



SSF Tools: Mock Integration Executor User Guide

Document Revision History

Revision Date	Written/Edited By	Comments
May 2018	Matt Shirilla	Initial documentation.
June 2018	Paul Wheeler	Minor amendments for SSD v6.

© Copyright 2018 SailPoint Technologies, Inc., All Rights Reserved.

SailPoint Technologies, Inc. makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SailPoint Technologies shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Restricted Rights Legend. All rights are reserved. No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of SailPoint Technologies. The information contained in this document is subject to change without notice.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Regulatory/Export Compliance. The export and reexport of this software is controlled for export purposes by the U.S. Government. By accepting this software and/or documentation, licensee agrees to comply with all U.S. and foreign export laws and regulations as they relate to software and related documentation. Licensee will not export or reexport outside the United States software or documentation, whether directly or indirectly, to any Prohibited Party and will not cause, approve or otherwise intentionally facilitate others in so doing. A Prohibited Party includes: a party in a U.S. embargoed country or country the United States has named as a supporter of international terrorism; a party involved in proliferation; a party identified by the U.S. Government as a Denied Party; a party named on the U.S. Government's Entities List; a party prohibited from participation in export or reexport transactions by a U.S. Government General Order; a party listed by the U.S. Government's Office of Foreign Assets Control as ineligible to participate in transactions subject to U.S. jurisdiction; or any party that licensee knows or has reason to know has violated or plans to violate U.S. or foreign export laws or regulations. Licensee shall ensure that each of its software users complies with U.S. and foreign export laws and regulations as they relate to software and related documentation.

Trademark Notices. Copyright © 2018 SailPoint Technologies, Inc. All rights reserved. SailPoint, the SailPoint logo, SailPoint IdentityIQ, and SailPoint Identity Analyzer are trademarks of SailPoint Technologies, Inc. and may not be used without the prior express written permission of SailPoint Technologies, Inc. All other trademarks shown herein are owned by the respective companies or persons indicated.

Table of Contents

Introduction	4
Installation	4
Components	4
How it Works	4
Configuration	5
Mock Integration Executor Specific Configuration	5
assumeSuccessfulTicketOperations	5
customObjectName	5
simulatedLatency	5
IdentityIQ Generic Configuration	6
ManagedResources	6
StatusMap	6

Introduction

The Mock Integration Executor is an easy way to simulate a ticket system in your development environment so that you can test integration-related workflows and other integration-dependent features. It can also help you simulate situations where there is a high latency in the integration provisioning methods. It will work with IdentityIQ 6.4 and above.

Installation

The Mock Integration Executor ships with the SSD and is deployed by default as part of the build process if your IdentityIQ version is 6.4 or greater. Deployment of the IntegrationConfig object used by the tool can be switched on or off with the “deployMockIntegrationExecutor” property in the build.properties file. Set this to “true” to enable deployment, or “false” to disable.

Components

The tool includes two Java classes for the Integration Executor. After compilation by the build process, the Java classes can be found here:

- WEB-INF/classes/sailpoint/services/standard/mockintegrationexecutor/MockIntegrationExecutor.class
- WEB-INF/classes/sailpoint/services/standard/mockintegrationexecutor/TicketSystemClient.class

In addition there is an IntegrationConfig object:

- config/SSF_Tools/MockIntegrationExecutor/Source/IntegrationConfig/IntegrationConfig-MockIntegrationConfig.xml

How it Works

The Mock Integration Executor integrates with IdentityIQ workflows in the same way as any other integration. For more information on how to configure workflow options that affect integrations in access request workflows refer to these documents:

Lifecycle Manager Workflows: <https://community.sailpoint.com/docs/DOC-4897>

LCM Subprocess Workflows: <https://community.sailpoint.com/docs/DOC-4917>

This is how the Mock Integration Executor is used:

1. An access request is submitted, approved, and the workflow is in the Approve and Provision stage.
 - In the Access Request UI there will be a section for the Mock Integration Executor that shows:
 - The status is pending.
 - The Provisioning Request ID is the IdentityRequest #.
 - The Mock Integration Executor “opens a ticket” by adding to the tickets map in the MockIntegration Custom object.
 - The ticket number is the number of the access request.

- The ticket has a status of “queued”.
 - If the Custom object does not exist, the Mock Integration Executor will create it, so you never have to.
- 2. The developer manually edits the status of the ticket in the Custom object by updating it from “queued” to “committed” using the IdentityIQ debug screen.
- 3. The Perform Maintenance task runs and processes the “Check Status of Queued Items” workflow. In the Access Request UI the section for the Mock Integration Executor shows the status is finished.

Note that by default in IdentityIQ the status of the ticket will only be checked once every hour, even when the Perform Maintenance task is run more frequently. During development and testing stages it is useful to have it check more frequently. This can be done by modifying the script that sets the value of the *provisioningStatusCheckInterval* variable of the “Check Status of queued items” workflow; the integer value returned by the script is the number of minutes it will wait before checking ticket statuses.

Configuration

Mock Integration Executor Specific Configuration

These are settings that are specific to the Mock Integration Executor, configured in the IntegrationConfig object.

assumeSuccessfulTicketOperations

There is an option in the IntegrationConfig to automatically set the status of the ticket to "queued" when it is provisioned and set it status to "committed" automatically when the first check status occurs. This is handy if you want to avoid setting the ticket status manually, yet you want to have the provisioning by integration.

Note that IdentityIQ checks the status of a ticket immediately after provisioning, so if you use this option you are effectively opening and closing the ticket in the provisioning step. The workflow will never go to the check status step if you use this option.

```
<entry key="assumeSuccessfulTicketOperations" value="true"/>
```

customObjectName

The IntegrationConfig object has a customObjectName attribute that sets the name of the Custom object that contains the tickets. The Custom object will be created automatically when the first ticket is generated, if it does not already exist.

```
<entry key="customObjectName" value="MockIntegration"/>
```

simulatedLatency

The Mock Integration Executor simulates the latency of the ticketing system by calling “Thread.sleep(*simulatedLatency*);” at the start of any provisioning or check status call. This is useful for testing how your IdentityIQ environment handles high latency provisioning. The value is in milliseconds.

```
<entry key="simulatedLatency" value="1000"/>
```

IdentityIQ Generic Configuration

These are configuration items that are common to all integrations and can be used with the Mock Integration Executor. These are set in the IntegrationConfig object.

ManagedResources

This property denotes which application(s) will be provisioned via the integration.

```
<ManagedResources>
  <ManagedResource>
    <ApplicationRef>
      <Reference class="sailpoint.object.Application" name="AppName1"/>
    </ApplicationRef>
  </ManagedResource>
  <ManagedResource>
    <ApplicationRef>
      <Reference class="sailpoint.object.Application" name="AppName2"/>
    </ApplicationRef>
  </ManagedResource>
</ManagedResources>
```

StatusMap

This attribute maps the status from the ticket system to the internal IdentityIQ status.

```
<entry key="statusMap">
  <value>
    <Map>
      <entry key="closed" value="committed"/>
      <entry key="open" value="queued"/>
      <entry key="retry" value="queued"/>
    </Map>
  </value>
</entry>
```