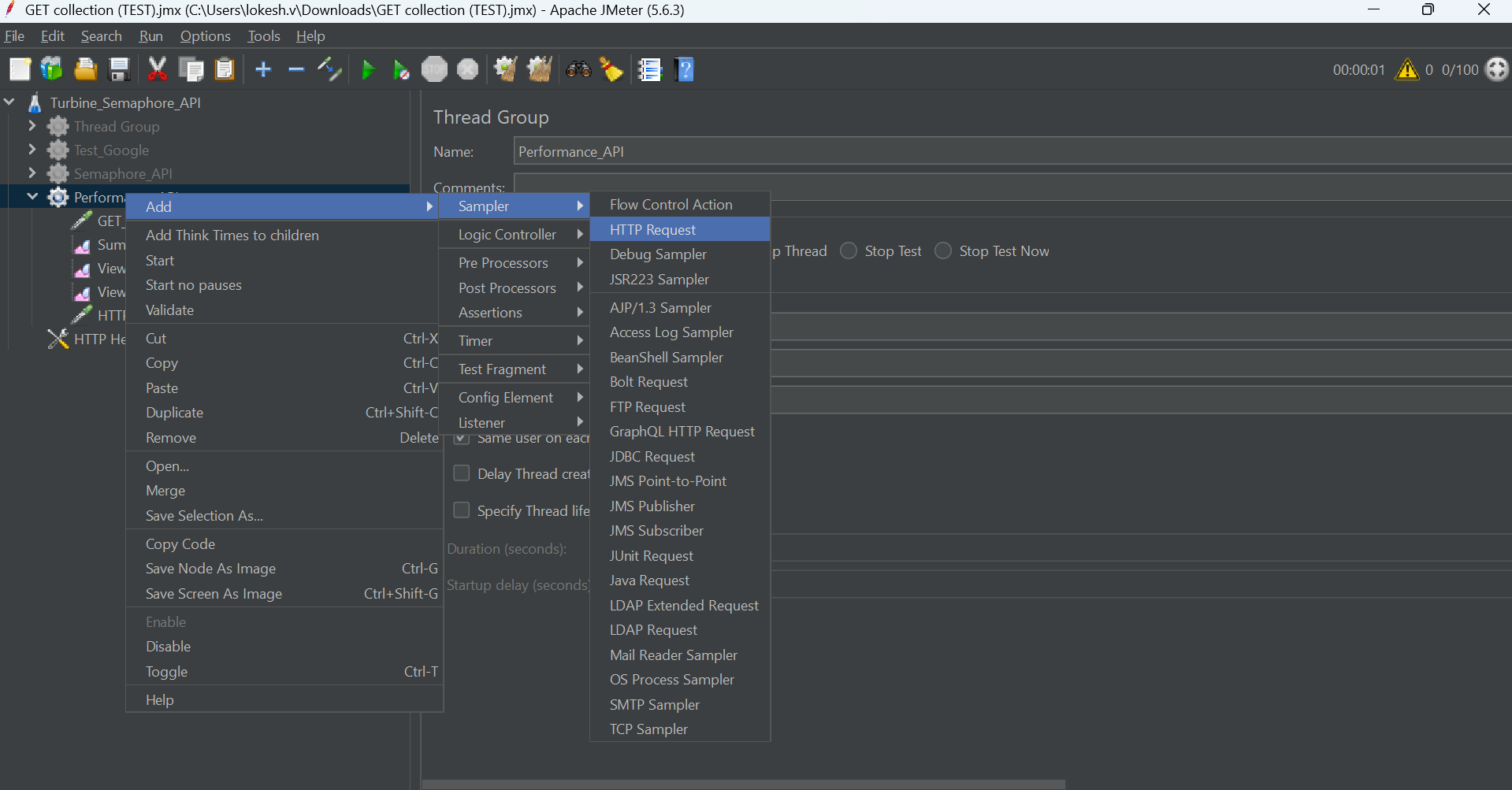
Description automatically generated

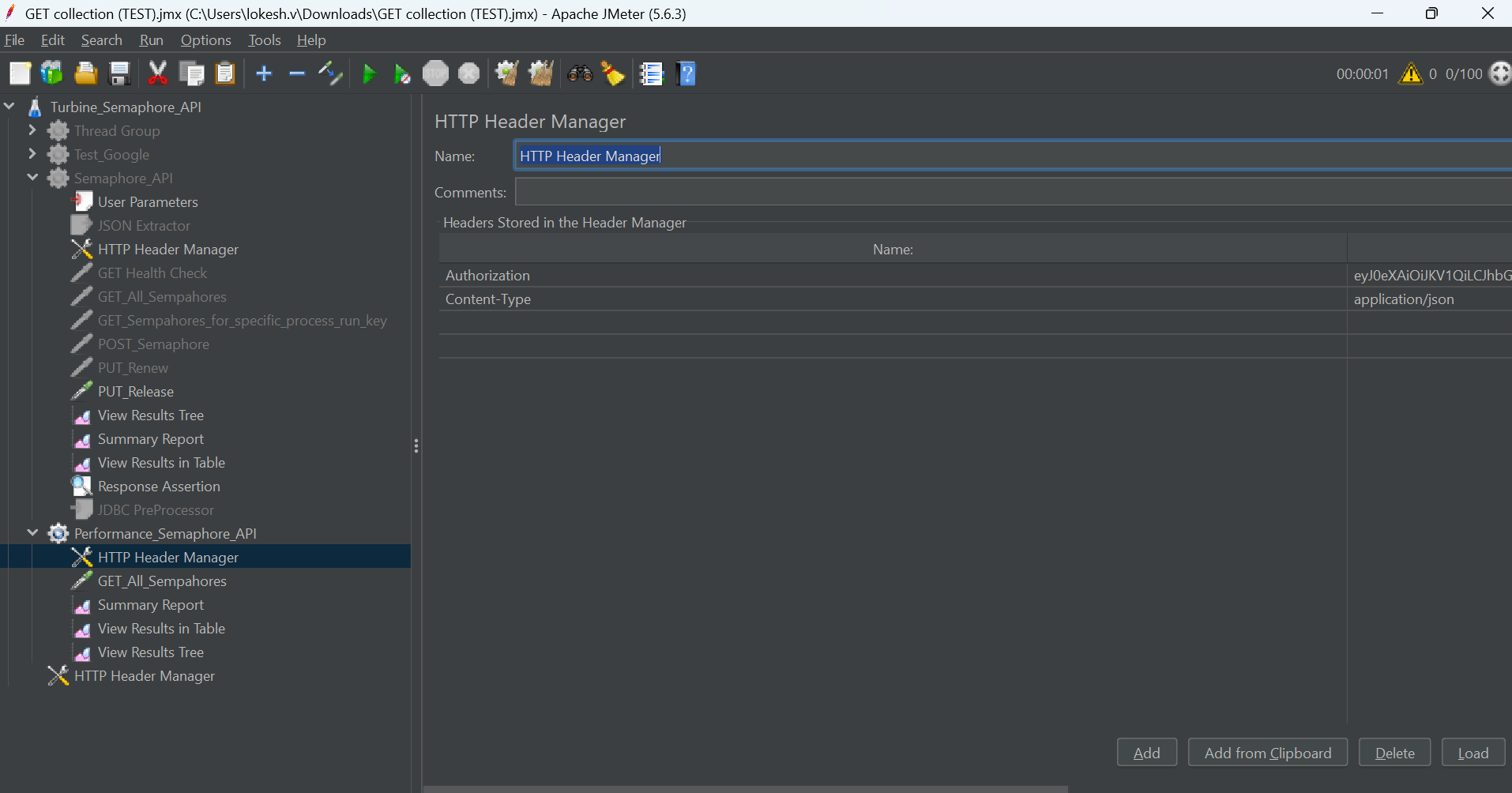
2. Select “Test Plan” on the tree

Rename Test Plan as Performance\_Semaphore\_API as mentioned in the below snapshot

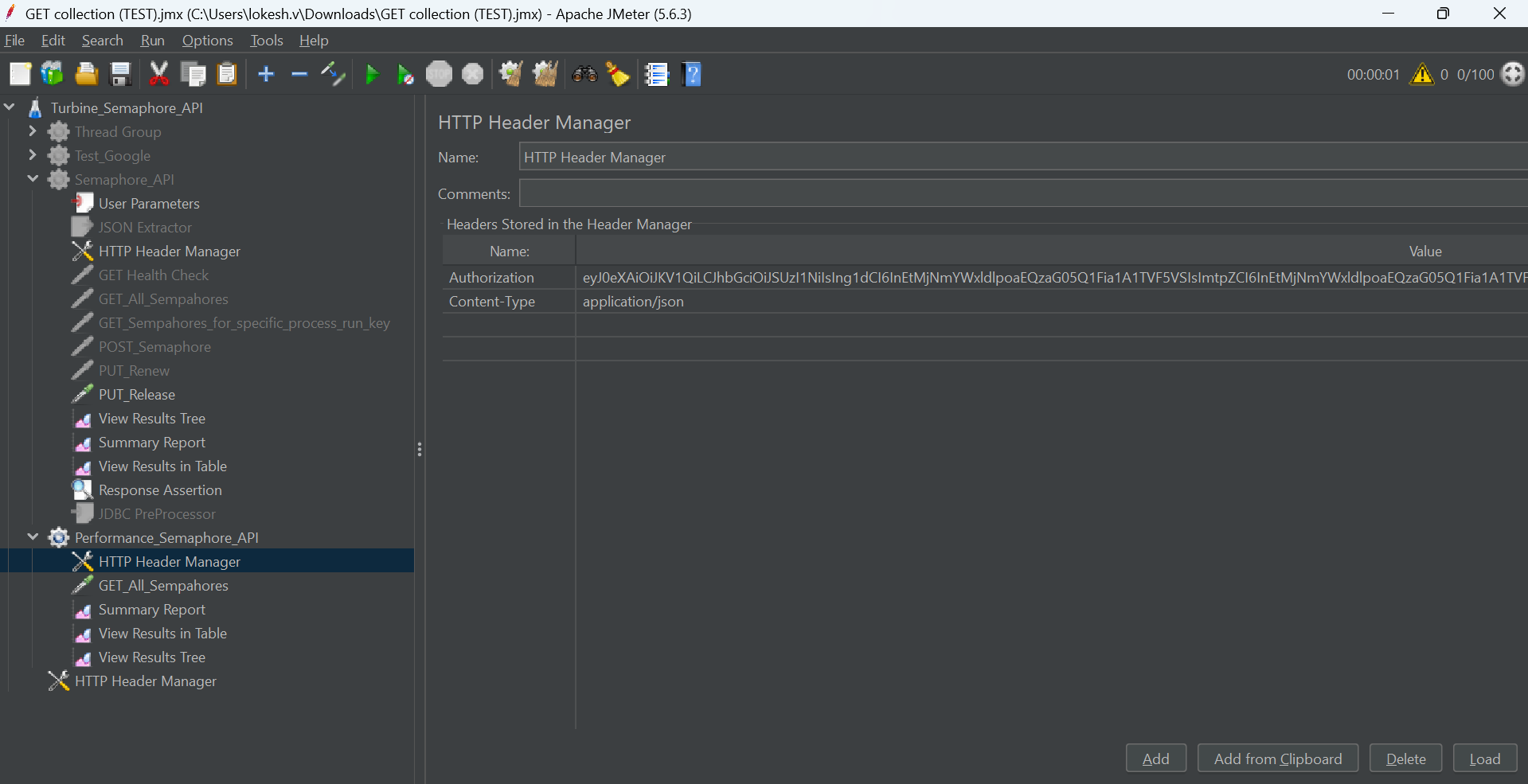


3. Right click on the “Performance\_Semaphore\_API” and add a Sampler 🡪 HTTP Request.

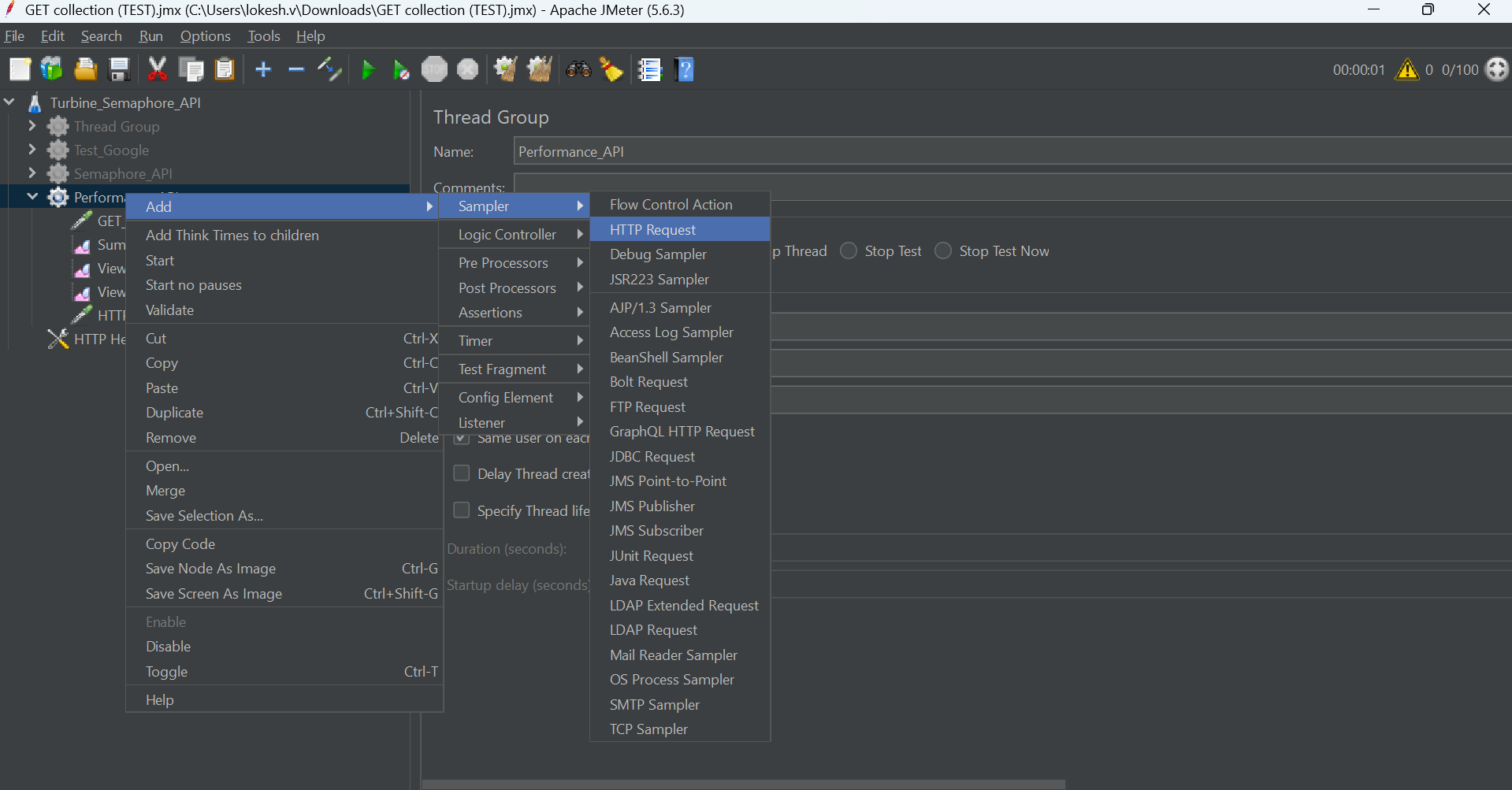
4. Rename as mentioned below snapshot to HTTP Header Manager.



1. Add Authorization and content type along with the values for the same.



1. Right click on the Performance\_Semaphore\_API 🡪 Sample 🡪 HTTP Request

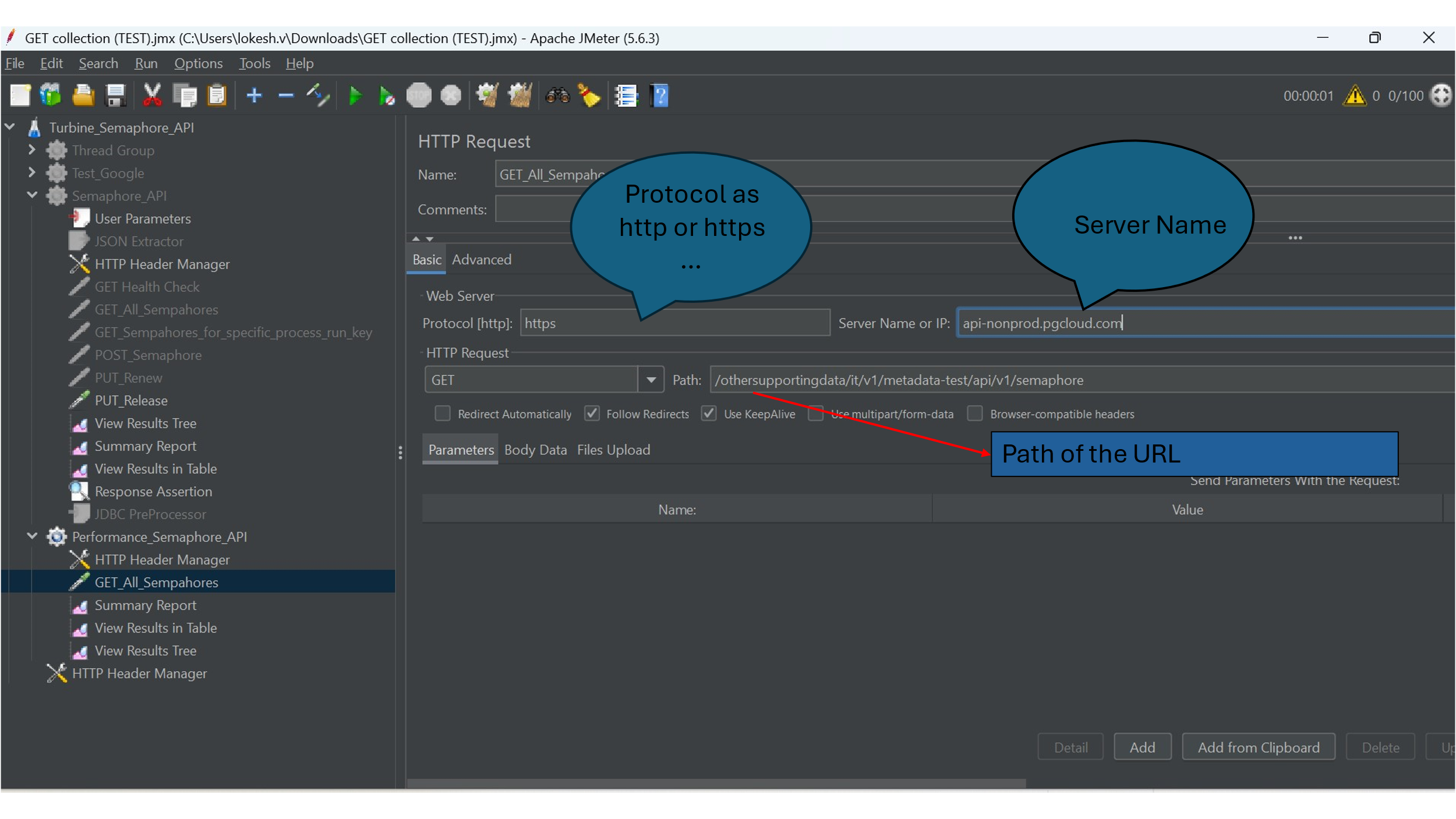


1. Rename as GET\_ALL\_Semaphores as mentioned in below snapshot.

A screenshot of a computer

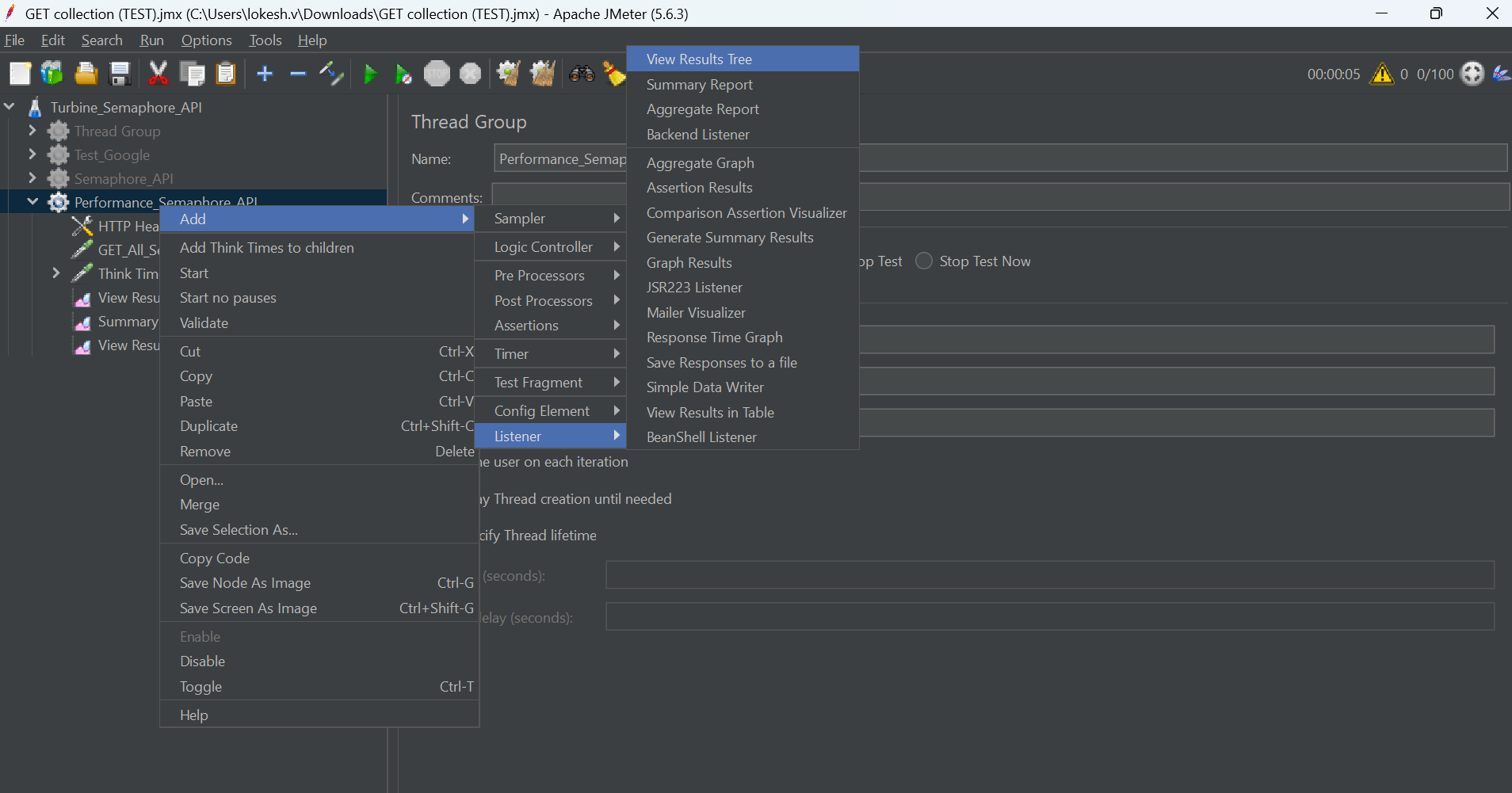
Description automatically generated

1. Make sure you add all the details mentioned in the below snapshot.



1. Right click on the Performance\_Semaphore\_API and ADD🡪 Listener🡪 View Result Tree.

Below snapshot shows how to add listener to capture the results.



1. You can add as different listeners like summary report, View Results in Table which we used for capturing the results in our examples.

# Test Execution:

In This Stage We run the test case and Get the Performance metrics for the simple GET all Semaphores for 20 virtual users.

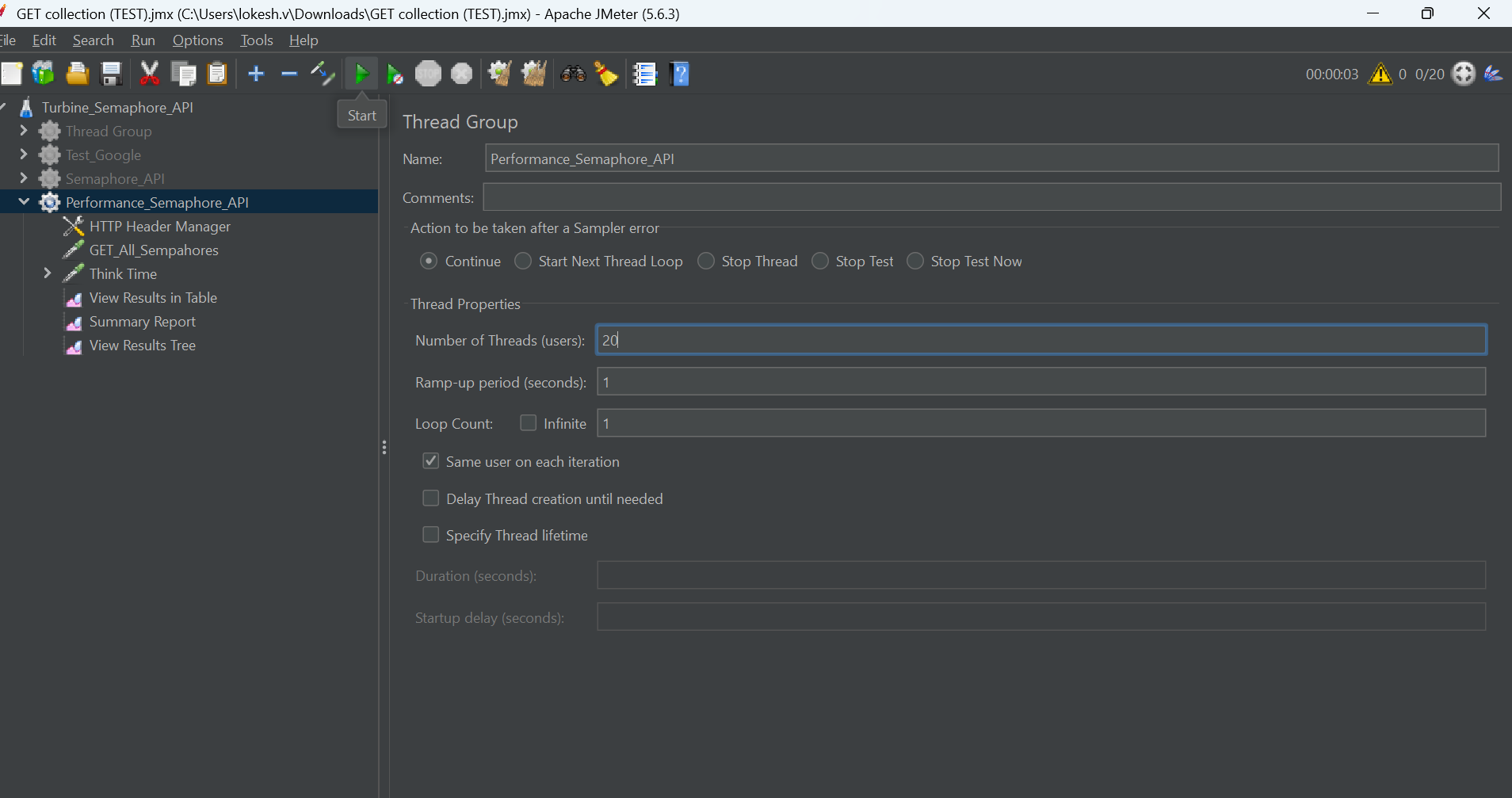
Here are a few examples of semaphore API methods used where in client sends the request and server response to measure the performance for 20 Users.

1. GET Request for all semaphores.

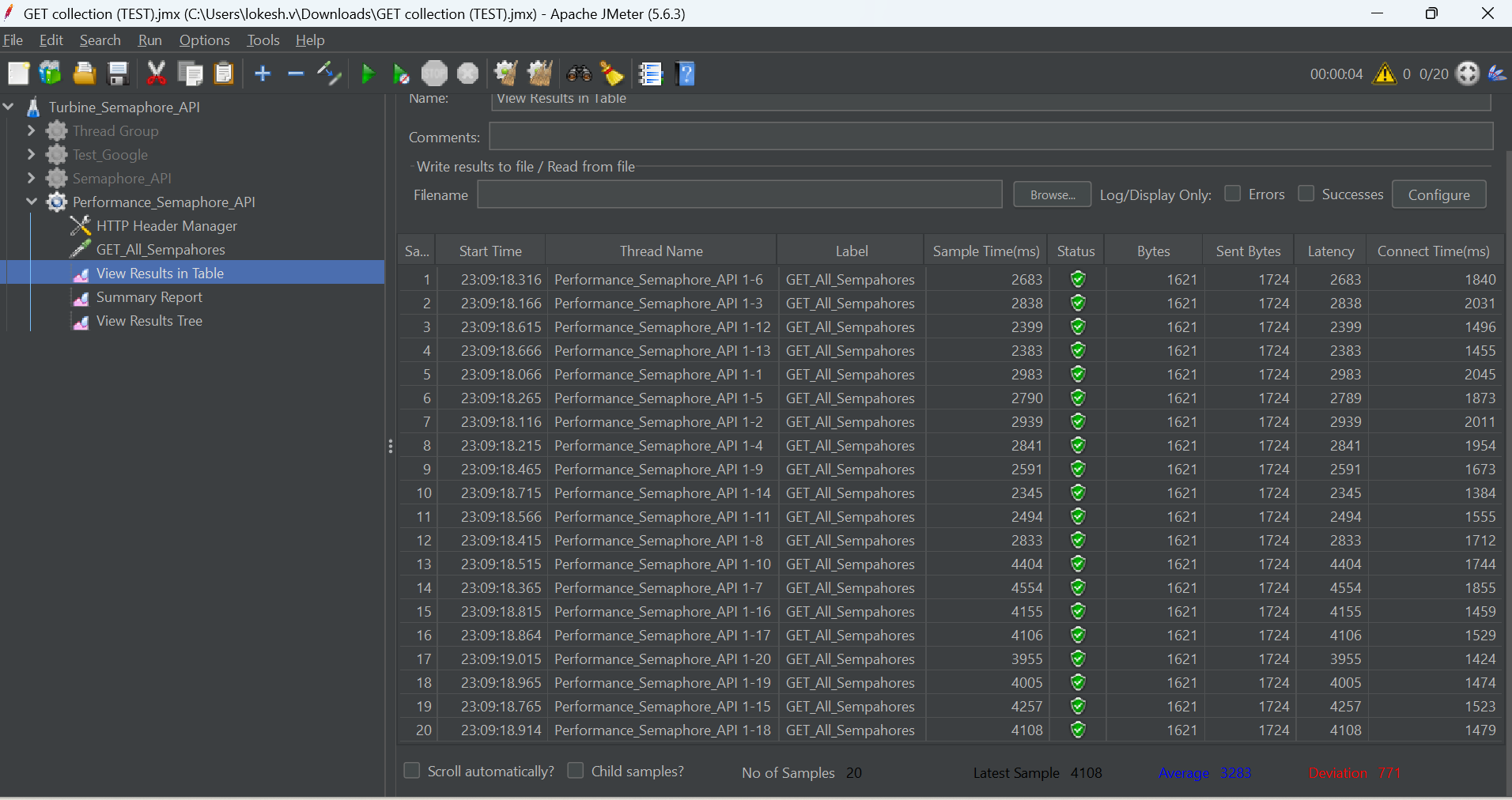
A screenshot of a computer

Description automatically generated

After configuring all the parameters and necessary listeners we need to click on the start/run button as shown in below snapshot.



After running the API for 20 users, click on the **View Results in Table**, below is the example for 20 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.



Below are the snapshot results of the **Summary Report** and different Performance metrics captured as summary Report for **20 Virtual Users**.

A screenshot of a computer

Description automatically generated

Below is the Performance metrics results in **View Results Tree** format which captured for individual performance metrics for **20 Virtual Users**.

A screenshot of a computer

Description automatically generated

Below is the Performance Request body sent for each individual Virtual user.

A screenshot of a computer

Description automatically generated

Below snapshot is the Response body displayed for each individual performance measured as part of performance metrics.

A screenshot of a computer

Description automatically generated

# Test Monitoring and Reporting

This is the final stage where we monitor the results and report the test results to the stakeholders through email. We generate the detailed reports from the JMeter Results for different Virtual Users in below examples we have covered incrementing of 20 users from each cycle and capture the results from 20 Virtual Users to 100 users and for each increment we capture the results as mentioned below which include benchmarks for each increment.

A screenshot of a computer

Description automatically generated

After running the **API for 40 users**, click on the **View Results in Table**, below is the example for 40 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

Below are the snapshot results of the Summary Report, and different Performance metrics captured as summary Report for **40 Users**.

A screenshot of a computer

Description automatically generated

Below is the Performance metrics results in **View Results Tree** format which captured individual sample result performance metrics for **40 Virtual Users**.

A screenshot of a computer

Description automatically generated

Below is the Request Body sent for each individual Virtual User.

A screenshot of a computer

Description automatically generated

Below snapshot shows the Response Body for each individual Virtual User.

A screenshot of a computer

Description automatically generated

After running the API for 60 users, click on the View Results in Table, below is the example for 60 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

After running the **API for 60 users**, click on the **View Results in Table**, below is the example for 60 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

Below are the snapshot results of the **Summary Report** and different Performance metrics captured as summary Report for **60 Virtual Users**.

A screenshot of a computer

Description automatically generated

After running the API for 80 users, click on the View Results in Table, below is the example for 80 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

Below are the snapshot results of the **Summary Report** and different Performance metrics captured as summary Report for **80 Virtual Users**.

A screenshot of a computer

Description automatically generated

After running the **API for 80 users**, click on the **View Results in Table**, below is the example for 80 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

After running the API for 100 users, click on the View Results in Table, below is the example for 100 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

After running the **API for 100 users**, click on the **View Results in Table**, below is the example for 100 users, different metrics captured during the test execution, we can see the different values performance metrics in the snapshot.

A screenshot of a computer

Description automatically generated

Below are the snapshot results of the **Summary Report** and different Performance metrics captured as summary Report for **100 Virtual Users**.

A screenshot of a computer

Description automatically generated

The Above Screenshot shows the Execution results for monitoring and reporting how it looks in JMeter to be considered as example only.

# Performance Tool Integration with CI/CD

JMeter has documented the integration with Jenkins here Jenkins is used for example purposes only , we can integrate with any CI/CD tool. We just need to enable the plugin and follow the steps to achieve the integration. [Using JMeter with Jenkins](https://www.jenkins.io/doc/book/using/using-jmeter-with-jenkins/)



# Performance Test Tool as JMeter.

How to use the tool to support testing please follow the below link

<https://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html>