Feature: Semaphore Locking

Scenario: Acquire Lock When No Blocking Semaphores Exist

Given there are no blocking semaphores in the system

When a lock is requested

Then the lock should be acquired successfully

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

This scenario describes the behavior of acquiring a lock when no blocking semaphores exist in the system. You can implement the step definitions corresponding to this scenario in your test automation framework.

Feature: Semaphore Locking

Scenario: Reject Acquiring Lock When Another Exclusive Semaphore Exists for Different Process Key

Given there is an exclusive semaphore held by a different process key

When a lock is requested for a different process key

Then the lock acquisition should be rejected

And the system should indicate that the lock acquisition has failed due to an existing exclusive semaphore

This scenario describes the behavior of rejecting the acquisition of a lock when another exclusive semaphore is held by a different process key. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Allow Multiple Shared Semaphores When Limit is Not Exceeded

Given the system allows for multiple shared semaphores

And the current number of shared semaphores does not exceed the limit

When multiple shared semaphores are requested

Then the system should allow the acquisition of each semaphore

And indicate that each semaphore has been acquired successfully

This scenario describes the behavior of allowing the acquisition of multiple shared semaphores when the limit is not exceeded. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Reject Acquiring Shared Semaphore When Concurrent Process Limit Reached

Given the system has reached the concurrent process limit for shared semaphores

When a shared semaphore is requested

Then the system should reject the acquisition of the semaphore

And indicate that the acquisition has failed due to reaching the concurrent process limit

This scenario describes the behavior of rejecting the acquisition of a shared semaphore when the concurrent process limit for shared semaphores has been reached. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Reject Acquiring Shared Semaphore When Limit is Set Below Current Semaphore Count and Total Processes Exceed It

Given the system has more shared semaphores currently acquired than the new limit

And the total number of processes exceeds the new limit

When a shared semaphore is requested

Then the system should reject the acquisition of the semaphore

And indicate that the acquisition has failed due to exceeding the new limit

This scenario describes the behavior of rejecting the acquisition of a shared semaphore when the limit is set below the current number of semaphores and the total number of processes exceeds the new limit. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Allow Downsizing Concurrent Process Limit When Total Number of Processes Doesn't Exceed It

Given the system has a current concurrent process limit

And the total number of processes does not exceed the current limit

When the concurrent process limit is downsized

Then the system should allow the limit to be changed successfully

And indicate that the new limit has been applied

This scenario describes the behavior of allowing the downsizing of the concurrent process limit when the total number of processes doesn't exceed it. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Allow Multiple Shared Semaphores Without Limit

Given there is no limit set for the number of shared semaphores

When multiple shared semaphores are requested

Then the system should allow the acquisition of each semaphore

And indicate that each semaphore has been acquired successfully

This scenario describes the behavior of allowing multiple shared semaphores without any limit. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Reject Exclusive Semaphore When Shared Semaphore Exists and Process Run Keys Are Different

Given there is a shared semaphore held by a different process run key

And the process run keys for the shared semaphore and the requested exclusive semaphore are different

When an exclusive semaphore is requested

Then the system should reject the acquisition of the exclusive semaphore

And indicate that the acquisition has failed due to the presence of a shared semaphore with a different process run key

This scenario describes the behavior of rejecting the acquisition of an exclusive semaphore when a shared semaphore exists and the process run keys are different. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Acquire Exclusive Semaphore When Shared Semaphore Exists and Process Run Keys Are the Same

Given there is a shared semaphore held by the same process run key

And the process run keys for the shared semaphore and the requested exclusive semaphore are the same

When an exclusive semaphore is requested

Then the system should allow the acquisition of the exclusive semaphore

And indicate that the exclusive semaphore has been acquired successfully

This scenario describes the behavior of acquiring an exclusive semaphore when a shared semaphore exists and the process run keys are the same. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Locking

Scenario: Acquire Multiple Locks for Exclusive Semaphore for the Same Process Key

Given there is an exclusive semaphore held by a process key

When multiple locks are requested for the exclusive semaphore with the same process key

Then the system should allow the acquisition of each lock

And indicate that each lock for the exclusive semaphore has been acquired successfully

This scenario describes the behavior of acquiring multiple locks for an exclusive semaphore for the same process key. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Cleanup

Scenario: Remove Semaphores Older Than Provided Retention Time

Given there are semaphores in the system

And some of these semaphores are older than the provided retention time

When the cleanup process is triggered

Then the system should remove semaphores older than the provided retention time

And indicate that the cleanup process has been successfully completed

This scenario describes the behavior of removing semaphores older than the provided retention time. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Management

Scenario: Fetch Semaphores

Given there are semaphores stored in the system

When the fetch semaphores operation is performed

Then the system should return a list of all semaphores

And indicate that the operation was successful

This scenario describes the behavior of fetching semaphores from the system. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Cleanup

Scenario: Release Orphaned Semaphores

Given there are orphaned semaphores in the system

When the release orphaned semaphores process is triggered

Then the system should release all orphaned semaphores

And indicate that the release process was successful

This scenario describes the behavior of releasing orphaned semaphores from the system. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Management

Scenario: Release Semaphore

Given there is a semaphore currently held in the system

When the release semaphore operation is performed

Then the system should release the semaphore

And indicate that the semaphore has been successfully released

This scenario describes the behavior of releasing a semaphore from the system. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Management

Scenario: Return Conflict When Releasing Non-Existent Semaphore

Given there is no semaphore currently held in the system

When an attempt is made to release a semaphore

Then the system should return a conflict status

And indicate that the semaphore does not exist

This scenario describes the behavior of returning a conflict status when attempting to release a non-existent semaphore from the system. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Lock Renewal

Scenario: Renew Lock

Given there is an existing lock held in the system

When the renew lock operation is performed

Then the system should renew the lock

And indicate that the lock has been successfully renewed

This scenario describes the behavior of renewing a lock in the system. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.

Feature: Semaphore Renewal

Scenario: Return 404 When Renewing Non-Existent Semaphore

Given there is no semaphore currently held in the system

When an attempt is made to renew the semaphore

Then the system should return a 404 status

And indicate that the semaphore does not exist

This scenario describes the behavior of returning a 404 status when attempting to renew a non-existent semaphore in the system. You can implement the step definitions for this scenario in your test automation framework to validate this behavior.