Table of Contents

[1. Background on Semaphore API 2](#_Toc162991551)

[2. Test Approach 2](#_Toc162991552)

[3. Test Case Design 2](#_Toc162991553)

[4. Test Case Naming Convention. 3](#_Toc162991554)

[5. Test Environment: 5](#_Toc162991555)

[6. Test Execution: 7](#_Toc162991556)

[7. Test Monitoring and Reporting 9](#_Toc162991557)

[8. Test Tool 10](#_Toc162991558)

**This Document provides the details about how to Test Semaphore API.**

API testing is Performed based on set of documents starting from Test Cases, Test plan/Test strategy, Traceability matrix,

# 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 are API Methods which allow for acquiring and releasing a semaphore.

# Test Approach

1. Test cases will be created and refreshed whenever there are changes related to semaphore API.
2. Tests will be performed whenever there are changes that relates 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 Semaphore API will be Manual Tested using Postman and for automation testing will be using JMeter 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 devops.

**All Test Scenarios identified as of now.**

Fetch -Semaphore Scenarios

1. fetch semaphores.

Release – Semaphore scenarios.

1. should release semaphore.

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.

# 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.

**Top 2 - Test Cases:**

**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:**

1.the lock should be acquired successfully.

2.the system should indicate that the lock has been acquired.

**Actual 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:**

1.the lock should be rejected.

2.the system should indicate that the lock has been rejected.

**Actual 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.

A screen shot of a computer

Description automatically generated

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

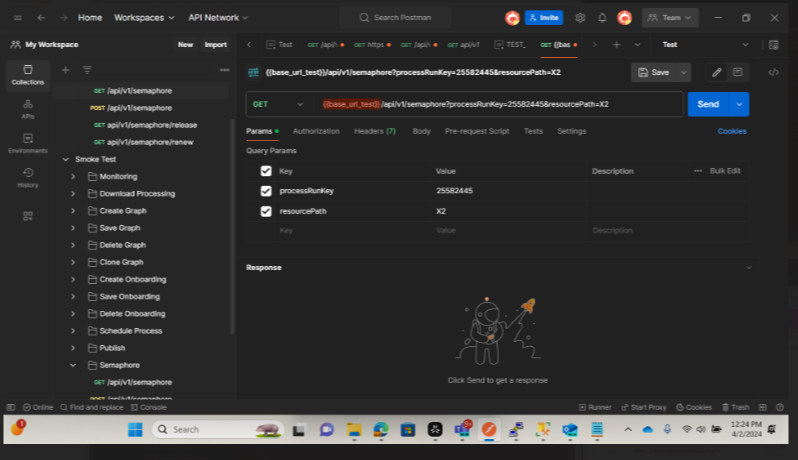
A screenshot of a computer

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 postman tool and configuring the Semaphore API, Data base to refer the row in a table, turbine UI to trigger the graphs.



The above screenshot shows the postman request for the process run key.

URL of the post man:  [My Workspace (postman.co)](https://pg-api.postman.co/workspace/My-Workspace~8748e596-dc7d-4991-8b6c-ce8f4d52e439/request/33684371-dbe3c538-2dbf-40ad-904b-773a6b953c5c?tab=params)

A screenshot of a computer

Description automatically generated

The above screen shot shows the process run key present in the Semaphore DB.

A screenshot of a computer

Description automatically generated

The above screen shot shows the process run key present in Metadata DB.

A screenshot of a computer

Description automatically generated

The Above Screenshot shows the process key present in the Turbine UI.

# Test Execution:

In This Stage We run the test case and validate the API test results for expected against the Actual results.

Here are a few examples of semaphore API methods used where in client send the request and server respond.

1. GET Request

A screenshot of a computer

Description automatically generated

1. The above screenshot shows the postman GET request for the process run key.
2. POST Request

Scenario 1: To acquire a lock if no blocking semaphores exist.

A screenshot of a computer

Description automatically generated

The Above Screenshot shows to trigger the POST request in swagger.

Scenario 2: To reject when acquiring and another exclusive semaphore exist for different process key.

A screenshot of a computer

Description automatically generated

The Above Screenshot shows the POST request to acquire the Lock.

A screenshot of a computer

Description automatically generated

The Above Screenshot shows the Swagger URL for POST request to acquire the lock for the request body.

# 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 collections in the post man including any issues/findings.

A screen shot of a computer

Description automatically generated

The Above Screenshot shows the Execution results for monitoring and reporting how it looks in Postman for to be consider as example only for pass scenarios.

A screenshot of a computer

Description automatically generated

The Above Screenshot shows the Execution results for monitoring and reporting how it looks in Postman for to be consider as example only for pass and failed scenarios.

How to generate reports in postman for **enterprise** edition

<https://learning.postman.com/docs/reports/reports-overview/>

# Test Tool

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

[Postman Learning Center](https://learning.postman.com/docs/designing-and-developing-your-api/testing-an-api/)