SailPoint

Version 6.4

Integration Guide

Table of Contents

Chapter 1: Overview 1

What is SailPoint IdentityIQ? 1
SailPoint Integration Guide Overview 1

Standard Deployment Integration Modules 3

Chapter 2: SailPoint Oracle Identity Manager Provisioning Integration Module 5

Overview 5
Supported features 5
Supported platforms 6
Installing the OIM Integration Web Application 6
Testing the OIM Integration Web Application 7
Properties that can be defined in xellerate.properties 8
Configuration for OIM application 9
Testing the OIM Integration Client 9
Aggregating from OIM 10
Known/Open issues 10

Chapter 3: SailPoint Sun Identity Manager Provisioning Integration Module 11

Introduction 11
Supported features 11
Supported platforms 12
General configuration 12
Configuration for Aggregation 12
Configuration for Provisioning 13

Chapter 4: SailPoint Novell Identity Manager Provisioning Integration Module ... 15

Overview 15
Supported features 15
Supported platforms 16
Pre-requisites 16
Connected systems 16
Role Base Entitlements (RBEs) 17
User Service Driver configuration 17
IdentityIQ Provisioning workflows 18
Enabling the Web Service 20
Configuring IdentityIQ for Novell Identity Manager Integration 20
Create Novell IDM application in IdentityIQ 20
Run the Novell Application Creator Task 20
Aggregation from Novell PIM 22

Chapter 5: SailPoint IBM Tivoli Provisioning Integration Module 23

Overview 23
Supported features 23
Supported platforms 23
General configuration 24
Configuration for Aggregation 24
Configuration for Provisioning 24

Chapter 6: SailPoint BMC Remedy Service Desk Service Integration Module 27
Overview 27
Supported features 27
Supported platforms 27
Pre-requisites 28
Basic configuration 28
Configuring BMC Remedy AR System for IdentityIQ Integration 29
Configuring IdentityIQ for BMC Remedy Action Request System Integration 31
Prerequisites 31
BMC Remedy Action Request System Integration 31
Sample scenario 35
Known/Open Issues 36
Chapter 7: SailPoint ServiceNow Service Integration Module 37
Overview 37
Supported features 37
Supported platforms 37
Pre-requisites 38
Basic configuration 39
Configuring ServiceNow for IdentityIQ Integration 41
ServiceNow Integration 41
Upgrade 46
Sample scenario 46
Troubleshooting 48
Chapter 8: SailPoint BMC Remedy Service Request Management Adapter 51
Overview 51
Supported platforms 52
Configuring BMC Remedy Service Request Management for IdentityIQ Integration 52
Install 52
Configure 52
Verify 54
Troubleshooting 54
Assisted Deployment Integration Modules 57
Chapter 9: SailPoint Microsoft Forefront Identity Manager Provisioning Integration Module 59
Overview 59
Supported features 61
Supported platforms 61
IQService 61
Microsoft Forefront Identity Manager 2010 Extensible Connectivity 2.0 Management Agent 62
Configuring IdentityIQ for Forefront Identity Manager Integration 63
Create Forefront Identity Manager application in IdentityIQ 63
Run Microsoft Forefront Identity Manager Application Creator Task 64
Operation specific configuration on Microsoft Forefront Identity Manger 65
Troubleshooting 67
Known/Open issue 68
Chapter 10: SailPoint HP Service Manager Service Integration Module 71
Overview 71

Supported features 71
Supported platforms 71
Pre-requisites 72
Troubleshooting 76

Chapter 11: SailPoint ServiceNow Service Catalog Integration 77

Overview 77
Supported features 78
Supported platforms 79
Pre-requisites 79
Installation and configuration in ServiceNow 79
Installation 79
Configuration 81
Configuration in SailPoint IdentityIQ 82
Troubleshooting 83

Chapter 12: SailPoint AirWatch Mobile Device Management Integration Module ..85

Overview 85
Supported features 85
Supported platforms 86
Pre-requisites 86
Configuration 86
Application configuration 87
Operation specific configuration 87

Chapter 13: SailPoint MobileIron Mobile Device Management Integration Module 89

Overview 89
Supported features 89
Supported platforms 90
Pre-requisites 90
Configuration 90
Application configuration 90
Operation specific configuration 91

Chapter 14: SailPoint Good Technology Mobile Device Management Integration Module 93

Overview 93
Supported features 94
Supported platforms 94
Pre-requisites 94
Configuration 94
Application configuration 94
Operation specific configuration 95

Chapter 15: SailPoint HP ArcSight Integration Module 97

Overview 97
Common Event Format (CEF) 97
Supported features 98
Pre-requisites 98
Configuration 98
Configuration to export IdentityIQ Data to ArcSight 98
Configuration to Import HP ArcSight CEF Flat File to SailPoint IdentityIQ 102

Chapter 16: SailPoint Lieberman Integration 105
Lieberman Target Collector 105
Troubleshooting 106
Chapter 17: SailPoint STEALTHbits Integration 107
STEALTHbits Target Collector 107
STEALTHbits target collector configuration 107
Configuring custom queries 109
Rights Mapping 109
Appendix 111
Appendix A: Common Identity Management Integration Configuration 113
Overview 113
Creating the IntegrationConfig Object 113
Provisioning 118

Chapter 1: Overview

The following topics are discussed in this chapter:

What is SailPoint IdentityIQ? 1
SailPoint Integration Guide Overview 1

What is SailPoint IdentityIQ?

SailPoint IdentityIQ is an identity and access management solution for enterprise customers that delivers a wide variety of IAM processes-including automated access certifications, policy management, access request and provisioning, password management, and identity intelligence. Furthermore, IdentityIQ has a flexible connectivity model that simplifies the management of applications running in the datacenter or the cloud.

Compliance Manager: Compliance Manager automates access certifications, policy management, and audit reporting through a unified governance framework. This allows you to streamline compliance processes and improve the effectiveness of identity governance-all while lowering costs

Lifecycle Manager: Lifecycle Manager manages changes to access through user-friendly self-service request and password management interfaces and automated lifecycle events. It provides a flexible, scalable provisioning solution for addressing the constantly evolving access needs of your business in a way that's both efficient and compliant.

Unified Governance Platform: Governance Platform lays the foundation for effective IAM within the enterprise. It establishes a single framework that centralizes identity data, captures business policy, models roles, and takes a risk-based, proactive approach to managing users and resources. The Governance Platform also offers extensible analytics that transforms disparate, technical identity data into relevant business information. Additionally, robust resource connectivity is provided that allows organizations to directly connect to applications running in the datacenter or in the cloud.

A unified governance platform allows organizations to build a single preventive and detective control model that supports all identity business processes, across all applications-in the datacenter and the cloud. SailPoint IdentityIQ applies consistent governance across compliance, provisioning and access management processes, maximizing investment and eliminating the need to buy and integrate multiple products.

Integration Modules: The Integration Modules enables you to manage applications running in the datacenter or cloud through a flexible set of integration options. It directly connects to applications and provides out-of-box integration with other IT management tools, such as third-party provisioning, IT Service Management, Mobile Device Management or IT Security. This wide variety of connectivity options speeds deployment and reduces costs.

SailPoint Integration Guide Overview

SailPoint Integration Modules deliver extended value from standard IdentityIQ deployments. SailPoint is committed to providing design, configuration, troubleshooting and best practice information to deploy and maintain strategic integrations. SailPoint has modified the structure of this document to aid customers and partner deployments. The focus of this document is product configuration and integration. For more details on design, troubleshooting and deployment best practices, refer to the Connector and Integration Deployment Center in Compass, SailPoint's online customer portal.

SailPoint Integration Guide Overview

This document provides a guide to the integration between the following products and IdentityIQ:

- **"Standard Deployment Integration Modules"**
  - Provisioning Integration Modules
    - SailPoint Oracle Identity Manager Provisioning Integration Module
    - SailPoint Sun Identity Manager Provisioning Integration Module
    - SailPoint Novell Identity Manager Provisioning Integration Module
    - SailPoint IBM Tivoli Provisioning Integration Module
  - Service Desk Integration
    - SailPoint BMC Remedy Service Desk Service Integration Module
    - SailPoint ServiceNow Service Integration Module
  - Service Catalog Integration
    - SailPoint BMC Remedy Service Desk Service Request Management Adapter
- **"Assisted Deployment Integration Modules"**
  - Provisioning Integration Modules
    - SailPoint Microsoft Forefront Identity Manager Provisioning Integration Module
  - Service Desk Integration
    - SailPoint HP Service Manager Service Integration Module
  - Service Catalog Integration
    - SailPoint ServiceNow Service Catalog Integration
  - Mobile Device Management Integration Modules
    - SailPoint AirWatch Mobile Device Management Integration Module
    - SailPoint MobileIron Mobile Device Management Integration Module
    - SailPoint Good Technology Mobile Device Management Integration Module
  - IT Security Integration Module
    - SailPoint HP ArcSight Integration Module
    - SailPoint Lieberman Integration Module
- SailPoint Data Access Governance Integration Module
  - SailPoint STEALTHbits Integration Module

This document is intended for the above products and IdentityIQ System Administrators and assumes a high degree of technical knowledge.

Standard Deployment Integration Modules

This section contains information on the following sections:

- **Provisioning Integration Modules**

  IdentityIQ can be configured to integrate with provisioning providers to automate access management for your implementation. Provisioning providers can be configured to communicate user and account information and automatically add or revoke access.

  - "SailPoint Oracle Identity Manager Provisioning Integration Module" on page 5
  - "SailPoint Sun Identity Manager Provisioning Integration Module" on page 11
  - "SailPoint Novell Identity Manager Provisioning Integration Module" on page 15
  - "SailPoint IBM Tivoli Provisioning Integration Module" on page 23

- **Service Management Integration Modules**

  IdentityIQ supports Service Management Systems for Service Desk Integration and Service Catalog Integration.

  - **Service Desk Integration**: Creates tickets in Helpdesk systems based on compliance and provisioning actions taken in IdentityIQ (For example, new account request).
    - "SailPoint BMC Remedy Service Desk Service Integration Module" on page 27
    - "SailPoint ServiceNow Service Integration Module" on page 37
  - **Service Catalog Integration**: Enables enterprises to include LCM requests (for example, access request, password change) in the service catalog of the Service Request Management system.
    - "SailPoint BMC Remedy Service Request Management Adapter" on page 51

Many SailPoint customers have deployed the Integration Modules in this section. SailPoint still encourages deployment teams to obtain the latest troubleshooting and best practice information beyond what is contained in this document. For more specific information, refer to the Connector and Integration Deployment Center in Compass, SailPoint's online customer portal.

Chapter 2: SailPoint Oracle Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

Overview 5
Supported features 5
Supported platforms 6
Installing the OIM Integration Web Application 6
Testing the OIM Integration Web Application. 7
Configuration for OIM application 9
Aggregating from OIM. 10
Known/Open issues 10

Overview

This chapter provides a guide to the integration between Oracle Identity Manager (OIM) and IdentityIQ. This chapter is intended for Oracle and IdentityIQ System Administrators and assumes a high degree of technical knowledge.

The integration is achieved by deploying a small web application in the application server that hosts OIM. IdentityIQ communicates with the web services contained in this application to read and write account information. Configuration of the OIM integration requires the username and password of the OIM administrator or another user with sufficient permissions.

Supported features

The Oracle Identity Manager Provisioning Integration Module supports the following functions:

- Account Management
  - Oracle Identity Manager user aggregation along with the connected child accounts and application
  - Create, Update, Delete
  - Enable, Disable, Unlock
  - Add/Remove Entitlements operations for Oracle Identity Manager connected child accounts
- User Management
  - Manages Oracle Identity Manager Users as Accounts
  - Create, Update, Delete
  - Enable, Disable, Unlock
  - Add/Remove Entitlements operations for Oracle Identity Manager Users

Supported platforms

- Oracle Identity Manager 11g R2
- Oracle Identity Manager 11g R1
- Oracle Identity Manager 9 (Oracle 10g)

Installing the OIM Integration Web Application

You must first deploy the OIM Integration Servlet web application to the application server hosting the OIM application. The `iiq.war` file for this web application is contained in the IdentityIQ distribution as $INSTALLDIR/`integration/OIM/iiqIntegration-OIM.jar` or in the distribution for an IdentityIQ patch in a `.jar` file named `Integration-oim-<version>.jar`.

The `iiqIntegration-OIM.jar` file contains `iiq.war` file. You can customize the `iiq.war` file in many ways before being deployed into the application server hosting OIM.

Note: Ensure that if you are deploying web application as war file, it should be named as iiq.war. If you are deploying the web application from a directory, then directory should be named as iiq.

The following are the required customization steps:

1. Configure access to OIM by modifying `WEB-INF/classes/xellerate.properties` to set.

- **XL.HomeDir**: the full path to the directory where OIM is installed
- **userName**: the OIM administrator that has the appropriate permission to read and write user and account data
- **password**: the password for the OIM administrator

  For more information on the other properties that need to be set in xellerate.properties file, see “Properties that can be defined in xellerate.properties” on page 8.

2. Copy the required API implementation .jar files from the OIM installation into the `WEB-INF/lib` directory of the integration application:

- (***For Oracle 11g***) Copy `OIM_ORACLE_HOME/designconsole/lib/oimclient.jar`
- (***For Oracle 10g***)
  - Copy the following files from the OIM web lib directory to **iiq.war** only if **iiq.war** and OIM server are installed on separate WebLogic managed server:
    - `javagroups-all.jar`
    - `oscache.jar`
    - `wlXLSecurityProviders.jar`
    - `XIMDD.jar`
    - `XL10SecurityProviders.jar`

- Copy the following files from the OIM external lib directory
  - xlAdapterUtilities.jar
  - xlAPI.jar
  - xlAttestation.jar
  - xlAuditor.jar
  - xlAuthentication.jar
  - xlBackOfficeBeans.jar
  - xlBackofficeClient.jar
  - xlCache.jar
  - xlCrypto.jar
  - xlDataObjectBeans.jar
  - xlDataObjects.jar
  - xlDDM.jar
  - xlGenConnector.jar
  - xlGenericUtils.jar
  - xliGCProviders.jar
  - xlInputPreprocessor.jar
  - xlInstaller.jar
  - xlLogger.jar
  - xlRemoteManager.jar
  - xlRequestPreview.jar
  - xlSampleApp.jar
  - xlScheduler.jar
  - xlUtils.jar
  - xlVO.jar
  - xlWebClient.jar
  - xlWSCustomClient.jar

Testing the OIM Integration Web Application

Verify if the installation was successful using the following steps:

Note: For each test URL throughout this document, change the host name and port to match your OIM Server instance.

1. From any browser enter the following URL:
   http://localhost:8080/iiq/resources/ping
   The following response is displayed:
   ```
   OIM integration ready
   ```
   Failure to get a ping response indicates a problem with the deployment of the servlet.
2. Verify the integration servlet can communicate with OIM by entering the following URL:
   http//localhost:8080/iiq/resources/users

   You should see a response containing the names of all OIM users. This might take a while to assemble depending on the number of users. To view details of a particular user, enter the following URL where <OMIUSER> is the name of a user in your OIM instance:
   http://localhost:8080/iiq/resources/user/<OIMUSER>

To see additional diagnostic information for of a particular user, enter the following URL where `<OMIUSER>` is the name of a user in your OIM instance:

http://localhost:8080/iiq/resources/debug/<OIMUSER>

If you are unable to request user information, there may be a problem with the credentials you entered in the `xellerate.properties` file. For more information, see "Properties that can be defined in xellerate.properties" on page 8.

Properties that can be defined in xellerate.properties

1. Add a ManagedResource definition in the ManagedResource list for an each OIM resource. For each resource, define a property prefix by adding a property whose name is the prefix and whose value is the OIM resource name.

   For example:
   AD=AD User

   Oracle=Oracle DB User

   This declares that any property that begins with ERP is related to the OIM resource named ERP Central Component.

2. For each ManagedResource, define the account attribute that represents the unique account identifier. The names used here must be the resource names used by OIM. The identityAttribute must have the internal form field name containing the account identifier. Use the OIM Design Console application to find the process form for each resource and view the field names. The example below gives two typical names, one used by the connector for Oracle database users and the other for the Active Directory connector.
   ```
   AD.id=UD_ADUSER_UID
   Oracle.id=UD_DB_ORA_U_USERNAME
   ```

3. Define the names of the child forms that support multiple attributes. The value is a CSV of the internal child form names:
   ```
   AD.childForms=UD_ADUSRC
   Oracle.childForms=UD_DB_ORA_R
   ```
   In this example UD_ADUSRC is the internal name for the child form AD User Group Details and UD_DB_ORA_R is the internal name for the child form DBUM Grant/Revoke Roles.

4. Each child form name in the Oracle.childForms property there is another property whose value is a CSV of the child form fields to return and the order in which they will appear in IdentityIQ.
   Oracle.UD_DB_ORA_R=UD_DB_ORA_R_ROLE,UD_DB_ORA_R_ADMIN_OPTION

   In the previous example, we will return two fields from the child form UD_DB_ORA_R. The first field has the Role name and the second has the Role Admin option.

5. Following is the configuration for resource with child forms: ERP Central Component:
   ```
   ERP=ERP Central Component
   ERP.id=UD_ECC_USER_ID
   ERP.childForms=UD_ECC_PRO,UD_ECCRL
   ERP.UD_ECC_PRO=UD_ECC_PRO_SYSTEMNAME,UD_ECC_PRO_USERPROFILE
   ERP.UD_ECCRL=UD_ECCRL_SYSTEMNAME,UD_ECCRL_USERROLE
   ```

   Note: Before IdentityIQ 6.0 there was a parameter in **xellerate.properties** file as **oldChildFormNames** which was used for the resources who have only one field in the childform, for example, Active Directory resource. For **IdentityIQ** version 6.0 onwards, the value must be set to true if the user wants to support **oldChildFormNames** where field returned would be form name + field name (For example, **UD_ADUSRC:UD_ADUSRC_GROUPNAME** field in Active directory).

6. To aggregate all the active and disabled OIM users in IdentityIQ, add a new parameter **OIM_USER_TYPE** in `xelerate.properties` file with the value as **ALL**. If **OIM_USER_TYPE** parameter is removed from the `xelerate.properties` file then only the active OIM users will be aggregated. By default only active OIM user are aggregated.

Configuration for OIM application

Perform the following steps to create an IdentityIQ application for OIM:

1. Navigate to the IdentityIQ **Define=>Application** page.
2. Create a new application of type **Oracle Identity Manager**.
3. On the Attributes tab, enter the Oracle Identity Manager Host and Oracle Identity Manager Port.
4. Click **Test Connection** to verify the connection to OIM.

Note: You can make use of the "OIM Application creator" task to discover all the resources present in OIM environment. The input for this task would be an newly created application of type "Oracle Identity Manager" and executing this task would result in the creation of all multiplexed resources.

Testing the OIM Integration Client

While any IdentityIQ feature that generates a provisioning request such as a certification remediation, a role assignment, or a Lifecycle Manager request can be used to test the integration, it is sometimes useful to test at the provisioning layer using the IdentityIQ integration console.

Launch the console by using the IdentityIQ script in the `INSTALLDIR/WEB-INF/bin` directory of the IdentityIQ installation to run iiq integration.

From the console command prompt, use the **list** command to display the names of all Application objects created in the system. Using the example in the previous section, verify an Application object of type **Oracle Identity Manager** exits.

Use the following command:

```
use OIMApplicationName
```

Use the **ping** command to initiate a test connection message with OIM. A successful connection will return the following message:

```
Response: Connection test successful
```

If any problem occurs in the communication of this application with the OIM Integration Web Application, troubleshoot this application by viewing the application server logs for both the IdentityIQ and OIM application servers. You can enable log4j tracing on both sides by using the following:

```
log4j.logger.sailpoint.integration=debug
```

```
log4j.logger.sailpoint.connector=debug
```

This lets you see if the requests are transmitting over the network, and how they are processed.

If the OIM servlet is deployed on Weblogic 11g, tracing can be enabled on it by adding an entry to the logging file on the Weblogic server. Following is the logging file:

```
<DOMAIN_HOME>/config/fmwconfig/servers/oim_server1/logging.xml
```

Following is the entry that needs to be added:

```
<logger name="SailPoint.integration.oim" level="TRACE:32"/>
```

For more information on enabling system logging in OIM is included in the *Oracle Identity Manager Administrator Guide*.

Aggregating from OIM

To aggregate OIM users and resource accounts, create and execute an IdentityIQ Account Aggregation task. Include the OIM application in the applications to scan list.

When the aggregation is complete from the OIM application, a new application is created for every resource in OIM. The application schema includes attributes seen in the resource accounts. All users in OIM have an account created that is associated with the OIM application and includes all of the standard and extended user attributes of those users. Additionally, all of the resource accounts are aggregated and associated with the newly created applications.

Once the initial aggregation from the OIM application is completed, you can aggregate from it again to read in information for all managed systems.

Note: **You can make use of the "OIM Application Creator task" to discover all of the Resources present in the OIM environment. The input for this task is an application of type Oracle Identity Manager. Executing this task results in the creation of all multiplexed Resource applications.**

Known/Open issues

Following is the known/open issue of Oracle Identity Manager:

- You cannot perform provisioning operations simultaneously on the OIM server from IdentityIQ and the OIM console. This is a class loading issue observed with OIM 11g, after deploying iiq servlet(iiq.war) on Weblogic OIM Managed Server.

  Workaround for this issue: Create another, empty WLS(Weblogic)Managed server next to the OIM Managed Server and only deploy the IIQ Servlet. Also, update the **Xellerate.properties** file by un-commenting the attribute **java.naming.provider.url**. This Url needs the host name of the host where OIM managed server is deployed and the listening port of the OIM managed server.
- Create OIM user and Update OIM user operations are not working with Oracle Identity Manager 11g R2.

Chapter 3: SailPoint Sun Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

Introduction 11
Supported features 11
Supported platforms 12
General configuration 12
Configuration for Aggregation 12
Configuration for Provisioning 13

Introduction

The integration between IdentityIQ and Sun Identity Manager is very straightforward and requires little configuration on either side to start the initial communication. This document covers the important points for configuring an integration between these products.

Supported features

The Sun Identity Manager Provisioning Integration Module provides the ability to provision Sun Identity Manager users, Target Application accounts, groups and entitlements from IdentityIQ.

The Provisioning Integration Module supports the following functions:

- User Management
  - Manages Sun Identity Manager Users as Accounts
  - Aggregating Users
  - Create, Update, Delete
  - Enable, Disable, Unlock, Reset Password
  - Add/Remove User Entitlements
- Target Application Accounts Management
  - Manages Target Application Accounts as Accounts
  - Target Accounts are aggregated as part of Sun IdM User aggregation
  - Create, Update, Delete
  - Enable, Disable, Reset Password
  - Add/Remove Account Entitlements

- Group Management
  - Manages Sun Identity Manager Groups as Account-Groups in IdentityIQ
  - Groups are aggregated as part of user aggregation task

Supported platforms

- Oracle Waveset 8.1.1
- Sun Identity Manager 8.1
- Sun Identity Manager 8.0
- Sun Identity Manager 7.1
- Sun Identity Manager 7.0

General configuration

IdentityIQ communicates with Sun Identity Manager using a Service Provisioning Markup Language (SPML) interface. To enable this communication, the SPML client library, `openspml.jar`, must be copied from the Sun IdM installation into the `INSTALLDIR/WEB-INF/lib` directory of the IdentityIQ installation and then the IdentityIQ application should be restarted. A system error will occur in IdentityIQ when defining a Sun Identity Manager application before this configuration step is completed.

Configuration for Aggregation

Configuring IdentityIQ to aggregate accounts from Sun Identity Manager is accomplished by creating an IdentityIQ application as outlined in the following steps:

Configure SPML in Sun Identity Manager

1. If you do not have a Configuration:SPML object in Sun Identity Manager, import **sample/spml.xml** from the Sun Identity Manager installation.

   Note: SPML v1 requests are used only for Aggregation and Provisioning.

2. Add the following object into the SPML classes list:

```
<Object name='IIQUserView'>
  <Attribute name='type' value='User'/>

  <!-- keep the default form quiet -->
  <Attribute name='form'       value='view'/>
  <Attribute name='viewForm'   value='Empty'/>
  <Attribute name='identifier' value='waveset.accountId'/>
  <Attribute name='default'    value='true'/>

<!-- May need this if the install ordinarily turns on the meta-view -->
  <Attribute name='viewOptions'>
    <Object>
      <Attribute name='ApplyMetaView' value='false'/>
    </Object>
```

```
      </Attribute>

<!-- This is important for ModifyRequests to remove selected elements.
      We'll have to put something here for every multi-valued attribute
     in every managed resource.  Tedious but at this point necessary.
      <Attribute name='multiValuedAttributes'>
        <List>
          <String>IDM/IdentityIQ Integration Demo::groups</String>
          <String>accounts[IDM/IdentityIQ Integration Demo].groups</String>
        </List>
      </Attribute>
      -->
    </Object>
```

Define IdentityIQ Application

1. The object name (in this case IIQUserView) should match the Native Object Type of the account schema IdentityIQ application that represents Sun Identity Manager.
2. When creating the Sun Identity Manager application in IdentityIQ, follow the default pattern for the rpcRouterURL to point to the Sun Identity Manager system.

To aggregate Sun Identity Manager users and resource accounts, create and execute an IdentityIQ Account Aggregation task for the Sun Identity Manager application.

When the aggregation task is complete, a new application is created for every managed resource in Sun Identity Manager. The application schema includes attributes from the resource accounts. All users in Sun Identity Manager have an account created that is associated with the Sun Identity Manager application and any administrative capabilities and assigned profiles are included as attributes or direct permissions of those accounts.

Once the initial aggregation from the Sun Identity Manager application is completed, you can aggregate from it again to read in information for all of the resource systems.

Configuration for Provisioning

No additional configuration is needed to enable provisioning to Sun Identity Manager. The Sun Identity Manager PIM is implemented using the latest IdentityIQ connector interface that provides read/write capability. If you wish to provision accounts for Sun Identity Manager applications using another method, then you must remove the **PROVISIONING** keyword from the **featureString** in the Sun Identity Manager application in IdentityIQ.

Chapter 4: SailPoint Novell Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

Overview 15
Supported features 15
Supported platforms 16
Pre-requisites 16
Configuring Novell Identity Manager for IdentityIQ Integration 16
Connected systems 16
Role Base Entitlements (RBEs) 17
User Service Driver configuration 17
IdentityIQ Provisioning workflows 18
Enabling the Web Service 20
Configuring IdentityIQ for Novell Identity Manager Integration 20
Create Novell IDM application in IdentityIQ 20
Run the Novell Application Creator Task 20

Overview

This chapter provides a guide to the Novell® Identity Manager and SailPoint integration and configuration for your enterprise. This chapter is intended for Novell and IdentityIQ System Administrators and assumes a high degree of technical knowledge of these systems.

Supported features

The Novell Identity Manager Provisioning Integration Module provides the ability to provision Novell Identity Manager users, Target Application accounts and entitlements from IdentityIQ.

The Provisioning Integration Module supports the following functions:

- User Management
  - Manages Novell Identity Manager Users as Accounts
  - Aggregating Users
  - Create, Update, Delete
  - Enable, Disable

- Target Application Accounts Management
  - Manages Target Application Accounts as Accounts
  - Target Accounts are aggregated as part of Novell Identity Manager User aggregation
  - Create, Update, Delete
  - Enable, Disable
  - Add/Remove Account Entitlements
- Group Management
  - Manages Novell Identity Manager Groups as Account-Groups
  - Aggregating Groups
    Note: While aggregating Novell IdM Users/Groups, the connected target system Accounts/Groups are also aggregated.

Note: Before you can use the provisioning feature of the connector, all PRDs must be defined and configured on Novell side. For more information about the PRD's, see "IdentityIQ Provisioning workflows" on page 18 section.

Supported platforms

- Novell Identity Manager 4.0
- Novell Identity Manager 3.0

Pre-requisites

It is mandatory to have the user application module installed on the Novell Identity Manager.

Configuring Novell Identity Manager for IdentityIQ Integration

The integration has dependencies on the following Novell Identity Manager components:

- Connected Systems Drivers
- Role Based Entitlements (RBEs)
- User Service Driver
- IdentityIQ Provisioning Workflows
- Enabling the Web Service

Connected systems

The DirXML drivers, connected to the Identity Vault from which the data is aggregated, have to be mapped to IdentityIQ Applications. This is done using the Novell Application Generator task (see below).

Role Base Entitlements (RBEs)

Role Based Entitlements (RBEs) need to be defined in the Identity Vault before automatic remediation from IdentityIQ can occur. Policies can be setup in the vault to add and remove account membership, group membership, etc. when these RBEs are granted or revoked. When accounts are aggregated into IdentityIQ, these RBEs show up as IdentityIQ entitlements.

User Service Driver configuration

The user service driver provides the provisioning and directory abstraction layer (VDX) web service that IdentityIQ uses to communicate with Novell Identity Manager to perform remediation and aggregation respectively. It is assumed that the user service driver is the Entitlement granting authority.

Note: A conflict occurs if you have an Entitlement Service Driver for the same driverset.

IdentityIQ Directory entities

Directory entities must be defined for the User Service driver that are used by IdentityIQ when it invokes the VDX web service. This entity is defined for the User Service driver under Directory Abstraction Layer.

For Account aggregation, entity must be created with the details mentioned in the following table:

Task	Entity Key	Object Class	Attributes
For Novell Application Creator Task	dirXML-Driver	DirXML-Driver	CN (allow search access to users), Object Class (allow search access to users)
	For Novell 4.*x* the following entity is also required		
	DirXML-Entitlement	DirXML-Entitlement	CN (allow search access to users), Object Class (allow search access to users). Provide value for the Search Container
For Account Aggregation	user	User	CN, Department, Direct Reports, DirXML-Accounts, DirXML-Associations, DirXML-EntitlementRef, Email, First Name, Group, Hidden attribute List, Last Name (allow search access to users), Login Disabled, Manager, Preferred Notification, Query List, Region, Telephone Number, Title
For Account Group Aggregation	group	Group	CN, Description (allow search access to users), DirXML-Associations, DirXML-EntitlementRef, Members

Note: - For entity **DirXML-Driver**, the attributes CN and objectClass are mandatory and they should be marked as Searchable.
- For **user** and **group** entities, the attribute keys should be in sync with account and group schema attribute names in IdentityIQ.
- For user entity, the **LastName** attribute is mandatory and it should be marked as Searchable.
- For group entity, the **Description** attribute is mandatory and it should be marked as Searchable.

Save these entities and import into the identity vault. Note the key of these entities as it is used in IdentityIQ.

IdentityIQ Provisioning workflows

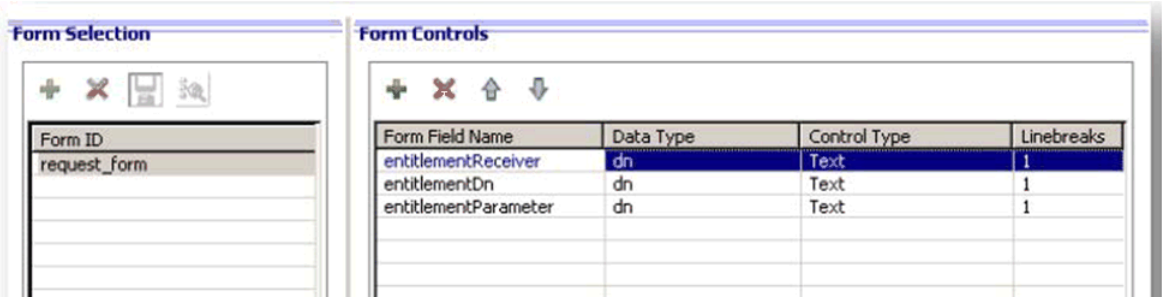
Provisioning workflows must be defined for the User Service driver that are used by IdentityIQ when it invokes the provisioning web service. This workflow is defined for the User Service driver under Provisioning Request Definitions. You can find sample workflows (PRD) under integration/nidm/exampleWorkflows. Import them under Provisioning Request Definitions. After importing add a valid object in Trustee rights. Sample workflows (PRD) for the corresponding operations are as described in the following table:

PRD	Operation	Form fields
SPCreateModifyUser.xml	Create and Update Vault User	FirstName, LastName, Title, Location, Department, Email, loginDisabled, manager, group
SPDeleteUser.xml	Delete Vault User	Form fields are not required for this workflow.
SPDisableEnableUser.xml	Enable and Disable Vault User	loginDisabled
SPEntitlements.xml	Request Access for Vault User	entitlementReceiver, entitlementDn, entitlementAction, entitlementParameter

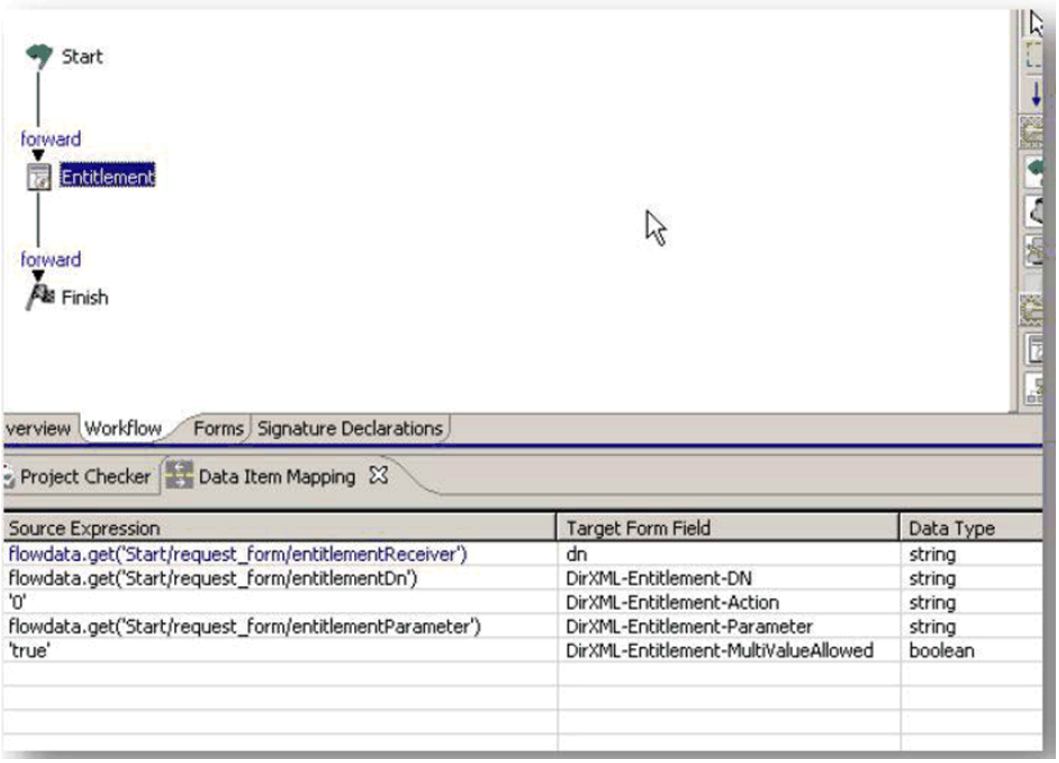
Note: Additional workflow steps and logic might be required based on your business needs.
- For Create and update user workflow, the field names and number of fields in provisioning policy on IdentityIQ side should be same as that of the input field names of workflow.
- For Enable/Disable vault user workflow and Request access workflow (that is, Entitlement), the names of input parameters of workflow should be same as given in above table. Also all mandatory fields on Novell side should be marked as "required" in IdentityIQ provisioning policy.

To define a workflow with the minimum required steps, perform the following:

1. Create a provisioning request definition of type Accounts/Entitlements.
2. Define a simple workflow with three steps, Star, Entity/Entitlement and Finish.
3. Define a form with fields as shown in the Sample PRD below:



4. Verify in the Entity/Entitlement step of the workflow that the Source expressions are defined correctly.



5. Save this workflow and import into the identity vault. Note the dn of this workflow as it is used in defining the Novell Application attributes later in the process.

Enabling the Web Service

The integration uses the following Web Service endpoints provided:

- Directory Abstraction Layer (VDX) Web Service for aggregation
- Provisioning Web Service for remediation

By default, the endpoints are disabled. To provide necessary credentials to the user, see the following sections in the *Novell User Application: Administration Guide:*

- Removing Administrator Credential Restrictions
- Enabling the Test Page

Configuring IdentityIQ for Novell Identity Manager Integration

This section describes the procedure for Novell Identity Manager Integration.

Create Novell IDM application in IdentityIQ

Create a new IdentityIQ application of type Novell Identity Manager. Enter the following required Novell Identity Manager information:

- **Novell Version**: Version of the Novell Identity Manager. The version will accept a value of either 3 or 4. In case any other value is provided, the default value 3 is used.
- **User Application URL**: URL of the User Application, in the following format: http://*applicationServerHost*:*port*/*applicationContext*
- **Username**: User Application Administrator who has the rights to administer Web Services such as Provisioning using workflow tasks and Directory Abstraction Layer (VDX)
- **Password**: Password of the user specified above.

Define Provisioning Request Definition for Novell application

Enter the dn for the workflow created in step 5. on page 19 as application attributes:

1. To Create Vault User: **Create Vault User**
2. To Delete Vault User: **Delete Vault User**
3. To Disable Vault User: **Disable Vault User**
4. To Enable Vault User: **Enable Vault User**
5. To Modify Vault User: **Modify Vault User**
6. To all the operations on Resource Accounts on Applications (Entitlements). This includes Create, Delete, Disable, Enable Resource Accounts or Add/Remove Entitlements: Entitlements

   Note: Account Entitlement for any driver should contain “account” substring in it.

Run the Novell Application Creator Task

This task creates an IdentityIQ application for each connected systems driver in Novell® Identity Manager.

1. Select the Novell application created in the previous section.
2. Click **Fetch Novell Applications**. This shows all the connected systems. Select the ones with which you want IdentityIQ to integrate.
3. Click **Save and Execute**.
4. It will generate child applications for all drivers selected in IdentityIQ.

To fetch Account Status of Account Entitlement, add the following entry in Application's attribute, from the Application debug page:

```
<entry key="statusAttribute" value="loginDisabled"/>
```

In the above entry, **loginDisabled** represents the Status of Accounts on that driver.

Additional steps for Novell 4.0

Following manual configuration changes are required on IdentityIQ as there are number of configuration changes between Novell versions 3.x and 4.x:

1. After defining Novell application, change few attribute names in account and group schema for example, cN to CN, dirXML-Associations to DirXML-Associations. Check attribute names of Novell entity.
2. For generated child applications, add the following attributes in account schema:
   ```
   <AttributeDefinition multi="true" name="Account" type="string"/>
   <AttributeDefinition entitlement="true" managed="true" multi="true" name="Group" type="string"/>
   ```
   Set the Group attribute as group attribute of account schema.
3. Parent Novell application definition in IdentityIQ contains mapping between attribute names in version 4.0 which are used in application internally.

```
<entry key="4.0">
    <value>
        <Map>
          <entry key="cN" value="CN"/>
           <entry key="dirXML-Associations" value="DirXML-Associations"/>
           <entry key="dirXML-EntitlementRef"
value="DirXML-EntitlementRef"/>
          <entry key="LastName" value="LastName"/>
          <entry key="Description" value="Description"/>
          <entry key="objectClass" value="ObjectClass"/>
          <entry key="gUID" value="GUID"/>
          <entry key="loginDisabled" value="LoginDisabled"/>
          <entry key="dirXML-Driver" value="DirXML-Driver"/>
          <entry key="entitlementParameter">
                          <value>
                                 <String>{"ID":"nidm.com"}</String>
                          </value>
          </entry>
        </Map>
    </value>
</entry>
```

4. By default, from IdentityIQ, while sending **User Account Entitlement** or **Group Entitlement**, parameter format is assumed as **Identity Manager 4**. If parameter format is configured as **legacy**, then add the following properties at child application for which parameter format is to be configured:

   ```
   <entry key="accountEntitlementParamFormat" value="legacy"/>
   <entry key="groupEntitlementParamFormat" value="legacy"/>
   ```

   If these properties are absent in application, then IdentityIQ will send parameter format in Identity Manager 4 format.

For Novell version 4.0, **entitlementParameter** indicates name of domain name of Novell setup. Change the value from `nidm.com` to the domain name on which Novell is installed.

Aggregation from Novell PIM

To aggregate Novell users and resource accounts, create and execute an IdentityIQ Account Aggregation task. Include the Novell application in the applications to scan list.

Once aggregation is complete, all users in Novell have an account created that is associated with the Novell application and includes all of attributes defined in Novell application account schema. Additionally, all of the resource accounts are aggregated and associated with the child applications.

Note: **Before running the aggregation task, ensure** "Run the Novell Application Creator Task" on page 20 **has been executed.**

Chapter 5: SailPoint IBM Tivoli Provisioning Integration Module

The following topics are discussed in this chapter:

Overview 23
Supported features 23
Supported platforms 23
General configuration 24
Configuration for Aggregation 24
Configuration for Provisioning 24

Overview

This chapter is designed to provide the necessary procedures, configuration steps, and general product guidelines to successfully integrate IBM Tivoli® Identity Manager (ITIM) into your SailPoint production environment.

This chapter is intended for ITIM and IdentityIQ System Administrators and assumes a high degree of technical knowledge of these systems.

Supported features

The IBM Tivoli Identity Manager Provisioning Integration Module provides the ability to provision Target Application accounts from IdentityIQ.

The Tivoli Provisioning Integration Module supports the following functions:

- User Management
  - Manages IBM Tivoli Identity Manager Users as Accounts
  - Aggregating Users
- Target Application Accounts Management
  - Manages Target Application Accounts as Accounts
  - Aggregating Target Accounts directly
  - Create, Update, Delete
  - Enable, Disable, Unlock, Reset Password

Supported platforms

- SailPoint Deployment
- IBM Tivoli Identity Manager 4.6, 5.0, and 5.1
- IBM WebSphere Application Server 5.1, 6.1 and 7.0

General configuration

The installation steps for ITIM integrations vary based on the functions you wish to perform. IdentityIQ in conjunction with ITIM allows the following functionality:

- Aggregation
- Provisioning Entitlements in ITIM

Configuration for Aggregation

Aggregating from IBM Tivoli Identity Manager involves configuring the ITIM application settings within the IdentityIQ user interface.

ITIM has two types of objects that can be aggregated; people and accounts. IdentityIQ refers to these as identities and accounts (or links). To aggregate from ITIM, perform the following:

1. **Create An ITIM Application**: Create a new application using the IBM Tivoli Identity Manager connector and fill in the required parameters following the steps provided in the IdentityIQ User's Guide.
   Use the tenant DN search base. For example,
   `erglobalid=0000000000000000000,ou=example,dc=com`

   Leave the search filter blank. This is auto-generated correctly during aggregation. This application is used to aggregate ITIM person objects.
2. **Setup Correlation Attribute**: Create an identity attribute that is sourced from the `erglobalid` on the ITIM application and mark it as searchable. This is used to correlate ITIM accounts to this identity.
3. **Create ITIM Account Applications**: Run the ITIM Application Creator task to inspect ITIM and retrieve information about the ITIM services (applications). This task auto-generates an application for each service defined in ITIM.
4. **Setup Correlation on the ITIM Account Applications**: Set the correlation rule on the generated applications to Correlation - ITIM Account. This correlates the account to the identity using the `erglobalid`. If the rule is not listed by default, import it from the `$ITIM_INTEGRATION_PACKAGE/resources/ITIM-Account-CorrelationRule.xml` location.
5. **Aggregate**: Run aggregation for the ITIM application first and then for each ITIM account application.

Configuration for Provisioning

Provisioning entitlements and role assignments in ITIM requires the installation of IdentityIQ's ITIM integration web application in WebSphere with ITIM. This process varies slightly depending on the version of WebSphere.

IdentityIQ roles are queued and pushed in ITIM on a schedule. This is accomplished by using the Synchronize Roles task.

1. **Prepare the WAR:** The `iiqIntegration-ITIM.war` file contains a properties file named `itim.properties` with information about how to connect using ITIM. In order to execute, this must be edited to include appropriate information about the ITIM installation. Additionally, the **.war** file does not include any of the required **ITIM .jar** files since these can change depending on the version and fixpack level of ITIM. These need to be copied out of the ITIM lib directory and added to the **.war** file.
   a. Expand the `iiqIntegration-ITIM.war` file in a temporary directory.

b. Edit the `WEB-INF/classes/itim.properties` file and change the properties match your environment. Save the file with your changes. The following can be changed:

- **PLATFORM_URL:** URL to use to communicate with ITIM.
- **PLATFORM_PRINCIPAL:** Principal to connect as.
- **PLATFORM_CREDENTIALS:** Password of the principal. Encrypting password is supported.
- **TENANT_DN:** The root DN of the ITIM tenant.

c. Copy the required `ITIM .jar` files into the lib directory. These **.jar** files are located in the deployed ITIM ear directory.

- (*For ITIM 4.6*): Example ITIM ear directory: $WAS_HOME/installedApps/itim/enRole.ear

  Following are the required files:

  - `api_ejb.jar`
  - `itim_api.jar`
  - `itim_server.jar`
  - `jlog.jar`

- (*For ITIM 5.0 and 5.1*): Example ITIM ear directory: `$WAS_HOME/profiles/<app server>/installedApps/<cell>/ITIM.ear`
  Following are the required files:

  - `api_ejb.jar`
  - `itim_api.jar`
  - `itim_common.jar`
  - `itim_server_api.jar`
  - `jlog.jar`

d. Update the `iiqIntegration-ITIM.war` file to include the updated `itim.properties` and required `ITIM .jar` files.

For example,

```
jar uvf iiqIntegration-ITIM.war WEB-INF/classes/itim.properties \
WEB-INF/lib/api_ejb.jar WEB-INF/lib/itim_api.jar \
WEB-INF/lib/itim_common.jar WEB-INF/lib/itim_server_api.jar \
WEB-INF/lib/jlog.jar
```

2. Install the IdentityIQ **ITIM Integration Web Application**: In the WebSphere Administrative Console, navigate to Enterprise Applications and select **Install.**
   a. Select **iiqIntegration-ITIM.war** as the application to install and type **iiqitim** as the context root.
   b. Continue through the rest of the installation wizard accepting the defaults.
   c. When completed, click **Save** to save the changes to the master configuration.
3. **Setup the Integration Config**: The **IntegrationConfig** object holds information about how to connect IdentityIQ to ITIM and all of the configuration requirements for various functions. ITIM supports dual role push mode, which means that both detectable and assignable roles can be used. An example can be found in the ITIM integration folder within your IdentityIQ installation directory in the `$INSTALLDIR/integration/ITIM/server/ resources/exampleIntegration.xml` directory.

The main properties that need to be set are:

- **executor**: sailpoint.integration.itim.ITIMIntegrationExecutor
- **ApplicationRef**: The reference to the ITIM application
- **Attributes**=> **URL**: The URL to the IIQ web service on the ITIM server. For example, *https://myitim.example.com:9080/iiqitim/resources*
  Note: It is highly recommended that you use SSL when transmitting sensitive electronic information.
- **Attributes**=> **username**: ITIM user's credentials used for basic HTTP authentication.
- **Attributes**=> **password**: ITIM user's password used for basic HTTP authentication.
- **ManagedResources map**: Mappings of local IdentityIQ applications to ITIM services, including mappings of local IdentityIQ attribute names to ITIM service attribute names.

For more information, see Appendix: A: Common Identity Management Integration Configuration**.**

4. **Verify**: Be certain that the integration has been installed correctly by using the ping command in the integration console. If successful, this should respond and list version information about the ITIM jar files that were put into the **iiqIntegration-ITIM.war** file. Compare this version information against the version of the ITIM server to ensure correct operation.
5. **Role Requests**: Set the roleSyncStyle to **dual** in the IntegrationConfig file as follows:
   ```
   <IntegrationConfig executor="sailpoint.integration.itim.ITIMIntegrationExecutor"
   name="ITIM 5.1 Integration" roleSyncStyle="dual">
   ```
   Other than this, the role should be assignable (for example, a business role) and the name has to match the name of the role in ITIM.

Chapter 6: SailPoint BMC Remedy Service Desk Service Integration Module

The following topics are discussed in this chapter:

Overview 27
Supported features 27
Supported platforms 27
Pre-requisites 28
Basic configuration 28
Configuring BMC Remedy AR System for IdentityIQ Integration 29
Configuring IdentityIQ for BMC Remedy Action Request System Integration 31
Sample scenario 35
Known/Open Issues 36

Overview

The integration between SailPoint and BMC Remedy Service Desk enables customers to create incidents and change requests in BMC Remedy Service Desk for the configured operations (for example, Change Password, Request Entitlement and so on) for the configured application. The seamless integration of SailPoint and BMC Remedy Service Desk Service Integration Module eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

Supported features

BMC Remedy Service Integration Module supports the following features:

- creating ticket for all provisioning operations that can be performed on Target Application accounts
- getting the status of the created tickets

Supported platforms

IdentityIQ supports the following versions of BMC Remedy AR System:

- BMC Remedy AR System 8.1.00
- BMC Remedy AR System 8.0.00
- BMC Remedy AR System 7.5.00 patch 001
- BMC Remedy AR System 7.1.00

Pre-requisites

- BMC Remedy Change Management Application must be installed.
- Ensure that the following softwares are operating correctly:
  - BMC Remedy AR System
  - BMC Remedy Change Management Application

Basic configuration

The integrated solution speeds the detection and remediation of identity management issues that increase the risk of compliance violations or security breaches, such as orphaned accounts, policy violations, and inappropriate access privileges. Organizations can take advantage of a centralized approach spanning thousands of users and hundreds of resources to strengthen IT controls and provide proof of compliance to auditors and executive management. The seamless integration of SailPoint and BMC Remedy eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

For any IT resources managed by BMC Remedy Service Desk, IdentityIQ automatically creates a trouble ticket within Remedy Service Desk, passing along all relevant identity data and reviewer comments to populate the ticket.

To ensure revocation requests get delivered and implemented, IdentityIQ manages all remediation and revocation requests within a guaranteed delivery model.

To determine the status of user accounts, IdentityIQ performs closed-loop audits on remediation requests and compares the actual state of user privileges with the original change request. If the request is still open, an alert will be sent to the reviewer for prompt action and closure.

The integration itself has been designed to be quick to install and easy to use. It makes use of Web Services via the Remedy Mid Tier to broker communications between the SailPoint server and the AR System server. On the backside of a user recertification, policy remediation action or access request action, the IdentityIQ server will direct provisioning and service desk requests to the configured implementers. Based on the IntegrationConfig configured for each target application, service desk request are issues to a given remediation/implementation point. Once the IntegrationConfig for Remedy has been loaded into the IdentityIQ server, all change/remediation actions result in the creation of new service desk request.

At the completion of the change control cycle within IdentityIQ, an “Open Ticket” request is made over the appropriate SOAP channel to the Mid Tier. From here change request tickets are opened and the new ticket number is returned to IdentityIQ. The schema for the service request is defined in the IntegrationConfig and allows for the flexibility to transfer complete details on the service desk request. The default settings will create a basic ticket as shown in the following figure (Figure 1—Change request).

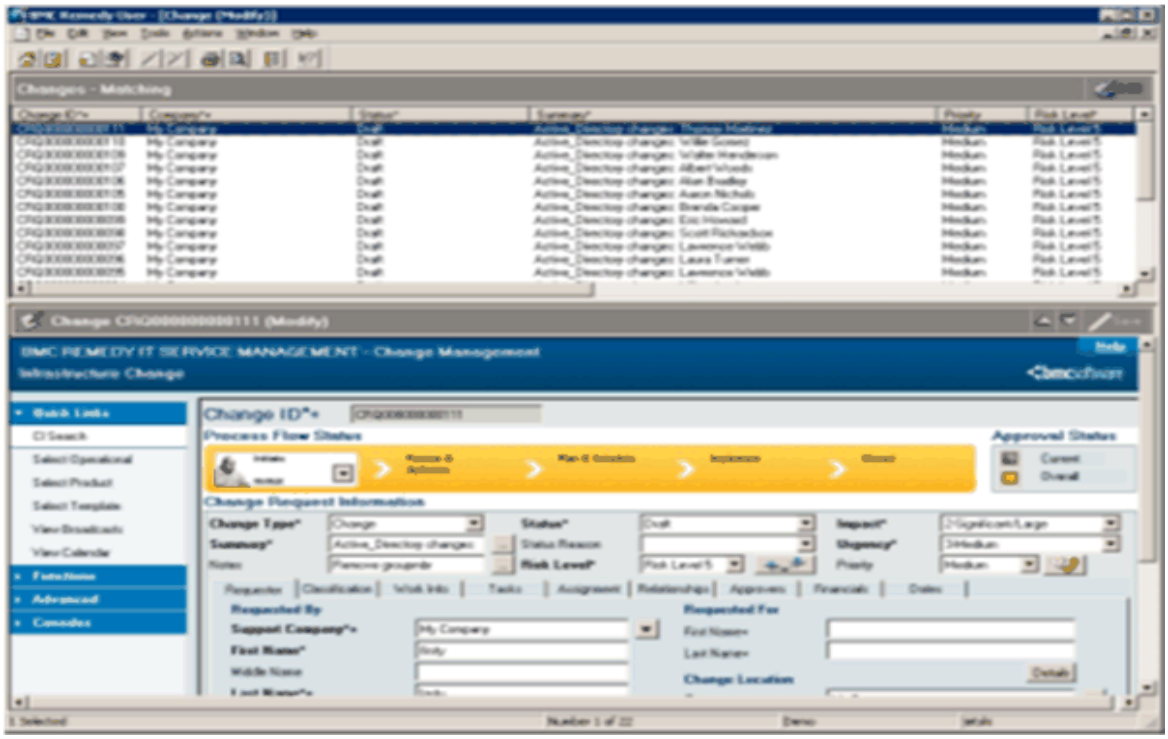


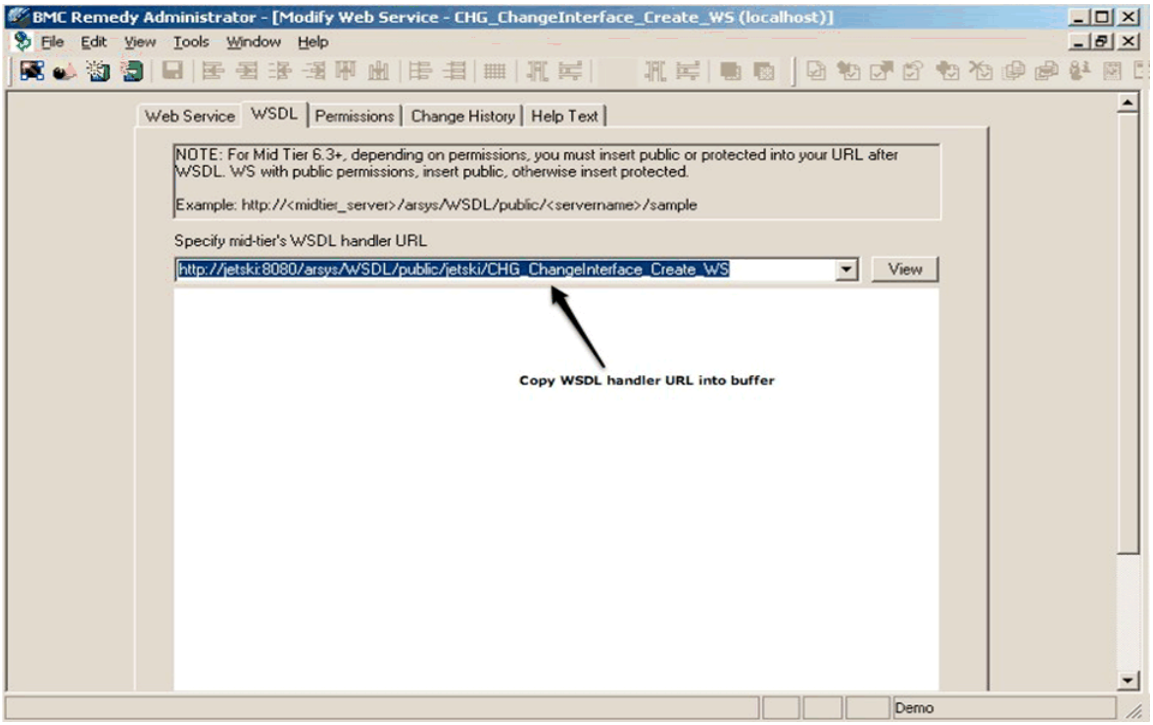
Figure 1—Change request

Configuring BMC Remedy AR System for IdentityIQ Integration

This section provides the required information for configuring IdentityIQ to integrate with BMC Remedy Action Request System (AR System). This integration enables IdentityIQ to create Change Management tickets for requested revocations, track ticket numbers in association with revocation tasks, and update IdentityIQ with the status of current Change Management tickets.

The following steps should be performed to modify the default Remedy integration configuration for a specific BMC Remedy application instance.

1. Confirm the default Remedy Change Management Application Web Services exist. This is done by launching the BMC Remedy Administrator, expanding the appropriate server object and clicking on the “Web Services” object.
2. Next, obtain the environment-specific Web Service “endpoint” by performing the following steps:
    a. Double-click on the Web Service and select the WSDL tab. Copy the WSDL handler URL into your buffer (For example, Ctrl-C)



b. With a web browser, visit the WSDL URL for the web service by entering the URL into the browser address field and pressing return.

c. Search for **soap:address location=** to find the endpoint URL. Copy this value. It will be used to replace the endpoint URL in the default IdentityIQ Remedy IntegrationConfig object.

```
– <wsdl:port binding="s0:CHG_ChangeInterface_Create_WSSoapBinding" name="CHG_ChangeInterface_Create_WSSoap">
    <soap:address location="http://jetski:8080/arsys/services/ARService?server=jetski&webService=CHG_ChangeInterface_Create_WS"/>
  </wsdl:port>
```

d. Review the Create InputMap section of the WSDL to understand the fields available for population through the Web Service. These fields should correspond to the fields listed in the <soapenv:Body> section of the default IdentityIQ IntegrationConfig object

3. Once you are familiar with the WSDL, modify the default IdentityIQ Remedy integration using the information collected about the web service.

   a. In the <IntegrationConfig> element of the integration configuration, modify the **username** and **password** entries in the attributes map to contain the credentials required for authentication to the web service.

   b. In the <IntegrationConfig> element of the integration configuration, modify the provision entry of the Attributes map by setting the endpoint, and, if necessary, the namespace, the prefix, the responseElement, and the soapMessage attributes (the default values: IdentityIQ Remedy IntegrationConfig):

      i. Set the value for endpoint to the value located in the WSDL earlier.

      Note: The value in the IdentityIQ integration configuration must be a valid HTTP URL and have any special characters escaped. The most common change that must be made is to replace all & symbols with &

      ii. The value for namespace comes from the **targetNamespace** attribute of the **xsd:schema** element in the WSDL.

iii. The value for prefix is the prefix of the XML elements that will be contained in the SOAP response sent by the mid tier server.

iv. The value for responseElement should be the ARS form field that corresponds to the id of the form that the web service creates.

v. The value for soapMessage should be the SOAP message body that IdentityIQ will send to ARS. The exact format of this message is a function of the form that is published as described by the form's WSDL. The XML elements in the **soapenv:Body** element should be changed to match the ARS form fields for the published web service. Each required ARS form field must have an element in the SOAP message. The value can be fixed or can be a variable that will be substituted using IdentityIQ's Velocity templating

The information in the reference section above show the variables that are provided and the example integration configuration provides examples of how they are used.

Configuring IdentityIQ for BMC Remedy Action Request System Integration

This is intended as an introduction to the configuration needed to integrate **IdentityIQ** with the BMC Remedy Action Request System. This integration enables **IdentityIQ** to interact with many of the product solutions that are built on top of the AR System Server including BMC Remedy Change Management, BMC Remedy IT Service Management Suite, and BMC Remedy Service Desk.

Prerequisites

When running a ServiceNow integration on an application server using Java 6, the JRE must be directed to use the SOAP/SAAJ implementation provided by Axis2. To do this, add the following Java options to the application server:

- `-Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl`
- `-Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl`
- `-Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl`

Without these options, a ClassCastException error or Signature creation failed error will occur when the Remedy integration is used.

BMC Remedy Action Request System Integration

SailPoint provides a default Remedy integration configuration. This configuration implements the stock integration between IdentityIQ and the Remedy Change Management Application to fulfill creation of tickets based on IdentityIQ access certification remediation events.

The default configuration is located in `iiqHome/WEB-INF/config/remedy-Integration.xml` directory, where *iiqHome* is the location where IdentityIQ was installed.

For BMC Remedy AR System versions 7.5.00 patch 001, 8.0 and 8.1, an additional sample xml file is located in *`iiqHome`*`/WEB-INF/config/remedyV8-integration.xml` directory.

This section explains the various entries that are specific for this integration. For more information of the entries in the `IntegrationConfig` file, see Appendix A: Common Identity Management Integration Configuration.

The integration configuration must include the following entries:

- **endpoint**: URL to the web service
- **namespace**: namespace of the XML returned by the web service
- **prefix**: prefix associated with the namespace

The integration configuration includes the following entries if the web service side of the integration is configured for authentication using the SOAP authentication specifications:

- username
- password
- authentication
- locale
- timeZone
- statusMap

The integration configuration includes the following entries if the http authentication is configured:

- basicAuthType: if http authentication is configured the value of basicAuthType is true.
- httpUserName
- httpUserPass

User must modify remedy integration configuration file with the following entries to create incident in BMC Remedy Action Request System:

- endpoint
- responseElement key
- SOAP envelop and body details
- status mapping

The web services and authentication entries are consumed by configuration entries for each web service. They can be positioned either within the configuration entries themselves or as children of the Attributes element. Entries that are children of the Attributes element can be thought of as global values, while entries within the configuration entities can be thought of as local.

For example, if both entries share the same authentication credentials, those credentials might be placed in the Attributes element as peers of the configuration entries and the integration code searches the parent entry for the credentials if they are not found in the configuration entries. Conversely, if the configuration entries have different endpoints (are handled by separate web services), each configuration entry specifies the endpoint of the web service to call and any value outside of the configuration entry is ignored.

There are two supported configuration entries for integration with Remedy. These entries are children of the integration Attributes element:

- getRequestStatus
- provision

The values of each are Map elements containing key/value pairings of the configuration data. They contain the specific data needed by the getRequestStatus()and provision() methods of the IdentityIQ integration executor and correspond to Remedy Web Service methods.

The **getRequestStatus** and **provision** entries contain the following entries:

- **soapMessage** (required): full XML template of the entire SOAP envelope that is sent to the web service. The integration code first runs this template through Apache's Velocity template engine to provide the data needed by the web service.
- **responseElement** (required): name of the element containing the results of the web service call (for example, the element containing the ticket number opened by the web service in response to the call from IdentityIQ).
- **statusMap** (optional, see “Sample getRequestStatus entry” on page 33 for an example)
- **username** (optional)
- **password** (optional)
- **authentication** (optional)
- **locale** (optional)
- **timeZone** (optional)
- **endpoint** (optional)
- **namespace** (optional)
- **prefix** (optional)

Before a template is sent to the web service, it is processed by the **Velocity template engine**. The integration code provides different data objects to Velocity for evaluation based on the integration method.

The **provision** call passes the following objects to Velocity:

- **config**: the integration configuration for provision, represented as a Map
- **provisioningPlan**: the data model of the provision request

The **getRequestStatus** call passes the following objects to Velocity:

- **config**: the integration configuration for getRequestStatus, represented as a Map
- **requestID**: the string ID of the request whose status is being queried

Both calls have access to a timestamp variable containing a current Date object and a dateFormatter object. The dateFormatter is built using an optional dateFormat attribute from the **config** object. If the dateFormat attribute does not exist, the formatter defaults to the pattern EEE, d MMM yyyy HH:mm:ss z.

Sample getRequestStatus entry

Note: The entries contained in the Map are the only required entries. Any authentication information required by this integration is inherited from the parent Attributes element.

```
<entry key="getRequestStatus">
  <value>
    <Map>
        <entry key="responseElement" value="Status"/>
        <entry key="soapMessage">
            <!-- XML template - DO NOT add line breaks before the CDATA! -->
            <value><String><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
#if ($config.username)
<soapenv:Header>
<ns1:AuthenticationInfo xmlns:ns1="urn:AuthenticationInfo">
                 <ns1:userName>$config.username</ns1:userName>
                 <ns1:password>$config.password</ns1:password>
        #if ($config.authentication)
```

```
                <ns1:authorization>$config.authentication</ns1:password>
#end
#if ($config.locale)
                <ns1:locale>$config.locale</ns1:password>
#end
#if ($config.timeZone)
                <ns1:timeZone>$config.timeZone</ns1:password>
#end
</ns1:AuthenticationInfo>
</soapenv:Header>
#end
<soapenv:Body>
       <iiq:Get xmlns:iiq="urn:GetAgreementWebService">
          <iiq:Issue_ID>$requestID</iiq:Issue_ID>
       </iiq:Get>
     </soapenv:Body>
</soapenv:Envelope>
]]>
                   </value>
                </entry>
            </Map>
         </value>
</entry>
```

Sample provision entry

Note: This Map contains its own web services information. Any authentication information required by this integration is inherited from the parent Attributes element.

```
<entry key="provision">
       <value>
           <Map>
                   <entry key="endpoint"
value="http://my.server.com:8080/path/to/WS"/>
                   <entry key="namespace" value="urn:openTicketWebService"/>
                   <entry key="prefix" value="xyz"/>
                   <entry key="responseElement" value="Issue_ID"/>
                   <entry key="soapMessage">
                         <!-- XML template - DO NOT add line breaks before the CDATA!
-->
                         <value><String><![CDATA[<?xml version="1.0"
encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
#if ($config.username)
<soapenv:Header>
<ns1:AuthenticationInfo xmlns:ns1="urn:AuthenticationInfo">
                 <ns1:userName>$config.username</ns1:userName>
                 <ns1:password>$config.password</ns1:password>
#if ($config.authentication)
                 <ns1:authorization>$config.authentication</ns1:password>
#end
#if ($config.locale)
                 <ns1:locale>$config.locale</ns1:password>
#end
#if ($config.timeZone)
                 <ns1:timeZone>$config.timeZone</ns1:password>
```

```
#end
</ns1:AuthenticationInfo>
</soapenv:Header>
#end
<soapenv:Body>
     <iiq:Get xmlns:iiq="urn:openTicketWebService">
         <iiq:Submitter>
         #foreach ($req in $provisionPlan.requesters)
              $req.name
         #end
         </iiq:Submitter>
         </iiq:SubmitDate>$timestamp</iiq:SubmitDate>
         <iiq:Summary>
               Remediation request from IIQ
         </iiq:Summary>
         <iiq:Description>
               Remove Active Directory for $provisionPlan.identity.fullname
         </iiq:Description>
         <iiq:Issue_ID>$requestID</iiq:Issue_ID>
       </iiq:Get>
     </soapenv:Body>
</soapenv:Envelope>
]]>
             </value>
          </entry>
       </Map>
    </value>
</entry>
```

Sample statusMap entry

The **noMappingFromWS** entries are placeholders as there are no results from the web service corresponding to those IdentityIQ result codes.

```
<entry key="statusMap">
      <value>
          <Map>
              <entry key="Closed" value="success" />
              <entry key="Rejected" value="failure" />
              <entry key="Draft" value="inProcess" />
              <entry key="Pending" value="inProcess" />
              <entry key="noMappingFromWS" value="retry" />
              <entry key="noMappingFromWS" value="warning" />
          </Map>
      </value>
</entry>
```

Sample scenario

The sample integration scenario is built around a sample system. In the sample scenario SailPoint (IIQ) will be issuing a change request to BMC Remedy Change Management (RCM) based on the results of a scheduled user entitlement and access review. As a result of remediation actions in this account recertification process, IdentityIQ will open change requests to control the flow of the manual remediation process.

Scenario

1. The ComplianceManager1 schedules an access review for a business critical application:
    a. The certification is scheduled and assigned to ApplicationOwner1.
    b. ApplicationOwner1 receives an email with a link to the online certification process as scheduled. The link is followed to the open certification.
    c. ApplicationOwner1 decides that GroupA on system LDAP should be removed.
    d. ApplicationOwner1 decides that RoleA on system RDBMS should be removed.
    e. ApplicationOwner1 completes the certification and signs off the process.
2. IdentityIQ evaluates the provisioning plan to enact the remediation requests from the certification:
    a. IdentityIQ policy describes the integration execution path for LDAP as being via an automated provisioning system.
    b. IdentityIQ policy describes the integration execution path for RDBMS as being via an automated RCM integration.
3. IdentityIQ creates a service request in RCM:
    a. IdentityIQ uses the **provision** interface to open a service request within Remedy, passing in details of the changes required to the RDBMS system.
    b. RCM responds with the service request number.
    c. IdentityIQ stores the service request number for later audit and review.

Known/Open Issues

Following is the known/open issue of BMC Remedy Service Integration Module:

- Only one ticket will be created for all account requests in the provisioning plan

Chapter 7: SailPoint ServiceNow Service Integration Module

The following topics are discussed in this chapter:

Overview 37
Supported features 37
Supported platforms 37
Pre-requisites 38
Basic configuration 39
Configuring ServiceNow for IdentityIQ Integration 41
ServiceNow Integration 41
Upgrade 46
Sample scenario 46
Troubleshooting 48

Overview

The integration between SailPoint and ServiceNow enables customers to create incidents and change requests in ServiceNow for the configured operations (for example, Change Password, Request Entitlement and so on) for the configured application. The seamless integration of SailPoint and ServiceNow Service Integration Module eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

Supported features

ServiceNow Service Integration Module supports the following features:

- creating ticket for all provisioning operations that can be performed on Target Application accounts
- getting the status of the created tickets

Supported platforms

SailPoint Integration for ServiceNow Service Integration Module supports the following ServiceNow versions:

- Eureka
- Dublin
- Calgary

Pre-requisites

- ServiceNow Instance should be up and running.

  The Administrator must be assigned `iiq_servicenow_sim_role` role for the required permissions.
- Apply the ServiceNow Service Integration Module update set:
  a. Login to the SailPoint Compass Community using the following link:

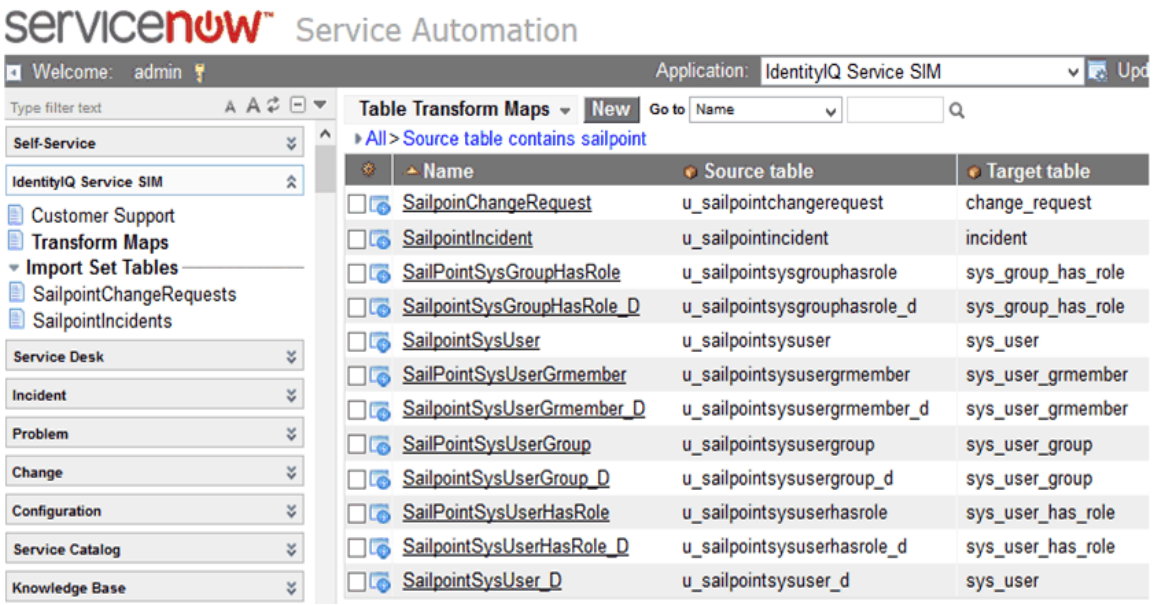
     https://community.sailpoint.com/groups/product-download-center
  b. On the Product Download Center page, click on IdentityIQ version 6.4.

     Based on the required version of ServiceNow, copy the relevant update set from the following respective files:

ServiceNow version	Update Sets
`iiqIntegrations-ServiceNow-calgary_6.4.zip`	`IdentityIQServiceNowSIM.v1.2_calgary_6.4.xml`
`iiqIntegrations-ServiceNow-dublin_6.4.zip`	`IdentityIQServiceNowSIM.v1.3_dublin_6.4.xml`
`iiqIntegrations-ServiceNow-eureka_6.4.zip`	`IdentityIQServiceNowSIM.v1.3_eureka_6.4.xml`

  c. Import relevant update set in ServiceNow instance. For more information and guidelines on usage of the update set, refer to the following wiki link:
     http://wiki.servicenow.com/

  Verify if the update set was successfully applied as displayed in the following figure:



- When running a ServiceNow integration on an application server using Java 6, the JRE must be directed to use the SOAP/SAAJ implementation provided by Axis2. To do this, add the following Java options to the application server:
  - `-Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl`
  - `-Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl`
  - `-Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl`

  Without these options, a ClassCastException error or Signature creation failed error will occur when the ServiceNow integration is used.
- **Ensure that the ServiceNow is up and running**: To ensure that ServiceNow is up and web services component is running, access the following WSDL:

  https://<ServiceNowHost>/incident.do?WSDL

  Alternatively, use a Soap UI tool to submit a simple request (for example, incident). For convenience you can use the basic authentication mechanism for authorization with SOAP UI tool to confirm that the web service layer is functional.
- **Ensure that `ServiceNowIntegrationExecutor` is being called**: ServiceNowIntegrationExecutor class is responsible for creating and sending SOAP requests to ServiceNow. You can add a simple SystemOut statement in `ServiceNowIntegrationRule` to ensure that this is being called when a provisioning request is submitted for this integration.
- When integrating with **changeRequest**, modify `sampleServiceNowIntegrationForChangeRequest.xml` sample file and import in IdentityIQ.

  When integrating with **incident users**, modify `sampleServiceNowIntegration.xml` file and import in IdentityIQ.

Basic configuration

The integrated solution speeds the detection and remediation of identity management issues that increase the risk of compliance violations or security breaches, such as orphaned accounts, policy violations, and inappropriate access privileges. Organizations can take advantage of a centralized approach spanning thousands of users and hundreds of resources to strengthen IT controls and provide proof of compliance to auditors and executive management. The seamless integration of SailPoint and ServiceNow eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

For any IT resources managed by ServiceNow Service Desk, IdentityIQ automatically creates a trouble ticket within ServiceNow Service Desk, passing along all relevant identity data and reviewer comments to populate the ticket.

To ensure revocation requests get delivered and implemented, IdentityIQ manages all remediation and revocation requests within a guaranteed delivery model.

To determine the status of user accounts, IdentityIQ performs closed-loop audits on remediation requests and compares the actual state of user privileges with the original change request. If the request is still open, an alert will be sent to the reviewer for prompt action and closure.

The integration itself has been designed to be quick to install and easy to use. It makes use of Web Services for communications between the SailPoint server and the ServiceNow. On the backside of a user recertification, policy remediation action or access request action, the IdentityIQ server will direct provisioning and service desk requests to the configured implementers. Based on the `IntegrationConfig` configured for each target application, service desk request are issues to a given remediation/implementation point. Once the

`IntegrationConfig` file for ServiceNow has been loaded into the IdentityIQ server, all change/remediation actions result in the creation of new service desk request as shown in Figure 1—Basic configuration.

The SailPoint IdentityIQ Integration for ServiceNow Service Integration Module generates incident for Provisioning requests. These incidents are generated on the `incident` table or change requests on the `change_request` table defined by customer using Imports Sets and Transform Maps. The module fetches the status of ticket by using the direct web services of target tables that is, `incident` or `change_request` and updates the SailPoint IdentityIQ database with the status.

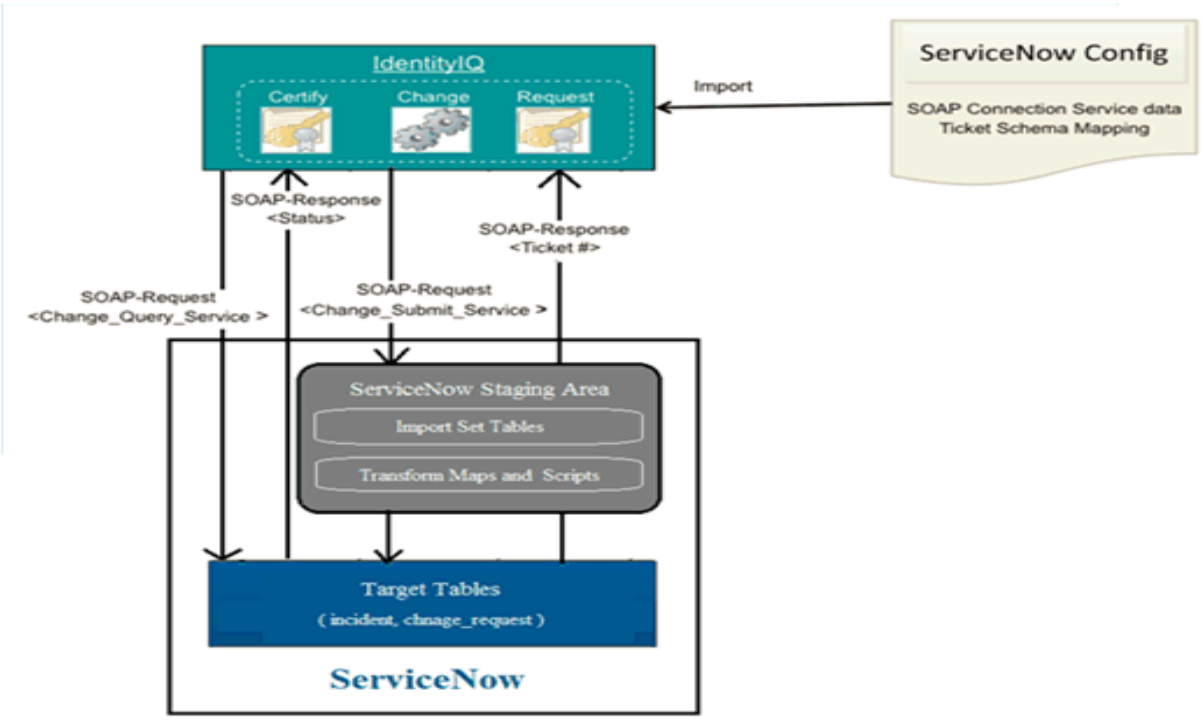


Figure 1—Basic configuration

At the completion of the change control cycle within IdentityIQ, an “Open Ticket” request is made over the appropriate SOAP channel to the ServiceNow web service. From here, tickets are opened and the new ticket number is returned to IdentityIQ. The schema for the service request is defined in the `IntegrationConfig` and allows for the flexibility to transfer complete details on the service desk request. The default settings will create a basic ticket as shown in Figure 2—Service Desk request.

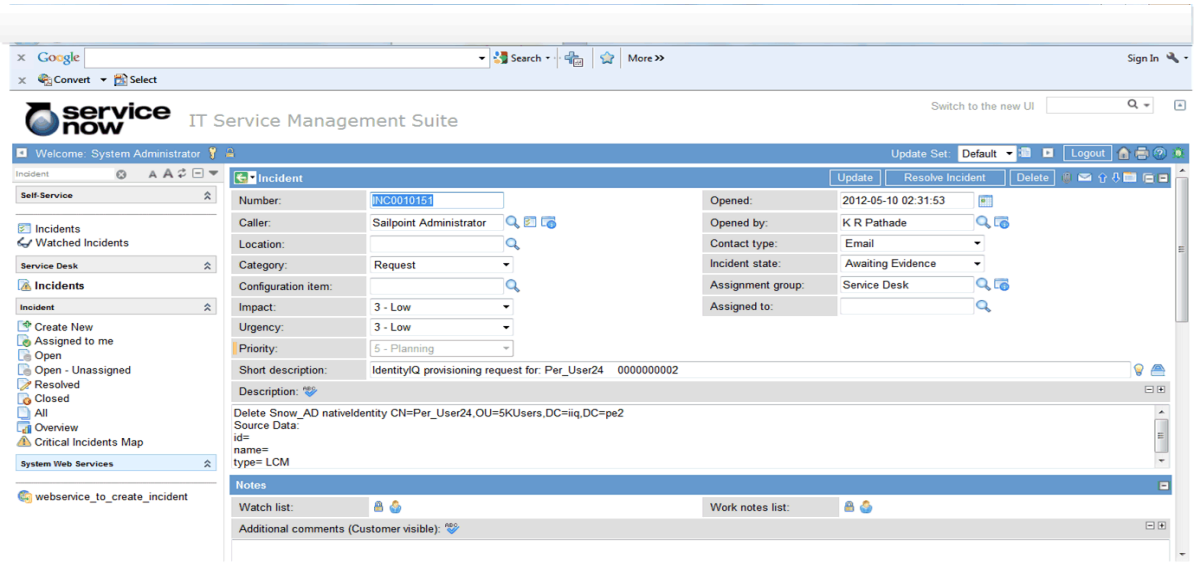


Figure 2—Service Desk request

Configuring ServiceNow for IdentityIQ Integration

This section provides the required information for configuring IdentityIQ to integrate with ServiceNow. This integration enables IdentityIQ to create tickets for requested revocations, track ticket numbers in association with revocation tasks, and update IdentityIQ with the status of current tickets.

ServiceNow Integration

This is intended as an introduction to the configuration needed to integrate IdentityIQ with the ServiceNow. This integration enables IdentityIQ to interact with ServiceNow Service Desk.

SailPoint provides a default ServiceNow configuration. This configuration implements the stock integration between IdentityIQ and the ServiceNow to fulfill creation of tickets based on IdentityIQ access certification remediation events.

The default configuration is located in `iiqHome/WEB-INF/config/sampleServiceNowintegration.xml` directory, where *iiqHome* is the location where IdentityIQ was installed.

This section explains the various entries that are specific for this integration. For more information of the entries in the `IntegrationConfig` file, see Appendix: A: Common Identity Management Integration Configuration.

The integration configuration must include the following entries:

- **endpoint**: URL to the web service
- **namespace**: namespace of the XML returned by the web service
- **prefix**: prefix associated with the namespace

The integration configuration includes the following entries if the web service side of the integration is configured for authentication using the SOAP authentication specifications:

- username (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- password (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- authentication
- locale
- timeZone
- statusMap

The integration configuration also includes the following properties if WS-Security is enabled on service-now side:

- authType
- keystorePath
- keystorePass
- keystoreType
- alias
- keyPass

For more information on enabling the WS-Security on ServiceNow side, see "Configuration required on ServiceNow side for WS-Security" on page 45.

The web services and authentication entries are consumed by configuration entries for each web service. They can be positioned either within the configuration entries themselves or as children of the Attributes element. Entries that are children of the Attributes element can be thought of as global values, while entries within the configuration entities can be thought of as local.

For example, if both entries share the same authentication credentials, those credentials might be placed in the Attributes element as peers of the configuration entries and the integration code searches the parent entry for the credentials if they are not found in the configuration entries. Conversely, if the configuration entries have different endpoints (are handled by separate web services), each configuration entry specifies the endpoint of the web service to call and any value outside of the configuration entry is ignored.

There are two supported configuration entries for integration with ServiceNow. These entries are children of the integration Attributes element:

- getRequestStatus
- provision

The values of each are Map elements containing key/value pairings of the configuration data. They contain the specific data needed by the **getRequestStatus()** and **provision()** methods of the IdentityIQ integration executor and correspond to ServiceNow Web Service methods.

The **getRequestStatus** and **provision** entries contain the following entries:

- **soapMessage** (required): full XML template of the entire SOAP envelope that is sent to the web service. The integration code first runs this template through Apache's Velocity template engine to provide the data needed by the web service.
- **responseElement** (required): name of the element containing the results of the web service call (for example, the element containing the ticket number opened by the web service in response to the call from IdentityIQ).
- **statusMap** (optional, see "Sample XML files for Incident and Change Request" on page 43 for an example)
- **username** (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- **password** (Can be blank if **Require basic authorization for incoming SOAP requests** is not selected on ServiceNow side)
- **authentication**
- **locale** (optional)
- **timeZone** (optional)
- **endpoint** (optional)
- **namespace** (optional)
- **prefix** (optional)
- **authType** (optional): Use **"**WS-Security**"** if WS Security is enabled on service-now side. Otherwise leave it blank.
- **keystorePath** (optional): Full path of keystore
- **keystorePass** (optional): Password of keystore
- **keystoreType** (optional): Type of keystore. For example, jks
- **alias** (optional): The alias of certificate in keystore
- **keyPass** (optional): The password of alias

Before a template is sent to the web service, it is processed by the **Velocity template engine**. The integration code provides different data objects to Velocity for evaluation based on the integration method.

The **provision** call passes the following objects to Velocity:

- **config**: the integration configuration for provision, represented as a Map
- **provisioningPlan**: the data model of the provision request

The **getRequestStatus** call passes the following objects to Velocity:

- **config**: the integration configuration for getRequestStatus, represented as a Map
- **requestID**: the string ID of the request whose status is being queried

Both calls have access to a timestamp variable containing a current Date object and a dateFormatter object. The `dateFormatter` is built using an optional **dateFormat** attribute from the **config** object. If the dateFormat attribute does not exist, the formatter defaults to the pattern EEE, d MMM yyyy HH:mm:ss z.

Sample XML files for Incident and Change Request

The entries contained in the Map are the only required entries. Any authentication information required by this integration is inherited from the parent Attributes element.

For more information and examples of the sample files, see the following sample files:

- `sampleServiceNowIntegration.xml`
- `sampleServiceNowIntegrationForChangeRequest.xml`

If any changes required in the mapping, change the default value/key values in “statusMap” and “statusMapCloserCode” as mentioned in the following table:

Entry key	Values
statusMap	
1	queued
2	queued
3	queued
4	queued
5	queued
6	committed
7	committed
statusMapCloserCode	
Solved (Work Around)	committed
Solved (Permanently)	committed
Solved Remotely (Work Around)	committed
Solved Remotely (Permanently)	committed
Closed/Resolved by Caller	committed
Not Solved (Not Reproducible)	failed
Not Solved (Too Costly)	failed

Configuration procedure

The following steps should be performed to modify the default ServiceNow integration configuration for a specific ServiceNow instance.

1. Obtain the environment-specific Web Service “endpoint”, for example, https://demo.service-now.com/incident.do?SOAP
   Either a web service can be created a web service pointing to system table can be used, for example, https://demo.service-now.com/incident.do?SOAP
2. Once you are familiar with the WSDL, modify the default IdentityIQ ServiceNow configuration using the information collected about the web service.
   a. In the <IntegrationConfig> element of the integration configuration, modify the **username** and **password** entries in the attributes map to contain the credentials required for authentication to the web service.
   b. If you have enabled WS-Security on ServiceNow side, modify entries for **authType**, **keystorePath**, **keystorePass**, **keystoreType**, **alias**, **keyPass** to contain keystore related details.
   c. In the <IntegrationConfig> element of the integration configuration, modify the provision entry of

the Attributes map by setting the endpoint, and, if necessary, the namespace, the prefix, the responseElement, and the soapMessage attributes (the default values: IdentityIQ ServiceNow IntegrationConfig):

i. Set the value for endpoint to the value located in the WSDL earlier.

Note: The value in the IdentityIQ integration configuration must be a valid HTTP URL and have any special characters escaped. The most common change that must be made is to replace all & symbols with &

ii. The value for namespace comes from the **targetNamespace** attribute of the **xsd:schema** element in the WSDL.

iii. The value for prefix is the prefix of the XML elements that will be contained in the SOAP response.

iv. The value for responseElement should be the ServiceNow form field that corresponds to the id of the form that the web service creates.

v. The value for soapMessage should be the SOAP message body that IdentityIQ will send to ServiceNow. The exact format of this message is a function of the form that is published as described by the form's WSDL. The XML elements in the **soapenv:Body** element should be changed to match the ServiceNow form fields for the published web service. Each required ServiceNow form field must have an element in the SOAP message. The value can be fixed or can be a variable that will be substituted using IdentityIQ's Velocity templating

The information in the reference section above show the variables that are provided and the example integration configuration provides examples of how they are used.

Configuration required on ServiceNow side for WS-Security

Perform the following steps to enable WS-Security on ServiceNow side:

1. Login to service-now instance with user having access to do system changes for example, admin.
2. Navigate to **System Definition => Certificates** and click on **New** button.
3. Enter some name for the certificate.
4. In command prompt, navigate to directory where the certificate is located.
5. Enter the following command, where **mycert.pem** is the name of the certificate:
   type mycert.pem
   The following similar output is displayed:
   ```
   -----BEGIN CERTIFICATE-----
   MIIDOTCCAiECBE/6tzswDQYJKoZIhvcNAQEEBQAwYTELMAkGA1UEBhMCSU4xCzAJBgNVBAgTAk1I
   ...
   ...
   -----END CERTIFICATE-----
   ```
6. Copy all above content (including Begin and End Certificate lines) and navigate to service-now and paste the above contents into **PEM Certificate** field.
7. Save the certificate.
8. Navigate to System **Web Services =>WS Security Profiles** and click on **New**.

9. In **Type** field, select X509 and in **X509 Certificate** field, select certificate which we uploaded.
10. Select bind session checkbox.
11. In Run as user field, select name of user on behalf of whom, you want to execute web-services for example, System Administrator.
12. Click on **Submit** button.
13. Navigate to **System Web Services => Properties**.
14. Select the **Require WS-Security header verification for all incoming SOAP requests** check box and save it.

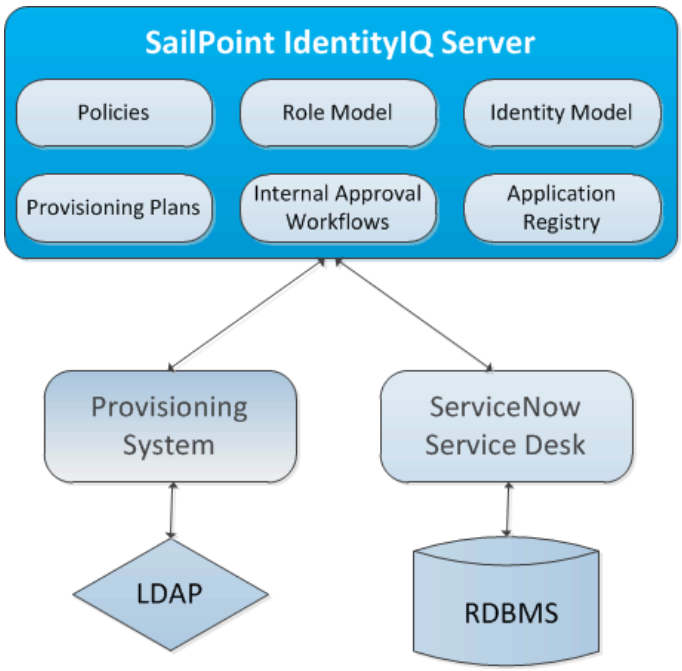
Upgrade

Perform the following procedure for upgrading IdentityIQ ServiceNow Service Integration Module from version 6.3 to 6.4:

1. IdentityIQ Service SIM update set must be applied on ServiceNow instance.
   For more information, see “Pre-requisites” on page 38.
2. The Administrator must be assigned the `iiq_servicenow_sim_role` role.
   For more information, see “Pre-requisites” on page 38.
3. In the IdentityIQ side the `sampleServiceNowIntegration` xml files must be re-imported after performing the corresponding changes.
   For more information about the `ServiceNowIntegration` xml file, see “ServiceNow Integration” on page 41.

Sample scenario

The sample integration scenario is built around a sample system as shown in the following figure:



In the sample scenario SailPoint (IIQ) will be issuing change request to ServiceNow based on the results of a scheduled user entitlement and access review. As a result of remediation actions in this account recertification process, IdentityIQ will open incident to control the flow of the manual remediation process.

Scenario

1. The ComplianceManager1 schedules an access review for a business critical application:
    a. The certification is scheduled and assigned to ApplicationOwner1.
    b. ApplicationOwner1 receives an email with a link to the online certification process as scheduled. The link is followed to the open certification.
    c. ApplicationOwner1 decides that GroupA on system LDAP should be removed.
    d. ApplicationOwner1 decides that RoleA on system RDBMS should be removed.
    e. ApplicationOwner1 completes the certification and signs off the process.
2. IdentityIQ evaluates the provisioning plan to enact the remediation requests from the certification:
    a. IdentityIQ policy describes the integration execution path for LDAP as being via an automated provisioning system.
    b. IdentityIQ policy describes the integration execution path for RDBMS as being via an automated ServiceNow integration.
3. IdentityIQ creates a service request in ServiceNow:
    a. IdentityIQ uses the **provision** interface to open a service request within ServiceNow, passing in details of the changes required to the RDBMS system.
    b. ServiceNow responds with the service request number.
    c. IdentityIQ stores the service request number for later audit and review.

Troubleshooting

This section provides the resolutions for the following errors that may be encountered while setting up and configuring ServiceNow Service integration Module.

1 - 'Authorization Required' error messages

The following type of error messages appear when the authorization data is not sent to ServiceNow:

```
Caused by: org.apache.axis2.AxisFault: Transport error: 401 Error:
Authorization Required
```

Resolution: Verify the procedure to configure appropriate authorization mechanism. For more information on the procedure, see "Configuration procedure" on page 44.

2 - Document Type Declaration (DTD) parsing errors

The DTD parsing errors appear when the following JVM arguments are not defined for your application server:

- `-Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl`
- `-Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl`
- `-Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl`

Resolution: Ensure that the application is pointing to the correct java runtime (that is, 1.6) and the above mentioned JVM arguments are defined for application server.

Note: Enter the following command to enable log4j tracing on this component: `log4j.logger.sailpoint.integration.servicenow =debug`, file

3 - For certificate based authentication the IdentityIQ Server and ServiceNow instance must have correct time set.

For certificate based authentication, ensure that the IdentityIQ Server and ServiceNow instance have the correct date, time, and timezone set.

4 - When the Test Connection fails error messages are displayed in IdentityIQ and log file of WebSphere

The following error message is displayed in IdentityIQ:

```
Unable to engage module: rampart
```

The following error message is displayed in the log file of WebSphere:

```
ERROR WebContainer : apache.axis2.deployment.ModuleDeployer:113 - The
rampart-1.6.1.mar module, which is not valid, caused Could not initialize
class org.apache.axis2.deployment.util.TempFileManager
```

Resolution: If the above error message is displayed in the log file of WebSphere, set the temporary directory in **Generic JVM arguments** of **Java Virtual Machine** by setting the following variable:

```
-Djava.io.tmpdir=<FullPathOfTempDir>
```

Note: **Ensure that the UNIX user where WebSphere is installed should be the owner of the temporary directory.**

5 - Fields on ServiceNow side are not completed displayed.

The value of the fields in Incident/Change Request tickets on ServiceNow side are not completed displayed but truncated.

Resolution: If the value of the fields in the Incident / Change Request tickets are displayed as truncated, then increase the **Max length** values for the respective **Column name** of `u_sailpointincident` / `u_sailpointchangerequest` tables.

For example, u_description (Decription) to 4,000

u_short_description (Short decription) to 200

Chapter 8: SailPoint BMC Remedy Service Request Management Adapter

The following topics are discussed in this chapter:

Overview 51
Supported platforms 52
Configuring BMC Remedy Service Request Management for IdentityIQ Integration 52
Install 52
Configure 52
Verify 54
Troubleshooting 54

Overview

The SailPoint BMC Remedy Service Request Management Adapter enables the use of Lifecycle Manager within Service Request Management (SRM) user interfaces.

The inclusion of the SailPoint BMC Remedy Service Request Management Adapter provides the following functionality within SRM:

- Enable, Unlock Accounts
- Request Entitlements
- Request IdentityIQ Roles
- Request Password Changes

Note: Any approvals or role request provisioning form configurations must be performed within the IdentityIQ interface.

When IdentityIQ receives a request from SRM, approvals, forms or other business processes that take place in IdentityIQ are visible through the request system's ticket updates. By default SRM has visibility to the following IdentityIQ business processes:

- **Process Started** - Ticket opened.
- **Approvals** - Tickets are updated as approvals take place.
- **Provisioning Performed** - Tickets are updated after provisioning completes.
- **Process Complete** - Notification is sent when the process ends for any reason. The notice includes information including errors, events, and overall status.

This chapter provides the installation and configuration steps to deploy the SailPoint Adapter into an existing infrastructure running BMC Remedy Service Request Management. This chapter is intended for SRM and IdentityIQ System Administrators and assumes a high degree of technical knowledge of these systems.

Supported platforms

SailPoint Adapter supports BMC Remedy Action Request System version 7.6.04 of BMC Identity Management Suite.

Configuring BMC Remedy Service Request Management for IdentityIQ Integration

This section provides the required information for configuring IdentityIQ to integrate with BMC Remedy Service Request Management.

Install

1. Copy the `IIQ_IQ6.4.00.AZ` file to a folder on the BMC Remedy Action Request (AR) server. This file resides in the following location where *`installDir`* represents the directory in which you expanded the IdentityIQ product from the media with which it was delivered:

   `installDir``/integration/BMCIRM/IIQ_6.4.00.AZ`

2. Run the following command and parameters from the directory in which the Adapter_Extractor resides on the AR server:

   AZExtraction.bat *ARserver port user password fullyqualifiedPathToAZfile mode*

   For example, **AZExtraction.bat pontoon 0 Demo sp06!#Hth c:\auser\iiq_7.6.04.AZ overwrite_all**

3. Restart the Action Request Server.

Configure

Use the following steps to configure the adapter in the AR environment.

1. Access the Application Administration Console.
2. Navigate to **Custom Configuration**=> **Identity Request Management**.
3. Click to expand **Adapter Configuration**.
4. Double-click **Adapter Configuration**.
5. Select the IdentityIQ Adapter from the first table.
6. Click the **Add Company - Site Configuration** option.
7. Fill in the parameters. See "Identity Request Management (IRM) Configuration Parameters" on page 53 for further information.
8. Click **Save Company - Site Configuration** to save the parameters.
9. Restart the Action Request server.

Identity Request Management (IRM) Configuration Parameters

There are three basic types of parameters:

- **Required connection**
  - **URL**: The URL of your IdentityIQ instance. For example, **http://pontoon:8080/iiq/**
  - **user**: Default is admin. This is the IdentityIQ proxy account on which to run services.
  - **Password**: The password for the IdentityIQ proxy account.
- **Workflow**: These are the workflows that are called during the various activities performed in IRM.
  - **accountWorkflow**: Specifies the workflow to call when performing unlock, enable and revoke all tasks. Default setting is **LCM Provisioning**.
  - **entitlementWorkflow**: Called when requests for role additions and removals are initiated. Default setting is **LCM Provisioning**.
  - **passwordWorkflow**: Called when password management requests are made through IRM. Default setting is **LCM Manage Passwords**.
  - **entitlementApprovalScheme**: Default setting is the owner.
  - **accountApprovalScheme**: Default setting is **None**.
  - **workflowTrace**: Default setting is **False**. Change value to **True** to enable IdentityIQ workflows to generate trace reports when executed.
  - **ticketingApplication**: The name of the application which is in conjunction with ticket updates
  - **doTicketingSystemNotification**: Provides added logic based on source, or other variables used to determine if tickets should be managed through a called subprocess. Default setting is **FALSE**.
- **Subprocess Workflow Variables**: The following are additional processes called when creating and updating an SRM ticket.
  - **project**: The current project, can be used by the rule when building the ticketing system provisioning plan.
  - **application:** The name of the application that represents the ticketing system.
  - **identityRequest**: Can be used by the rule to build up the ticket data, and is updated with the external ticket ID when one is returned by the call to the provisioning ticket.
  - **ticketDataGeneratorRule**: Rule that is launched to convert an IdentityRequest object and any additional workflow variables into a map of name /attributes for the SRM ticket.
  - **action**: An identifier which provides context to how the subprocess is called. The create method is typically only called once. The update call is performed after an action completes. This guides the rule on how to form the plan.
- **Launch** - The following settings specify which URL is called when the functions are executed within IRM.
  - **enableStartPageURL**: This is the page that is launched during the enable launch command. Default setting is `lcm/manageAccounts.jsf`.
  - **unlockStartPageURL**: Defaults to `lcm/manageAccounts.jsf`. This is the page that is launched during the unlock launch command.
  - **passwordStartPageURL**: Defaults to `lcm/managePasswords.jsf`. This is the page that is launched during the password launch command.

- **encodeURLParameters**: Defaults to **True**. This is used when launching the URLs, and when enabled the values on the URL are Base64 encoded.

Verify

Use the following steps to verify the installation and configuration:

1. Login to the BMC Remedy Action Request System as a user with request privileges
2. Access the Application tab.
3. Click **Service Request Management** to reveal its pop-up menu.
4. Click **Request Entry**.
5. Use the term “identity” in the search field to populate the Services window with identity management-related options.
6. Click **Request Access Right**.

If a separate window launches displaying the IdentityIQ user interface then the installation is deployed successfully. In the event of an error, verify the steps described in “Install” on page 52 and “Configure” on page 52 and see “Troubleshooting” on page 54 below.

Troubleshooting

If issues occur during installation or normal operation, SailPoint Support Engineers are available to assist. One valuable piece of information used in the troubleshooting process is a stack trace. This section provides the steps for enabling the stack trace capability for both IRM and IdentityIQ.

1 - Enable Stack Tracing for IRM

1. Locate the `log4j_pluginsvr.xml` file in the IRM installation directory. Default location is `C:\Program Files\BMC Software\ARSystem\pluginsvr\log4j_pluginsvr.xml`.
2. Define an appender that defines where the log/trace file resides.

   Note: Failure to define where the log/trace file resides results in using the default location which stores massive amounts of data.

   The following is an example appender:

```
<appender class="org.apache.log4j.RollingFileAppender" name="IIQPLUGINLOG">
<param name="File" value="C:/temp/iiqplugin.log"/>
<param name="MaxFileSize" value="999KB"/>

                <!-- Keep two backups -->

<param name="MaxBackupIndex" value="2"/>
      <layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d %-5p [%t] %C (%F:%L)- %m%n"/>
      </layout>
</appender>
```

3. Define a log level and appender for `iiq.integration`.

   The following is an example log level definition and appender:

```
<logger name="sailpoint.integration">
    <level value="trace"/>
    <appender-ref ref="IIQPLUGINLOG"/>
</logger>
```

2 - Enable Stack Tracing for IdentityIQ

Set the **workflowTrace** attribute to **TRUE** when configuring the IRM adapter parameters to enable logging. This makes any IdentityIQ provisioning actions traceable.S

Assisted Deployment Integration Modules

This section contains information on the following sections:

- **Provisioning Integration Modules**

  IdentityIQ can be configured to integrate with provisioning providers to automate access management for your implementation. Provisioning providers can be configured to communicate user and account information and automatically add or revoke access.

  - "SailPoint Microsoft Forefront Identity Manager Provisioning Integration Module" on page 59

- **Service Management Integration Modules**

  IdentityIQ supports Service Management Systems for Service Desk Integration and Service Catalog Integration.

  - **Service Desk Integration**: Creates tickets in Helpdesk systems based on compliance and provisioning actions taken in IdentityIQ (For example, new account request).
    - "SailPoint HP Service Manager Service Integration Module" on page 71
  - **Service Catalog Integration**: Enables enterprises to include LCM requests (for example, access request, password change) in the service catalog of the Service Request Management system.
    - "SailPoint ServiceNow Service Catalog Integration" on page 77

- **Mobile Device Management Integration Modules**

  Mobile Device Management Integration Module (MIM) manages Devices enrolled in Mobile Device Management (MDM) System. MIM also manages Users/Administrators of MDM System. Following is an architecture diagram of MIM:

  MDM systems import users from a central directory server or maintains its own repository. Operations to be performed on devices are sent to the respective MDM System which then performs the specified action on the target device. MIM does not communicate with the devices directly.

  MIM uses two separate applications to manage users and devices. Main MIM application manages users in the MDM System and the proxy MIM application manages devices in the MDM System. In some cases there is one application which manages devices.

This section contains information on the following:

- "SailPoint AirWatch Mobile Device Management Integration Module" on page 85
- "SailPoint MobileIron Mobile Device Management Integration Module" on page 89
- "SailPoint Good Technology Mobile Device Management Integration Module" on page 93

- **IT Security Integration Modules**

  IT Security Integration Modules (ISIMs) provide enhanced security with improved responsiveness and controls by exchanging identity and access information

  - "SailPoint HP ArcSight Integration Module" on page 97
  - "SailPoint Lieberman Integration" on page 105

- **Data Access Governance Integration Modules**

  SailPoint Data Access Governance Integration Modules expand IdentityIQ's core IAM capabilities such as access certification and access request to address data governance focused use cases.

  - "SailPoint STEALTHbits Integration" on page 107

A minority of SailPoint customers have deployed the Integration Modules in this section. SailPoint will provide assistance during the deployment of these integrations. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided on Compass, SailPoint's online customer portal. In some instances, SailPoint will guide the deployment team and actively participate in the design, configuration, and testing of the integration to the managed system.

For more specific information, refer to the Connector and Integration Deployment Center on Compass.

Chapter 9: SailPoint Microsoft Forefront Identity Manager Provisioning Integration Module

The following topics are discussed in this chapter:

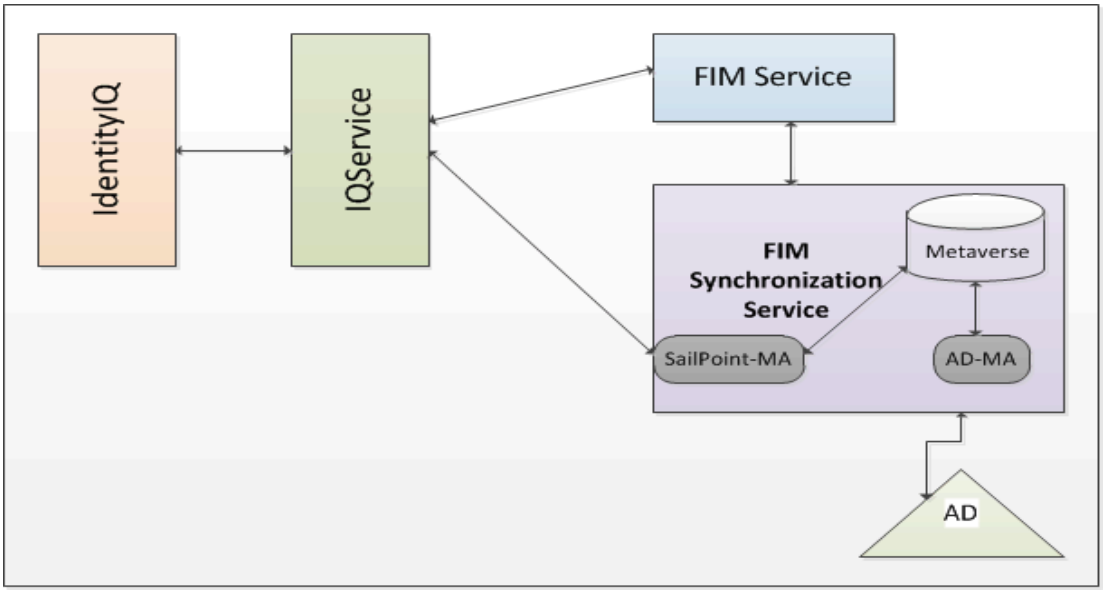
Overview 59
Supported features 61
Supported platforms 61
Configuring Microsoft Forefront Identity Manager for IdentityIQ Integration 61
IQService 61
Microsoft Forefront Identity Manager 2010 Extensible Connectivity 2.0 Management Agent 62
Configuring IdentityIQ for Forefront Identity Manager Integration 63
Create Forefront Identity Manager application in IdentityIQ. 63
Run Microsoft Forefront Identity Manager Application Creator Task 64
Operation specific configuration on Microsoft Forefront Identity Manger 65
Troubleshooting 67
Known/Open issue 68

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

This chapter provides a guide to the integration between Microsoft® Forefront® Identity Manager (FIM) and SailPoint IdentityIQ. This chapter is intended for FIM and IdentityIQ System Administrators and assumes a high degree of technical knowledge of these systems.

To execute request and response process, the request from IdentityIQ to FIM follows the following flow diagram:



The following table displays a comparison between terminology used in IdentityIQ and similar objects in Microsoft Forefront Identity Manager:

Microsoft Forefront Identity Management	IdentityIQ
Metaverse Person	Identity
Metaverse Group	An account group for the IdentityIQ applications created to correspond to the Microsoft Forefront Identity Manger Synchronization Engine. (Main application in IdentityIQ)
Account	An account for the IdentityIQ applications created for the corresponding Management Agents of Microsoft Forefront Identity Manger. (Child Application in IdentityIQ)
Group	An account group for the IdentityIQ applications created for the corresponding Management Agents of Microsoft Forefront Identity Manger. (Child Application in IdentityIQ)

Note: Microsoft Forefront Identity Manager Administrator must configure Metaverse Extension rule or Synchronization rule to provision data from Metaverse to SailPoint-MA connector space. In addition to this, Administrator must configure required rules to project SailPoint-MA connector space data to Metaverse and to other Management Agent connector space. For more information see, “Operation specific configuration on Microsoft Forefront Identity Manger” on page 65.

Supported features

The Microsoft Forefront Identity Manager Provisioning Integration Module provides the ability to provision Microsoft Forefront Identity Manager Metaverse users, Target Application accounts, and entitlements from IdentityIQ.

The Microsoft Forefront Identity Manager Provisioning Integration Module supports the following functions:

- User Management
  - Create, Update, Delete
  - Enable, Disable, Unlock, Reset Password
  - Aggregation, Delta Aggregation
- Target Application Accounts Management
  - Create, Update, Delete
  - Enable, Disable, Unlock, Reset Password
  - Aggregation, Delta Aggregation
- Account - Group Management
  - Aggregation, Delta Aggregation for Groups
  - Aggregation, Delta Aggregation for target application Groups
  - Add/Remove Entitlement for Users
  - Add/Remove Entitlement for target application Accounts

Supported platforms

- Microsoft® Forefront® Identity Manager 2010 R2
- Microsoft® Forefront® Identity Manager 2010

Configuring Microsoft Forefront Identity Manager for IdentityIQ Integration

The Microsoft Forefront Identity Manager for IdentityIQ integration has dependencies on the following components:

- IQService
- Microsoft® Forefront® Identity Manager (FIM) 2010 Extensible Connectivity 2.0 Management Agent (ECMA 2.0)

IQService

You must install and register an IQService on windows host computer before provision to Microsoft® Forefront® Identity Manager. The IQService is a native Windows service that enables IdentityIQ to participate in a Windows environment and access information from Microsoft® Forefront® Identity Manager Synchronization Engine.

For more information on installation of IQService, see "Appendix F: IQService" of the *SailPoint IdentityIQ Direct Connectors Administration and Configuration Guide*.

Note: - Do not install IQService on the Windows Computer where Microsoft Forefront Synchronization Service is running.
- The IQService ECMA Port should be allowed to communicate through fire wall on IQService installed host.

To enable provisioning using IQService in Microsoft® Forefront® Identity Manager, perform the following:

1. Download `iiqIntegration-FIM-SyncEngineComponents.zip` from IdentityIQ installable zip files `$INSTALLDIR/integration/FIM`. Copy the `iiqIntegration-FIM-SyncEngineComponents.zip` file to a computer where IQService is installed.
2. Unzip the `iiqIntegration-FIM-SyncEngineComponents.zip` file and find `App.config` file located in the zip file.
3. Copy the `App.config` file to the IQService home directory.
4. Rename `App.config` file to `IQService.exe.config`.

Microsoft Forefront Identity Manager 2010 Extensible Connectivity 2.0 Management Agent

Using Microsoft Forefront Identity Manager Synchronization Engine Client create Extensible Connectivity 2.0 Management Agent using the `.dll` library file provided by SailPoint Technologies.

Note: Before creating Microsoft Forefront Identity Manager Extensible Connectivity 2.0 Management Agent, create the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\SailPoint\ECMA`. This registry key is useful to log the Extensible Connectivity 2.0 Management Agent debug information.

To create the registry key and management agent, perform the following:

1. Download `iiqIntegration-FIM-SyncEngineComponents.zip` from IdentityIQ installable zip files `$INSTALLDIR/integration/FIM`. Copy the `iiqIntegration-FIM-SyncEngineComponents.zip` file to a computer where FIM is installed under a temporary (*temp)* directory.
2. Unzip the `iiqIntegration-FIM-SyncEngineComponents.zip` file in the *temp* directory. The *temp* directory will have the following directory structure:
   - `temp\Config`
   - `temp\FimMA\Fim2010MA`
   - `temp\FimMA\Fim2010R2MA`
3. Create the registry key as follows:
   a. Open the Registry Editor. Import the `SailPointECMARegistry` file provided in the `temp\Config` file of the Registry Editor.
   b. After import, `HKEY_LOCAL_MACHINE\SOFTWARE\SailPoint\ECMA` registry key is created in the registry.

The registry contains the following default values:

- tracefile - C:\ecma_trace.log (Name and location of log file)
- tracelevel - 0 (Trace level for logging information)

To enable logging debug information for Extensible Connectivity 2.0 Management Agent, set the trace level to **3** and to disable logging, set the trace level to **0**.

4. Copy the `.dll` library files provided in the `temp\FimMA\<FIM_Version>MA` to the following directory, where `FIM_Version` is the installed version of Microsoft Forefront Identity Manager:
`FIM_HOME\2010\Synchronization Service\Extensions\`
5. Copy the `sailpoint.ecma.bindings.config` and `sailpoint.ecma.client.config` files provided in `temp\Config` directory to the following directory:
`FIM_HOME\2010\Synchronization Service\bin\`

   Note: To perform modification of the `sailpoint.ecma.client.config` and `sailpoint.ecma.bindings.config` files, provide the write permission to the 'Users' group.
6. Stop FIM Synchronization Service.
7. Navigate to FIM_HOME\2010\Synchronization Service\bin\ directory and add the following configuration to `miiserver.exe.config` file under the <configuration> section:

```
<!-- SailPoint Management Agent(ECMA2.0) Configuration -->
<system.serviceModel>
   <bindings configSource="sailpoint.ecma.bindings.config"/>
   <client configSource="sailpoint.ecma.client.config"/>
</system.serviceModel>
```

8. Start FIM Synchronization Service.
9. Using Microsoft Forefront Identity Manager Client GUI, import Management Agent using `temp\Config\SailPoint-MA.xml` configuration file.
This will create the Management Agent with default Run profiles.

   Note: Do not change the name and properties of the default Run profiles for above created Extensible Connectivity 2.0 Management Agent (SailPoint-MA).

   Note: To provisioned Metaverse Persons/Groups to newly created ECMA 2.0 Management Agent (that is, SailPoint-MA), use Metaverse Rules Extension or Synchronization Rules.

Configuring IdentityIQ for Forefront Identity Manager Integration

This section describes the following:

- Create an IdentityIQ application of type Microsoft Forefront Identity Manager
- Run Microsoft Forefront Identity Manager Application Creator Task
- Aggregation from Microsoft Forefront Identity Manager Provisioning Integration Module

Create Forefront Identity Manager application in IdentityIQ

1. Create a new IdentityIQ application of type Microsoft Forefront Identity Manager. Enter the following required attributes of Forefront Identity Manager:

Note: The application name for IdentityIQ application of type Microsoft Forefront Identity Manager must be different than that of any management agent's name in FIM server side.

Attributes	Description
IQService Host	Hostname of the computer where IQService is installed.
IQService Port	Port on which IQService is listening for IdentityIQ.
IQService ECMA Port	This port is opened by IQService to listen with Extensible Connectivity 2.0 Management Agent created in Microsoft Forefront Identity Manger Synchronization Engine.
User	Forefront Identity Manager user who has Administrator rights and must be a member of the following security groups on Microsoft Forefront Identity Manger Host: • FIMSyncBrowse • FimSyncAdmins User must belong to FIM Web Service (FIM portal) "Administrators" set. **Note: The name of the User can be in UPN (user@domain) or NetBIOS (domain/user) format.**
Password	Password of Forefront Identity Manager user.
FIM Server host	Hostname or IP of the computer where Microsoft Forefront Identity Manager is installed.
FIM Service Port	Port for the Microsoft Forefront Identity Manager Service.
FIM Synchronization Service Host	Enter the host name or IP of the computer where Microsoft Forefront Identity Manager Synchronization Service is installed. **Note: This attribute is not required, if Synchronization service and Web Service are installed on the same host machine.**
FIM MA Name	Name of Extensible Connectivity 2.0 Management Agent created in Microsoft Forefront Identity Manger Synchronization Engine using library (**.dll**) provided by SailPoint Technologies.

2. Click **Test Connection** to verify the connection to Microsoft Forefront Identity Manager.

Note: You can make use of the "FIM Application creator" task to discover all the application present in FIM environment. The input for this task would be newly created application of type "Microsoft Forefront Identity Manager" and executing this task would result in the creation of all multiplexed applications.

Run Microsoft Forefront Identity Manager Application Creator Task

This task creates an IdentityIQ application for each target system in Microsoft Forefront Identity Manager.

1. Select the Microsoft Forefront Identity Manager application created in step 1. on page 63.
2. Specify **Native Object Types of Account** this is Object type which is referred as account on native managed system (for example, Account, Person, User).
3. Specify **Native Object Types of Group** this is Object type which is referred as group on native managed system (for example, groups, Groups).

4. Click **Save** and **Execute**.

Operation specific configuration on Microsoft Forefront Identity Manger

To communicate with Microsoft Forefront Synchronization Service, IQService uses Windows Management Instrumentation (WMI). Hence, enure that Windows Firewall on Microsoft Forefront Identity Manger Host should allow Windows Management Instrumentation (WMI) traffic.

This section describes the various configurations required for the following operations:

- Aggregation
- Provisioning

Aggregation from Microsoft Forefront Identity Manager Provisioning Integration Module

To aggregate Microsoft Forefront Identity Manager Persons/Groups and target Management Agent accounts/groups, create and execute an IdentityIQ Account Aggregation task for Application of type Microsoft Forefront Identity Manager.

Once aggregation is complete, all Persons/Groups from Microsoft Forefront Identity Manger Metaverse are created as accounts/groups in IdentityIQ and gets associated with IdentityIQ Application of type Microsoft Forefront Identity Manager. Additionally, all of the target Management Agent accounts/groups are aggregated automatically and associated with the respective child applications in IdentityIQ.

To support paging in IdentityIQ Aggregation task, Extensible Connectivity 2.0 Management Agent library supports paging in Export, Full Export run profiles. Maximum batch size supported by Extensible Connectivity 2.0 Management Agent library is 500. The default batch size provided is 100.

Configuration for Aggregation

In Aggregation, SailPoint IdentityIQ does not read users and groups from Microsoft Forefront Identity Manger Metaverse. Instead, it reads users and groups from newly created ECMA 2.0 Management Agent (that is, SailPoint-MA). Before starting aggregation from SailPoint IdentityIQ, provision Microsoft Forefront Identity Manger Metaverse Persons/Groups to newly created ECMA 2.0 Management Agent (that is, SailPoint-MA).

To provision Metaverse Persons/Groups to newly created SailPoint Management Agent (ECMA 2.0), use Metaverse Rules Extension or Synchronization Rules.

Provisioning from Microsoft Forefront Identity Manager Provisioning Integration Module

In provisioning, SailPoint IdentityIQ pushes users/groups data only to newly created SailPoint Management Agent (ECMA 2.0) connector space.

Configuration for Provisioning

To project SailPoint Management Agent connector space data changes to Microsoft Forefront Identity Manger Metaverse, use the following configuration rules for newly created SailPoint Management Agent (ECMA 2.0) as per their sites specifications:

- Management Agent Rules Extensions
- Mapping rules (Attribute Flow Rules)
- Join Rules
- Projection Rules
- Deprovisioning Rules

Provisioning examples

This section describes the various examples of provisioning.

- **Create Account for Microsoft Forefront Identity Manager target Management Agent Active Directory**: This operation includes the following steps:
    a. IdentityIQ sends the **Create Account Request** for FIM Active Directory Management Agent.

    Request is received by newly created SailPoint Management Agent (ECMA 2.0) on FIM side.
    b. SailPoint Management Agent (ECMA 2.0) adds new account in its own connector space.
    c. Microsoft Forefront Identity Manger Synchronization Engine projects these connector space changes to metaverse using configuration rules (rules listed in “Configuration for Provisioning” section).

    In Metaverse, new Person is created and link between Metaverse person and SailPoint Management Agent (ECMA.2.0) Account is created.
    d. To provision new Metaverse Person to Active Directory Management Agent connector space, Synchronization Engine uses Metaverse rules Extension or Synchronization rule. If rule condition is satisfied, Synchronization engine provisions Metaverse Person to Active Directory Management Agent connector space.

    Note: Depending on the Metaverse rules Extension or Synchronization rule configured in Synchronization Engine, Account on other Management Agent (native system) will be created.
    e. Synchronization Engine creates link between Metaverse Person and Active Directory Management Agent connector space account.
    f. After running **Export** run profile on FIM Active Directory Management Agent, the Active Directory

Management Agent connector space's account gets created on native system.

- **Enable Account for Microsoft Forefront Identity Manger target Management Agent Active Directory**: Enable Account operation from IdentityIQ will set the value for SailPoint Management Agents **enableStatus** attribute to **True**. The **enableStatus** attribute can be mapped to user defined Metaverse attribute for propagating the changes to other interested Management Agent's connector space attributes. After running **Export** run-profile on interested management agents, status for accounts will be enabled.
- **Disable Account for Microsoft Forefront Identity Manger target Management Agent Active Directory**: Disable Account operation from IdentityIQ will set the value for SailPoint Management Agents **enableStatus** attribute to **False**. The **enableStatus** attribute can be mapped to user defined Metaverse attribute for propagating the changes to other interested Management Agent's connector space attributes. After running **Export** run-profile on interested management agents, status for accounts will be disabled.
- **Unlock Account for Microsoft Forefront Identity Manger target Management Agent Active Directory**: Unlock Account operation from IdentityIQ will set the value for SailPoint Management Agents **lockStatus** attribute to **False**. The **lockStatus** attribute can be mapped to user defined Metaverse attribute for propagating the changes to other interested Management Agent's connector space attributes. After running **Export** run-profile on interested management agents, status for accounts will be unlocked.
- **Change/Reset Account password for Microsoft Forefront Identity Manger target Management Agents**: Change Account password operation from IdentityIQ will set the value for SailPoint Management Agents **password** attribute. The **password** attribute can be mapped to user defined Metaverse attribute for password. The metaverse password attribute should be mapped in metaverse outbound mappings for target management agents **export_password** attribute. After running **Export** run-profile on interested management agents, password gets changed on native system's accounts.

Troubleshooting

This section provides the resolutions for the following errors that may be encountered while setting up and configuring Microsoft Forefront Identity Manager.

1 - An unexpected error occurred: sailpoint.connector.ConnectorException: Errors returned from IQService. The maximum message size quota for incoming messages (3310720) has been exceeded. To increase the quota, use the MaxReceivedMessageSize property on the appropriate binding element.

When running any task (aggregation, application generation, and so on) related to Microsoft Forefront Identity Manager Application, the following error message may appear:

```
An unexpected error occurred: sailpoint.connector.ConnectorException: Errors returned
from IQService. The maximum message size quota for incoming messages (3310720) has
been exceeded. To increase the quota, use the MaxReceivedMessageSize property on the
appropriate binding element.
```

Resolution: Perform the following:

1. Increase the size of **MaxReceivedMessageSize** parameter as required, from the `IQService.exe.config` file and then restart IQservice.
2. If similar type of error is observed for other parameters then increase the size of particular parameter from the `IQService.exe.config` file and then restart IQservice.

2 - An unsecured or incorrectly secured fault was received from the other party. See the inner FaultException for the fault code and detail. An error occurred when verifying security for the message."

When running any task (aggregation, application generation, and so on) related to Microsoft Forefront Identity manager application, the following error message may appear:

```
An unsecured or incorrectly secured fault was received from the other party. See the inner FaultException for the fault code and detail. An error occurred when verifying security for the message.
```

Resolution: Perform the following:

1. Verify date and time on the host computer where IQService is running.
2. Verify date and time on the host computer where Forefront Identity Manager is running.
3. Ensure that, both host do not have time difference more than 10 minute.

3 - While creating Extensible Connectivity 2.0 Management Agent (SailPoint-MA), the following error may be observed in ECMA log file:

```
System.ServiceModel.QuotaExceededException: The maximum message size quota for incoming messages (65535) has been exceeded. To increase the quota, use the MaxReceivedMessageSize property on the appropriate binding element.
```

Resolution: Perform the following:

1. Increase the size of **MaxReceivedMessageSize** parameter as required from the `sailpoint.ecma.bindings.config` file and restart FIM Synchronization Service.
2. If similar type of error is observed for other parameters then increase the size of particular parameter from the `sailpoint.ecma.bindings.config` file and then restart FIM Synchronization Service.

Known/Open issue

Following are the known/open issues of Microsoft Forefront Identity Manager:

- User updates to attributes that affect target system accounts require a synchronization rule to be setup and run on the FIM server for the change to propagate.
- Due to Microsoft Forefront Identity Manager Synchronization's engine behavior, the Provisioning Integration Module will process all the request sequentially.
- Refresh schema action on SailPoint provided Extensible Connectivity 2.0 Management Agent (ECMA2.0) library is not supported.
- Delta Aggregation limitations
  - If new delta data is available in the SailPoint Management Agent (SailPoint-MA) connector space and user is unaware of it then there is a possibility that user might run Provisioning operation on the new delta data before running delta aggregation. If the new delta data is Provisioned before delta aggregation then the status of all the new delta data will be changed to **Awaiting Export**. In this scenario, running delta aggregation will not work since the status delta data is not **Pending Export**.
  - At a given time, only account aggregation or group aggregation is supported in a sequence such that account aggregation must be run first.

Known/Open issue

Chapter 10: SailPoint HP Service Manager Service Integration Module

The following topics are discussed in this chapter:

Overview 71
Supported features 71
Supported platforms 71
Pre-requisites 72
Configuring HP Service Manager for IdentityIQ Integration 72
Troubleshooting 76

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

The integration between SailPoint and HP Service Manager Desk enables customers to create incidents and change requests in HP Service Manager for the configured operations (for example, Change Password, Request Entitlement and so on) for the configured application. The seamless integration of SailPoint and HP Service Manager Service Integration Module (HP SIM) eliminates the need to build and maintain a custom integration, and speeds time-to-deployment.

Supported features

HP Service Manager Service Integration Module supports the following features:

- Creates Incidents and Change Requests in HP Service Manger platform through provisioning request in IdentityIQ.
- Fetching the status of Incident from HP Service Manager and update the status of the respective Access Requests in IdentityIQ.
- Fetching the status of Change Request from HP Service Manager and update the status of the respective Access Requests in IdentityIQ.

Supported platforms

SailPoint HP Service Manager Service Integration Module supports HP Service Manager version 9.30.

Pre-requisites

- Ensure that HP Service Manager 9.30 is installed and running.
- Ensure that the HP Service Manager 9.30 Server service is stated.
- When running a HP integration on an application server using Java 6, the JRE must be directed to use the SOAP/SAAJ implementation provided by Axis2. To do this, add the following Java options to the application server:
  - `-Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl`
  - `-Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl`
  - `-Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl`

  Without these options, a `ClassCastException` error or Signature creation failed error will occur when the HP integration is used.
- Ensure that the HP Service Manager Service Integration Module is up and running: Ensure that the following WSDL is accessible:
  - For incident: **http://<host>:<port>/SM/7/IncidentManagement.wsdl**
  - For change request: **http://<host>:<port>/SM/7/ChangeManagement.wsdl**

  Where <host> is the host name of the system where HP Service Manager is setup and <port> is port number configured for the above web services on HP Service Manager setup.Alternatively, use a Soap UI tool to submit a simple request (for example, incident). For convenience you can use the basic authentication mechanism for authorization with SOAP UI tool to confirm that the web service layer is functional.

Configuring HP Service Manager for IdentityIQ Integration

This section provides the required information for configuring IdentityIQ to integrate with HP Service Manager. This integration enables IdentityIQ to create tickets for requested revocations, track ticket numbers in association with revocation tasks, and update IdentityIQ with the status of current tickets.

SailPoint provides a default HP Service Manager Service Integration configuration. This configuration implements the integration between IdentityIQ and the HP to fulfill creation of tickets based on IdentityIQ access certification remediation events.

When integrating with **changeRequest**, modify `sampleHPSIMIntegrationForChangeRequest.xml` sample file and import in IdentityIQ. When integrating with **incident**, modify `sampleHPSIMIntegrationConfig.xml` file and import in IdentityIQ.

The default configuration is located in *`iiqHome`*`/WEB-INF/config/` directory, where *iiqHome* is the location where IdentityIQ was installed.

This section explains the various entries that are specific for this integration. For more information of the entries in the `IntegrationConfig` file, see Appendix: A: Common Identity Management Integration Configuration.

The integration configuration must include the following entries:

- **endpoint**: URL to the web service
- **namespace**: namespace of the XML returned by the web service
- **prefix**: prefix associated with the namespace

The integration configuration includes the following entries if the web service side of the integration is configured for authentication using the SOAP authentication specifications:

- username
- password
- statusMap

The web services and authentication entries are consumed by configuration entries for each web service. They can be positioned either within the configuration entries themselves or as children of the Attributes element. Entries that are children of the Attributes element can be thought of as global values, while entries within the configuration entities can be thought of as local.

For example, if both entries share the same authentication credentials, those credentials might be placed in the Attributes element as peers of the configuration entries and the integration code searches the parent entry for the credentials if they are not found in the configuration entries. Conversely, if the configuration entries have different endpoints (are handled by separate web services), each configuration entry specifies the endpoint of the web service to call and any value outside of the configuration entry is ignored.

There are two supported configuration entries for integration with HP Service Manager. These entries are children of the integration Attributes element:

- getRequestStatus
- provision

The values of each are Map elements containing key/value pairings of the configuration data. They contain the specific data needed by the **getRequestStatus()** and **provision()** methods of the IdentityIQ integration executor and correspond to HP Service Manager Web Service methods.

The **getRequestStatus** and **provision** entries contain the following entries:

- **soapMessage** (required): full XML template of the entire SOAP envelope that is sent to the web service. The integration code first runs this template through Apache's Velocity template engine to provide the data needed by the web service.
- **responseElement** (required): name of the element containing the results of the web service call (for example, the element containing the ticket number opened by the web service in response to the call from IdentityIQ).
- **SOAPAction** (required): SOAP request's action.
- **endpoint** (required)
- **namespace** (required)
- **prefix** (required)

Before a template is sent to the web service, it is processed by the **Velocity template engine**. The integration code provides different data objects to Velocity for evaluation based on the integration method.

The **provision** call passes the following objects to Velocity:

- **config**: the integration configuration for provision, represented as a Map
- **provisioningPlan**: the data model of the provision request

The **getRequestStatus** call passes the following objects to Velocity:

- **config**: the integration configuration for getRequestStatus, represented as a Map
- **requestID**: the string ID of the request whose status is being queried

Both calls have access to a timestamp variable containing a current Date object and a dateFormatter object. The dateFormatter is built using an optional dateFormat attribute from the **config** object. If the dateFormat attribute does not exist, the formatter defaults to the pattern EEE, d MMM yyyy HH:mm:ss z.

Sample XML files for Incident and Change Request

The entries contained in the Map are the only required entries. Any authentication information required by this integration is inherited from the parent Attributes element.

For more information and examples of the sample files, see the following sample files:

- `sampleHPServiceManagerIntegration.xml`
- `sampleHPServiceManagerIntegrationForChangeRequest.xml`

If any changes required in the mapping, change the value/key values in “statusMap” and “statusMapCloserCode” as mentioned in the following table:

Entry key	Values
statusMap	
Closed	committed
Pending Other	queued
Referred	queued
Replaced Problem	queued
Resolved	committed
Open	queued
Accepted	queued
Rejected	queued
Work In Progress	queued
Pending Customer	queued
Pending Vendor	queued
Pending Change	queued
initial	queued
statusMapCloserCode	
Automatically Closed	committed
Not Reproducible	committed
Out of Scope	committed

Entry key	Values
Request Rejected	committed
Solved by Change/Service Request	committed
Solved by User Instruction	committed
Solved by Workaround	committed
Unable to solve	failed
Withdrawn by User	failed

Configuration procedure

The following steps should be performed to modify the default HP Service Manager Service Integration configuration for a specific HP Service Manager Server.

1. Obtain the environment-specific Web Service “endpoint”, for example, **http://<host>:<port>/SM/7/ws**.
2. Once you are familiar with the WSDL, modify the default IdentityIQ HP Service Manager configuration using the information collected about the web service.
    a. In the <IntegrationConfig> element of the integration configuration, modify the **username** and **password** entries in the attributes map to contain the credentials required for authentication to the web service.
    b. In the <IntegrationConfig> element of the integration configuration, modify the provision entry of the Attributes map by setting the endpoint, and, if necessary, the namespace, the prefix, the responseElement, and the soapMessage attributes (the default values: IdentityIQ HP Service Manager IntegrationConfig):

        i. Set the value for endpoint to the value located in the WSDL earlier.

        Note: The value in the IdentityIQ integration configuration must be a valid HTTP URL and have any special characters escaped. The most common change that must be made is to replace all and symbols with &

        ii. The value for namespace comes from the **targetNamespace** attribute of the **xsd:schema** element in the WSDL.

        iii. The value for prefix is the prefix of the XML elements that will be contained in the SOAP response.

        iv. The value for **responseElement** should be the HP Service Manager form field that corresponds to the id of the form that the web service creates.

        v. The value for **soapMessage** should be the SOAP message body that IdentityIQ will send to HP Service Manager. The exact format of this message is a function of the form that is published as described by the form's WSDL. The XML elements in the **soapenv:Body** element should be changed to match the HP Service Manager form fields for the published web service. Each required HP Service Manager form field must have an element in the SOAP message. The value can be fixed or can be a variable that will be substituted using IdentityIQ's Velocity templating

The information in the reference section above show the variables that are provided and the example integration configuration provides examples of how they are used.

Troubleshooting

This section provides the resolutions for the following errors that may be encountered while setting up and configuring HP Service Manager Service Integration Module.

1 - 'Authorization Required' error messages

The following type of error messages appear when the authorization data is not sent to HP Service Manager:

```
Caused by: org.apache.axis2.AxisFault: Transport error: 401 Error:
Authorization Required
```

Resolution: Verify the procedure to configure appropriate authorization mechanism. For more information on the procedure, see "Configuration procedure" on page 75.

2 - Document Type Declaration (DTD) parsing errors

The DTD parsing errors appear when the following JVM arguments are not defined for your application server:

- `-Djavax.xml.soap.SOAPConnectionFactory=org.apache.axis2.saaj.SOAPConnectionFactoryImpl`
- `-Djavax.xml.soap.MessageFactory=org.apache.axis2.saaj.MessageFactoryImpl`
- `-Djavax.xml.soap.SOAPFactory=org.apache.axis2.saaj.SOAPFactoryImpl`

Resolution: Ensure that the application is pointing to the correct java runtime (that is, 1.6) and the above mentioned JVM arguments are defined for application server.

Chapter 11: SailPoint ServiceNow Service Catalog Integration

The following topics are discussed in this chapter:

Overview 77
Supported features 78
Supported platforms 79
Pre-requisites 79
Installation and configuration in ServiceNow 79
Configuration in SailPoint IdentityIQ 82
Troubleshooting 83

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

SailPoint Service Catalog Integration is an integration between ServiceNow and SailPoint IdentityIQ. This allows users of both systems to easily navigate from ServiceNow into IdentityIQ, and gives users a "one stop shop" to request all IT related items.

The integration between SailPoint and ServiceNow gives mutual customers a complementary identity access governance and service management solution that works together to ensure strong controls are in place to meet ever stringent security and compliance requirements around user access to sensitive applications. The integration also allows users to perform other activities (such as password changes, approve access, manage account) that are configured within the system.

The chapter describe the implementation approach and configuration of the SailPoint Service Catlog Integration in ServiceNow. The following information enables administrators to deploy and maintain the integration.

The flow of SailPoint Service Catalog Integration in ServiceNow is as follows:

- User logs in to ServiceNow.
- The **SailPoint Service Catalog Integration** application is available to the logged in user.
- Click on any of the links mentioned in the request section of SailPoint Service Catalog Integration Application. Request flows to IdentityIQ and IdentityIQ page is opened in ServiceNow.
- Complete the request in IdentityIQ page opened in ServiceNow and Ticket would be generated in ServiceNow for the same request. Initially the status of ServiceNow ticket would be **open** or **openNoApproval**.
- Once the request is approved in IdentityIQ, ServiceNow ticket’s status would be updated.
- Once the provisioning of request is done on IdentityIQ side, the ticket’s status would be updated (closed) in ServiceNow.

The following diagram represents the high level flow diagram for Service Catalog Integration with SailPoint IdentityIQ:

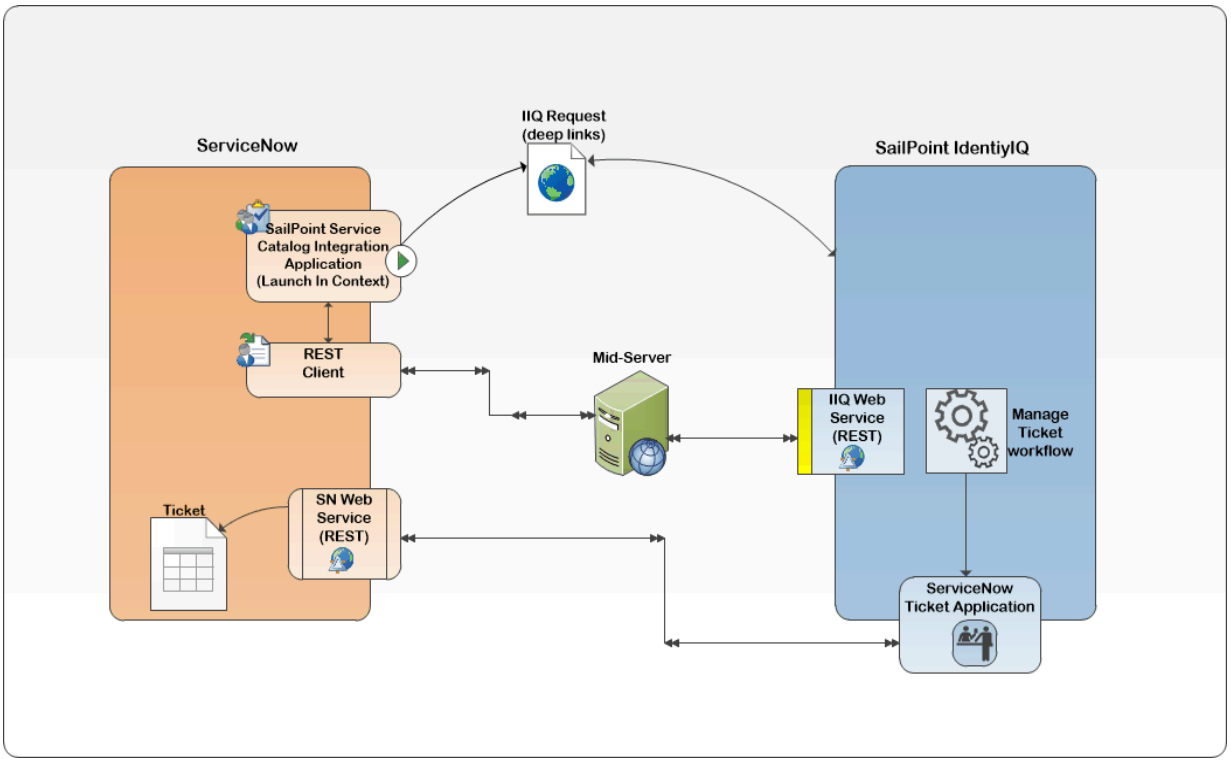


Figure 1—Basic Configuration

If a request is submitted from ServiceNow then SailPoint IdentityIQ creates or updates tickets in ServiceNow using ServiceNow Ticket Application from IdentityIQ.

Supported features

The SailPoint Service Catalog Integration enables the use of IdentityIQ Lifecycle Manager within ServiceNow user interfaces. The inclusion of the SailPoint Service Catalog Integration Application provides the following functionality within ServiceNow:

- Manage Account for Me
  - Enable/Disable/Unlock/Delete Accounts
- Request Access for Me/Others
  - Request for Entitlement
  - Request for Roles
- Request Password Changes
- My Access Approvals

- Track My Requests
  - Get Request Status
  - Get complete details of request Item

Supported platforms

SailPoint Service Catalog Integration supports the following ServiceNow instance versions:

- Eureka
- Dublin
- Calgary

Pre-requisites

- For the integration to successfully authenticate user, the ServiceNow accounts (users) must have identities in IdentityIQ with the same name.
- ServiceNow Mid Server must be installed. See “Mid server installation” on page 81.

Note: Ensure that the WebServer hosting IdentityIQ must be SSL enabled.

Installation and configuration in ServiceNow

This section describes the installation and configuration procedures for SailPoint IdentityIQ ServiceNow Service Catalog Integration.

Installation

This section describes the installation procedure.

Install update set in ServiceNow

Apply the IdentityIQ ServiceNow Service Catalog Integration update set:

1. Login to the SailPoint Compass Community using the following link:
   https://community.sailpoint.com/groups/product-download-center
2. On the Product Download Center page, click on IdentityIQ version 6.4.
   Based on the required version of ServiceNow, copy the relevant update set from the following respective files:

ServiceNow version	Update Sets
`iiqIntegrations-ServiceNow-calgary_6.4.zip`	`IdentityIQServiceNowServiceCatalog.v1.8_calgary_6.4.xml`
`iiqIntegrations-ServiceNow-dublin_6.4.zip`	`IdentityIQServiceNowServiceCatalog.v1.8_dublin_6.4.xml`

ServiceNow version	Update Sets
iiqIntegrations-ServiceNow-eureka_6.4.zip	IdentityIQServiceNowServiceCatalog.v1.8_eureka_6.4.xml

3. Import relevant update set in ServiceNow instance. For more information and guidelines on usage of the update set, refer to the following wiki link:
http://wiki.servicenow.com/index.php?title=Saving_Customizations_in_a_Single_XML_File

   In the above link, refer to section 3 “Loading Customizations from a Single XML File”.

After successfully applying the update set in ServiceNow the SailPoint IdentityIQ ServiceNow Service Catalog Integration Application is created.

Overview “SailPoint IdentityIQ ServiceNow Service Catalog Integration” Application

The SailPoint IdentityIQ ServiceNow Service Catalog Integration provides easy access to IdentityIQ Services, Access Requests and Access Approvals.

Note: The SailPoint IdentityIQ ServiceNow Service Catalog Integration application is available to only those user who has the `spnt_identityiq_user` role assigned in ServiceNow.

1. My Access Requests
   - Displays a list of access requests which include the Access Requests Opened By or Requests for the current user logged in.
   - By clicking into the IdentityIQ request, a user can see the Request information, as well as all of the Request Items with a brief description of what the Request Item entailed.
   - By clicking on the **View Detail** button a user can view the details of this Request in IdentityIQ.
2. My Access Approvals
   - Allows user to view the pending approvals in IdentityIQ for the logged in user. In addition to this, the module shows number of pending approvals for the logged in user with count next to link.
3. Request Access for Me
   - Allows user to request application access.
   - The **Request Access for Me** directly links to the request access page, allowing currently logged in user to request entitlements and roles.
4. Request Access for Others
   - The **Request Access for Others** link displays a dialog box so that the current user can select a different user to request access on their behalf. Once a user is selected, this links to the request access page.

   Note: The “Request Access for Others” requires either the ServiceNow `admin` or the `spnt_identityiq_manager` role to display. The `spnt_identityiq_manager` role is the custom role created to give to managers of other users from IdentityIQ in ServiceNow.
5. Change Password
   - Manages password for different accounts managed in IdentityIQ.
   - The **Change Password** link directs the link to the logged in users to manage the passwords in IdentityIQ.

6. Manage Accounts for Me
    - The **Manage Accounts for Me** opens the manage accounts page in IdentityIQ.
7. Track My Request
    - Allows user to check status regarding a recent access request.
    - Track My Requests opens the Access Request page in IdentityIQ.
8. Properties
    - Provides the form for logging in to the IdentityIQ API and performing a REST request to the MID SERVER and writing debug inform action to the Log.
9. Logs
    - Allows to view debugging, error and general information messages as they are written to the Log.
10. Customer Support
    - Allows the user to navigate to a support page located at Customer Support.

Mid server installation

For an overview and installation instructions on setting up the mid server, see the following ServiceNow wiki:

http://wiki.servicenow.com/index.php?title=MID_Server

Configuration

Set "SailPoint Service Catalog Integration" Application properties in ServiceNow to point to the SailPoint IdentityIQ instance

Open the **Properties** module from **SailPoint Service Catalog Integration** Application and modify the following properties:

Note: The properties module from SailPoint Service Catalog Integration application is available only to those user who have `spnt_identityiq_admin` **role in ServiceNow.**

- **Use Single-Sign On for IdentityIQ Requests**: Determines whether SailPoint IdentityIQ is setup for Single Sign On.
    - True: ServiceNow assumes no authentication is required when opening direct links into IdentityIQ.
    - False: ServiceNow uses the remoteLogin web service to retrieve a one-time use authentication token to login the current user.
- **IdentityIQ Instance Endpoint URL**: URL specifying the SailPoint IdentityIQ endpoint.
- **User to login to the IdentityIQ API**: Username for authentication during REST Requests to IdentityIQ.

  Note: This user should have System Administrator Permission in IdentityIQ
- **Password to login to the IdentityIQ API**: Password for authenticating during REST Requests to IdentityIQ.
- **Mid Server to use to make REST requests to IdentityIQ**: The name of the Mid Server to make REST Requests through to IdentityIQ.
- **IdentityIQ Log Level**: Log level for the IdentityIQ Access Request application. By default the value of the **IdentityIQ Log Level** parameter is **debug**, for production environments it is suggested to set the value to **error**.

- **Interval in which to poll for approvals**: Identifies the interval in which the approvals on the server side are verified. Default: 15 minutes.
  If the IdentityIQ Approval record for the user logged in has not been updated within 15 minutes, the user is logged out to IdentityIQ to request an update of the number of approvals outstanding.
- **Application Name within IdentityIQ for the ServiceNow Integration**: Application Name within IdentityIQ for the ServiceNow Integration. Default: `SailPointServiceCatalog`.
- **IdentityIQ ServiceNow Application Name**: ServiceNow users are aggregated in IdentityIQ through ServiceNow connector application.

Mid server Setup

The Mid Server must be setup with a polling time lower than normal. This allows the shortest lag time when loading IdentityIQ pages when ServiceNow must request a remoteLogin token.

MID Server poll time: Sets the MID Server polling interval (in seconds).
Type: integer (seconds)
Default value: 15
`mid.poll.time`

Note: It is suggested to set the "MID Server poll time" property to 5 seconds, to increase the rate in which REST Message Requests will be picked up and processed.

Note: ServiceNow and IdentityIQ instance must be accessable to Mid-server.

Configuration in SailPoint IdentityIQ

Perform the following procedure to create Application in IdentityIQ to manipulate Tickets in ServiceNow:

1. Import **ServiceNow Example Ticket Plan Generation Rule** rule in IdentityIQ from `WEB-INF/config/examplerules.xml` file.
2. Create **SailPointServiceCatalog** ticketing application using default configuration file located in *`iiqHome`*`/WEB-INF/config/SailPointServiceCatalog.xml` directory.
   In the above directory, *iiqHome* is the location where IdentityIQ is installed.
3. Modify the following parameters:
   - **url**: Service Now JSON API endpoint **https://<servicenow-base-url>/iiq.do**
   - **username**: ServiceNow user who has System Administrators role
   - **password**: Password of an user

   Note: - If rule name is changed in Step 1. then change the value for `ticketDataGenerationRule` entry in `SailPointServiceCatalog.xml` file.
   - If Application Name is changed in SailPoint Identity Access Request Application properties then change name in `SailPointServiceCatalog.xml` file.
   - Ensure that `ServiceNowServiceCatalog.xml` file contains feature string "PROVISIONING" in Schema section.
4. Enable **iFrame** option in IdentityIQ by performing the following steps:
   a. Navigate to IdentityIQ debug page.
   b. Select **System Configuration** from **Configuration Objects**.
   c. Find the key **allowiFrame** and set the value to **true.**
   d. Save the **System Configuration.**

Troubleshooting

1 - Enable Traces for SailPoint IdentityIQ Access Request Application in ServiceNow

Resolution: To enable the traces in SailPoint IdentityIQ Access Request Application set the property **IdentityIQ Log Level** (`com.sailpoint.iiq.log_level`) to **debug**.

2 - Enable Stack Tracing for IdentityIQ

Resolution: Set the workflow **Trace** attribute to **true** when configuring the ServiceNow SailPoint IdentityIQ Access Request Application parameters to enable logging. This enables any IdentityIQ provisioning actions traceable.

Chapter 12: SailPoint AirWatch Mobile Device Management Integration Module

The following topics are discussed in this chapter:

Overview 85
Supported features 85
Supported platforms 86
Pre-requisites 86
Configuration 86
Application configuration 87
Operation specific configuration. 87

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

This document provides a guide to the AirWatch Mobile Device Management (AirWatch) integration and configuration for your enterprise.

Using this integration, IdentityIQ can retrieve the devices managed by AirWatch, perform operations on them, and manage AirWatch's user account. These entities are managed in IdentityIQ using separate applications named as follows:

- AirWatch Mobile Device Management (MDM) Application (referred to as User Application in this document) for managing AirWatch user accounts
- Device Application (containing the prefix **-Devices**) is created by IdentityIQ during aggregation and is used for managing devices.

Supported features

The AirWatch MDM Integration Module supports the following features:

- Account Management
  - Account Aggregation on the user application to bring in AirWatch user accounts
  - Account Aggregation on the device application to bring in devices managed in AirWatch MDM
  - Delete, Unlock devices

  The following table represents what each of the above operation implies to IdentityIQ and AirWatch MDM:

IdentityIQ operation	Resulting change on AirWatch
Account aggregation on user application	The AirWatch user accounts are brought into IdentityIQ.
Account aggregation on device application	The devices that are managed by the AirWatch MDM are retrieved into IdentityIQ.
Delete account for AirWatch device application	Enterprise/Device Wipe and Delete device is triggered on the device via AirWatch MDM system. For more information, see “Operation specific configuration” on page 87.
Unlock account for AirWatch device application	Unlock device is triggered on the device via AirWatch MDM system.
Adding entitlements to account	Adding profiles to device from AirWatch system.
Deleting entitlements from account	Removing profiles from AirWatch system.

- Account - Group Management
  - Device Group Aggregation where device profiles are addressed as groups.
  - Add and remove entitlement (profile) from device.

Supported platforms

SailPoint AirWatch Mobile Device Management Integration Module supports AirWatch Platform Services version 7.0.0.0.

Pre-requisites

Administrator user configured for AirWatch MDM Application must have the following role for provisioning activities:

- REST API Devices Read
- REST API Devices Write
- REST API Devices Execute
- REST API Devices Delete
- REST API Devices Advanced

Note: IfAirWatch MDM application is behind proxy server, see the “Special Java Considerations” section of the *SailPoint IdentityIQ Installation Guide*.

Configuration

This section describes the application and additional operation specific configurations.

Application configuration

To create an application in IdentityIQ for AirWatch the following parameters are required:

Parameters	Description
Application Type	AirWatch MIM (Mobile Device Management Integration Module).
AirWatch Server	AirWatch server URL where it's REST API are accessible. For example, https://apidev-as.awmdm.com.
AirWatch Administrator	Administrator of AirWatch server.
AirWatch Administrator Password	Password of the administrator.
API Key	AirWatch server's API key defined for REST API.

Operation specific configuration

This section describes the various configurations required for the following operations:

- Aggregation
- Provisioning

Aggregation

- **Aggregation of devices**: Before aggregating devices against the device application create a correlation rule in the device application to map devices to its AirWatch user. For example, **UserName** is an attribute of the device which specifies the name of the user it belongs to, and **Display Name** of the identity is also **UserName**. So in correlation rule specify application attribute as **UserName** and Identity attribute as **Display Name**.
- **Parameterized device aggregation**: By default AirWatch device aggregation retrieves device profiles and device applications. If you do not want to manage these entities, you can filter them for not being retrieved into IdentityIQ.

  The following configurable parameters impact aggregation:

  - **aggregateDeviceProfile**: (is an application attribute on AirWatch MDM application) determines if the profiles connected to devices are to be retrieved or not. The default behavior is to retrieve the profiles connected to the devices. To change this behavior, set the following value through the debug pages:

    ```
    <entry key="aggregateDeviceProfile” value="false"/>
    ```

  - **aggregateDeviceApp**: (is an application attribute on AirWatch MDM Application) determines if the application installed on devices must be retrieved or not. The default behavior is to retrieve the applications installed on the device. To change this behavior, set the following value through the debug pages:

    ```
    <entry key="aggregateDeviceApp" value="false"/>
    ```

Provisioning

The following provisioning operations are available in IdentityIQ when integrating with AirWatch:

- Delete device

Delete device

- **Delete Device operation from LCM**: In addition to Delete Device, you will be prompted to select **Entire Device Wipe** or **Enterprise Wipe Only** options before deleting the device.
- **Delete Device operation from Certification**: The default wipe operation will be the **Enterprise Wipe Only**. To change this default behavior to **Entire Device Wipe** add the following entry in the application debug of the device application:

```
<entry key="defaultWipeFromCertification" value="Entire Device Wipe"/>
```

Chapter 13: SailPoint MobileIron Mobile Device Management Integration Module

The following topics are discussed in this chapter:

Overview 89
Supported features 89
Supported platforms 90
Pre-requisites 90
Configuration 90
Application configuration 90
Operation specific configuration. 91

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

SailPoint IdentityIQ manages MobileIron devices. It does not handle MobileIron users because MobileIron has not provided the supporting User API. It manages these MobileIron entities using the MobileIron REST API's over HTTPS. All the devices are aggregated under MobileIron Mobile Device Management (MDM) Application.

Supported features

The SailPoint MobileIron MDM Connector provides the ability to provision MobileIron devices from IdentityIQ.

The MobileIron MDM Connector supports the following functions:

- Account Management
  - Device aggregation which include device Attribute, Group Attribute - Labels and Entitlements - Apps, Labels.
  - MobileIron MIM support Create, Delete and unlock operations where Create is Registering a Device, Delete is Retire a Device along with Device Wipe that will wipe all the existing device applications and Unlock means unlocking a device by removing a PIN if Set.
- Account - Group Management
  - Device Group Aggregation where groups are Labels in MobileIron.

The following table displays a comparison between operations from IdentityIQ and resultant operations in MobileIron MDM:

IdentityIQ operation	Resulting change on MobileIron
Account aggregation on MobileIron application	The devices that are managed by the MobileIron MDM are retrieved into IdentityIQ with the label attribute represented as the entitlement for that device.
Account group aggregation on MobileIron application	The labels present in the MobileIron MDM are retrieved into IdentityIQ.
Create account of MobileIron application	Registering a device on MobileIron system. For more information, see “Operation specific configuration” on page 91.
Delete account of MobileIron application	Wipe/Retire device is triggered on the device via MobileIron MDM System. For more information, see “Operation specific configuration” on page 91.
Unlock account for MobileIron application	Unlock device is triggered on the device via MobileIron MDM system.
Adding entitlements to account	Adding labels to device from MobileIron MDM system.
Deleting entitlements from account	Removing device labels from MobileIron MDM system.

Supported platforms

SailPoint IdentityIQ MobileIron Mobile Device Management Integration Module supports API Version for MobileIron WebService 5.7.1.

Pre-requisites

Administrator user configured for MobileIron MDM application must have the API role for provisioning activities.

Note: If MobileIron MDM application is behind proxy server, see the “Special Java Considerations” section of the *SailPoint IdentityIQ Installation Guide*.

Configuration

This section describes the application and additional operation specific configurations.

Application configuration

To create an application in IdentityIQ for MobileIron the following parameters are required:

Parameters	Description
Application Type	MobileIron MIM (Mobile Device Management Integration Module).
MobileIron URL	The URL pointing to the MobileIron system.

Parameters	Description
MobileIron Administrator	Administrator of MobileIron system.
MobileIron Administrator Password	Password of the administrator.

Operation specific configuration

This section describes the various configurations required for the following operations:

- Aggregation
- Provisioning

Aggregation

MobileIron Device Aggregation retrieves all active status devices, their connected Labels and Applications. IdentityIQ manages labels as entitlement.

Default aggregation without importing or selecting **MobileIron MIM Correlation** for the MobileIron MIM application will create identity in IdentityIQ with **model** name attribute of MobileIron.

For example, `User Name: SM-T301 Account ID: SM-T301`
`User Name: iPhone4 Account ID: iPhone4`

The **MobileIron MIM Correlation** will display the effect only if identity are already created before account aggregation with same name as of **userDisplayName** attribute of MobileIron.

For example, `User Name: slupinson Account ID: SM-T301,`
`User Name: AJohn Account ID: iPhone4`

Select the **MobileIron MIM Correlation** under the Correlation tab.

The **MobileIron MIM Correlation** would be populated only if, the following correlation xml input is imported before running the account aggregation task:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE CorrelationConfig PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<CorrelationConfig name="MobileIron MIM Correlation">
  <AttributeAssignments>
    <Filter operation="EQ" property="displayName" value="userDisplayName"/>
  </AttributeAssignments>
</CorrelationConfig>
```

Provisioning

The following provisioning operations are available in IdentityIQ when integrating with MobileIron:

- Add device
- Delete device

Add device

Default template is present for adding a device (Registering a device to a user). If the user is not present, add a user in the MobileIron system (Local repository) and then assign a device to it.

Note: If new local user is created during "Device Registration" through IdentityIQ then the VSP (MobileIron Server) sets the password for a new local user to the userid.

Valid values for the platform attribute in the **Create Device Provisioning policy** are as follows:

- M - Windows Mobile
- B - BlackBerry
- I - iOS
- S - Symbian
- P - Palm webOS
- A - Android
- L - Mac OS X

Delete device

- **Delete Device via LCM**: In delete device operation, in addition to delete device, user is prompted to select one of the following options:
  - **Device Wipe** (Factory data reset): Device wipe wipes all the data from the device and sets it back to the factory setting.
  - **Enterprise Wipe (Retire)**: Enterprise Wipe removes the MobileIron Client present on the device.
- **Delete Device via Certification**: When a delete device operation is performed from certification, it does not respect the additional operations **Device Wipe** and **Enterprise Wipe**. Hence, IdentityIQ provides an extra MobileIron application attribute named **defaultWipeFromCertification**. Users have to manually add this attribute in the MobileIron MIM application debug page and provide values as **Device Wipe** or **Enterprise Wipe**.

  ```
  <entry key="defaultWipeFromCertification" value="Device Wipe"/>
  ```

  When deleting a device from IdentityIQ using the Device Wipe option, the operation first wipes the device (Data Reset) and then IdentityIQ waits (sleep) for 5 second (default). After the sleep duration IdentityIQ deletes the device from MobileIron system and changes the status from wiped to retire.

  The sleep duration for Wipe operation is a configurable application attribute which can be configured as follows:

  ```
  <entry key="WipeSleepInterval" value="5"/>
  ```

Chapter 14: SailPoint Good Technology Mobile Device Management Integration Module

The following topics are discussed in this chapter:

Overview 93
Supported features 94
Supported platforms 94
Pre-requisites 94
Configuration 94
Application configuration 94
Operation specific configuration. 95

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

This document provides a guide to the Good Technology Mobile Device Management (MDM) integration and configuration for your enterprise.

Using this integration, IdentityIQ can retrieve the devices managed by Good Technology and perform few operations on them. In addition, this integration also enables IdentityIQ to manage Good Technology's devices. These entities are managed in IdentityIQ using separate applications, named as follows:

- Good Technology MDM Application (referred to as User Application in this document) for managing Good Technology user accounts.
- Device Application (containing the suffix **- Devices**) is created by IdentityIQ during user aggregation and is used for managing devices.

Supported features

The Good Technology MDM Integration Module supports the following features:

- Account Management
  - Account Aggregation on the user application to bring in Good technology role member
  - Device Aggregation on the device application to bring in devices managed in Good Technology MDM
  - Add, Unlock Delete devices
- Account-Group Management:
  - Account Group Aggregation on the user application to bring in Good Technology roles
  - Account Group Aggregation on the device application to bring in policy sets managed in Good Technology MDM
  - Adding/ Removing Entitlement (Policy Set) to device account.

Supported platforms

SailPoint IdentityIQ Good Technology Mobile Device Management Integration Module supports Good Mobile Control version 2.4.1.539.

Pre-requisites

Administrator user configured for Good Technology MDM Application must have the Role with All Rights or following rights for provisioning activities:

- Add handheld for a user
- Add additional handhelds for a user
- Delete handhelds
- Wipe Good for Enterprise app or entire device data
- Reset Good for Enterprise
- Set handheld policy
- View Policy Sets
- Manage policies (handheld and application)
- Manage roles

Configuration

This section describes the application and additional operation specific configurations.

Application configuration

To create an application in IdentityIQ for Good Technology the following parameters are required:

Parameters	Description
Application Type	Good Technology MIM (Mobile Device Management Integration Module).
Good Technology Hostname	Provide computer name of the GMC server.
Good Technology Port	Provide port number of the GMC Web Service. For example, 19005
Good Technology Username	Administrator of GMC server.
Good Technology Password	Password of the administrator.
Page Size	Page size for aggregating devices.

Operation specific configuration

This section describes the various configurations required for the following operations:

- Aggregation
- Provisioning

Aggregation

Aggregation of devices: Good Technology device aggregation retrieves all devices, their connected Policy Set and applications on the device. IdentityIQ manages Policy Set as entitlement.

Default aggregation without creating a correlation rule for the Good Technology Mobile Device Management Integration Module (MIM) application will create identity in IdentityIQ with Display Attribute (Device Model by default) of Good Technology.

For example,

- Name: iPhone 4 (GSM, Rev. A) Account ID: phone 4 (GSM, Rev. A)

  The correlation rule will display the effect only if identity is already created before account aggregation with same name as of correlation attribute of Identity.

- Name: "Demo User1" Account ID: iPhone 4 (GSM, Rev. A)

  Import the following correlation xml which correlates devices to Identity based on name of a user in Good Technology and display name of an Identity:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE CorrelationConfig PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<CorrelationConfig name="Good Technology Device Correlation">
<AttributeAssignments>
<Filter operation="EQ" property="displayName" value="Name"/>
</AttributeAssignments>
</CorrelationConfig>
```

Provisioning

The following provisioning operations are available in IdentityIQ when integrating with Good Technology:

- Add device
- Unlock device
- Delete device

Add device

While adding a device from IdentityIQ, default provisioning policy accepts the following fields:

- **User DN**: DN of a user present in Good Technology User repository. This is a mandatory field.
- **Messaging Server**: Name of the Good Messaging server. If this field is left blank, IdentityIQ will locate a single GMM server if present. If multiple GMM servers exist, you must provide a value for this field else, the add operation will fail.
- **Policy Set**: Name of the Policy set.

Unlock device

While resetting a password for a device from IdentityIQ, default provisioning policy accepts Unlock Code displayed on the device. Upon successful reset password, temporary unlock code is generated and it will be saved in **Provisioning Request ID** field in Unlock Access Request in IdentityIQ.

Delete device

- **Delete Device operation from LCM**: In addition to delete device, you will be prompted to select **Entire Device** or **Enterprise Data Only** options before deleting the device.
- **Delete Device operation from Certification**: The default wipe operation will be the **Enterprise Data Only**. To change this default behavior to **Entire Device** add the following entry in the application debug of the device application:

```
<entry key="defaultWipeFromCertification" value="Entire Device"/>
```

Chapter 15: SailPoint HP ArcSight Integration Module

The following topics are discussed in this chapter:

Overview 97
Supported features 98
Pre-requisites 98
Configuration 98
Configuration to export IdentityIQ Data to ArcSight 98
Configuration to export IdentityIQ Data to ArcSight 98
Configuration to Import HP ArcSight CEF Flat File to SailPoint IdentityIQ 102

Overview

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

HP ArcSight ESM is a Universal log management solution that helps enterprises identify and prioritize current and potential security threats. SailPoint IdentityIQ collects the security event information such as Audit and Syslog information. The SailPoint IdentityIQ integration with HP ArcSight allows both end systems to take remediation action in case of security threats.

IdentityIQ integration with ArcSight enables the following scenarios:

1. Identity and Account data stored in IdentityIQ can be exported to ArcSight. ArcSight administrator can store this data in an ArcSight Active List.
2. IdentityIQ syslog and audit events can be exported to ArcSight for correlation, such as successful provisioning of privileged accounts, password changes, login failure etc.
3. IdentityIQ can import filtered activity event data from ArcSight based on which activity-based remediation processing can be triggered.

The integration with ArcSight uses a Syslog file format. Event records are expected in standard ArcSight Common Event Format (CEF). Events received are matched with users held within the IdentityIQ warehouse, and used to trigger activity policies when certain types of event are recognised. These triggers result in a business process being executed which generates a full re-certification for the affected user, and also causes a re-calculation of the user's risk score and update of risk reports and dashboard content to highlight the activity.

Note: Creating an ArcSight Active Channel or Active List is outside the scope of this document. This document assumes the ArcSight administrator is familiar with steps to create an ArcSight Active Channel or Active List. It provides the IdentityIQ information an ArcSight administrator will require to create an ArcSight Active Channel or Active List.

Common Event Format (CEF)

CEF is an extensible, text-based, high-performance format designed to support multiple device types in the simplest manner possible. CEF defines syntax for log records comprised of a standard header and a variable

extension, formatted as key-value pairs. CEF uses syslog as a transport mechanism and the following format, comprised of a syslog prefix, a header and an extension, as shown below:

For example,

```
Jan 18 11:07:53 host CEF:Version|Device Vendor|Device Product|Device
Version|Signature ID|Name|Severity| [Extension]
```

```
Dec 19 08:31:10 host CEF:0|Security|threatmanager|1.0|101|out of hours workstation
login|10|suser=hbutler src=activedirectorydomain ip=10.1.76.224
```

Supported features

- **Export from IdentityIQ to ArcSight**: IdentityIQ data such as Identity, Account, Audit and Syslog data can be exported from IdentityIQ and imported into ArcSight. The data can be exported to file in CEF or to database tables.
- **Import into IdentityIQ from ArcSight**: This integration supports including event logging data from ArcSight and associate it to Identities in IdentityIQ so that potential policy violations can be triggered or provide greater visibility as part of access reviews as to any suspicious or error prone access a user may have.

Pre-requisites

(*Applicable for import of ArcSight Events into SailPoint IdentityIQ*) At least one application must be configured in SailPoint IdentityIQ and Users/Groups aggregated into SailPoint IdentityIQ system.

Note: Users present in HP ArcSight must also be present in SailPoint IdentityIQ.

Configuration

This section describes the general, operation specific configurations and the steps that must be performed to configure the HP ArcSight Integration Module

Configuration to export IdentityIQ Data to ArcSight

The Identity, Account, Audit and Syslog information from SailPoint IdentityIQ can be exported to ArcSight using CEF flat file or database:

1. Export data from SailPoint IdentityIQ to ArcSight tables.
2. Export data from SailPoint IdentityIQ to Flat file in Common Event Format.

Export Data from SailPoint IdentityIQ to ArcSight tables

The **ArcSight Data Export** task enables you to export IdentityIQ and Audit data to external tables. You can select to export Identity information and Audit events from IdentityIQ Database.

Create the export databases on your destination datasource before using the ArcSight Data Export task.

1. Navigate to **Monitor => Tasks**.

2. Create a new ArcSight Data export task.
3. Provide the Data Source Parameters.
   ArcSight Data Export options are:

Options	Description
Datasource Parameters	
Database	Select a database type from the drop-down list.
User Name	Enter the user name parameter of the database.
Password	Enter the password of the database.
Driver Class	Enter the driver class used for the database.
URL	Enter the URL of the database.

4. Click on **Generate table Creation SQL** to generate table's schema and create database that includes export tables which you can hand off to a database administrator for execution.
   The task adds the following tables in database:

Tables	Description
sptr_arcsight_export	Table to maintain the task execution history.
sptr_arcsight_identity	Table contains exported data of Identity.
sptr_arcsight_audit_event	Table contains Audit Events information.

5. Select **Object Export** options.
   The **Object Export** options are:

Options	Description
Export Identities	Select the check box to export Identity related data in ArcSight tables. It provides the following options: • **Full**: Exports all the records irrespective if they were exported earlier. • **Incremental**: Exports only records that are updated since last run of this task. This option can even be selected when running the task for first time. When the task is running for first time, this option exports all records similar to the **Full** option.
Export Audits	Select the check box to export Audit Events in ArcSight table. It provides the following options: • **Full**: Exports all the records irrespective if they were exported earlier. • **Incremental**: Exports only records that are updated since last run of this task. This option can even be selected when running the task for first time. When the task is running for first time, this option exports all records similar to the **Full** option.

6. After completing the customizing report options, click **Save** for later use or Save and Execute to save the report and run it immediately.

Configuring HP ArcSight Task to populate host name or IP

The value of column application_host can be populated by adding a map **arcsightAppNameHostMap**. Adding the **arcsightAppNameHostMap** map the administrator configuring this integration can define the hostname (or IP address) which must be used for an Account. It is recommended this hostname (or IP address) is same as the configured in the ArcSight configuration.

The **arcsightAppNameHostMap** map must be defined in the **ArcSight Data Export** Task created above. The key in the map should be name of the application defined in IdentityIQ and value should be hostname, IP, or any string that ArcSight administrator understands.

1. To add the map go to debug page, navigate to TaskDefinition and open the ArcSight task configured above.
2. Add the entry as key = Name of Application defined in IdentityIQ and value as the string to identify host of Account like Hostname or IP.
3. Save the task definition.
   For example:

```
<entry key="arcsightAppNameHostMap">
    <value>
        <Map>
          <entry key="LinuxApp1" value="linux01.sailpoint.com"/>
          <entry key="LinuxApp2" value="127.15.19.21"/>
          <entry key="ADDirectApp" value="AD.sailpoint.com"/>
          <entry key="ServiceNowApp" value="https://sailpoint.service-now.com"/>
          <entry key="ACF2App" value="ACF2-Mainframe"/>
        </Map>
    </value>
</entry>
```

Note: If the application name is not defined in the map the host field will be blank.
As mentioned above, this document provides the information an ArcSight administrator requires to create an ArcSight Active List or Active Channel. The information below provides the same. Following fields are added in export table:

Table 1—IdentityIQ sptr_arcsight_identity export table

Fields	Description
linkid	Primary key for Link table in IdentityIQ database. This field will be copied from spt_link table id field. This will be the primary key for export table.
identityid	Primary key in Identity table. This field will be copied from spt_Identity table.
modified_dt	Populates timestamp when the record will be exported in export table. The field can be referred while configuring time based ArcSight database connector.
identity_display_name	Represents Display Name of Identity which will be copied from spt_identity table field (display_name).
identity_firstname	Represents first name of Identity which will be copied from spt_identity table field (firstname).
identity_lastname	Represents last name of Identity which will be copied from spt_identity table field (lastname).

Table 1—IdentityIQ sptr_arcsight_identity export table

Fields	Description
application_type	Populates the type of Account which is connected to the Identity like ActiveDirectory – Direct, ACF2 – Full, Box, Cloud Gateway, ServiceNow and so on.
application_host	The host name, IP, or any string which can be used by ArcSight administrator to identify the host of link/account uniquely. Customer can enter any string which can be sent to ArcSight to identify the host of link. This field can be populated as explained in “Configuring HP ArcSight Task to populate host name or IP” on page 100.
application_name	Populates the name of Application of the Account connected to the Identity.
link_display_name	The account connected to the identity which will be copied from spt_link table, field display_name.
entitlements	Represents comma separated list of entitlements to the link of Identity.
risk_score	Represents the composite risk score of Identity.

Table 2—IdentityIQ sptr_arcsight_audit_event export table

Fields	Description
auditid	The audit ID which is primary key for the export Audit table. The field will be copied from spt_audit_event table id field.
created_dt	Populates timestamp when the record will be exported in export table. The field can be referred while configuring time based ArcSight database connector.
owner	Describes the Owner of the audit generated.
source	Provides more details to help ArcSight administrator determine the source of audit.
action	Describes the action taken on entity.
target	Provides target details.
application	Describes the name of application the target belongs to.
account_name	The name of Account is populated in this field.
attribute_name	The name of attribute modified.
attribute_value	The value provided to the attribute.

Export Data from SailPoint IdentityIQ to Flat file

1. Navigate to **Analyze => Advanced Analytics**.
2. Navigate to Identity Search/ Audit Search/ Syslog Search/ Account Search Tab.
3. Select the Search Criteria and Fields to display.
4. Click on **Run Search**.
5. Click on **CEF Flat file** export button to export search results to file in CEF.

The Search Results page have the following options to save:

- **Save Search**: It is used to save the search criteria and fields to display.
- **Save Search as Report**: These type of reports can be accessed as a report, as schedules or for execution by performing the procedure:
  a. Navigate to **Analyze => Reports**.
  b. Right click on the report and schedule or execute the report.
  c. Navigate to Report Results tab to see the report result.
  d. Click on the **Report**.
  e. Click on the **CEF Flat file** export button to export the report to file in CEF.

     This will generate a file with data in CEF which can be used by ArcSight to import events in ArcSight ESM.

Note: For more information on Advanced Analytics, see *SailPoint IdentityIQ User Guide* version 6.4.

Configuration to Import HP ArcSight CEF Flat File to SailPoint IdentityIQ

1. Access the Application Configuration Console.
2. Navigate to Schema tab.
   Mark the field as correlation key for which you want to correlate activity from HP ArcSight to SailPoint IdentityIQ.

   For Example: `sAMAccountName` for Active Directory application.
3. Navigate to Activity Data Sources tab.

   Note: For more information on Activity Data Source, see *SailPoint IdentityIQ Administration Guide* version 6.4.
4. Click on **Add** to add new Activity Data Source.
5. Select Activity Data source Type as **CEF Log File**. The default Transformation rule and Correlation rule will be automatically selected.

   Note: You can change the value of `cefLinkAttribute` in correlation rule to set correlation key as per application.
6. Navigate to Transport Settings tab, select the Transport Type as local, ftp or scp.
7. Navigate to Log File Settings tab and in the File name provide the exact path of the CEF Flat file. For example, `C:\ArcSight\activedirectory.csv`
8. Click on **Save** button to save activity data source configuration.
9. Click on **Save** button to save the application.

If the correlation key is not marked and aggregation of account for that application is already performed, then perform the following:

- Access the Application Configuration console.
- Navigate to the Correlation tab.
- Click on **New** button to create a new Account Correlation.
- Click on next button and provide the name of the configuration.
- Select the Application Attributes and Identity Attributes and click on **Add** button.
- Click on **Save**.
- Click on **Save** to save the application.

  After the correlation configuration is done, execute the account aggregation (with optimization turned off to pick up the existing accounts) again.

10. Navigate to **Define => Identities**.
11. Click on the identity for which you want to enable Activity monitoring and import data from ArcSight.
12. Navigate to Activity Tab.
13. Select the **Activity Monitoring** checkbox.
14. Save the Identity.
15. Navigate to **Monitor => Tasks**.
16. Create a new Activity Aggregation Task.
17. Select an activity data source which is configured above in Step 8.
18. Save and execute the task.
    - To see the result of the task executed in previous step navigate to Task Results tab and click on the task.
    - To see the correlated events navigate to **Define => Identities**. Select the identity for which you have correlated the event. Navigate to Activity Tab. Check the Recent Activities section.

Note: After correlating the HP ArcSight event to Identity, the Policy Violation and Certification can be created and used to notify for any activity for that identity using the workflow.

Chapter 16: SailPoint Lieberman Integration

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

IdentityIQ Lieberman Integration can be used to manage Privileged Users in various target systems. Lieberman Enterprise Random Password Manager (ERPM) is used to manage password of privileged accounts on various target systems. Using this integration IdentityIQ Certifiers can certify the access and permissions assigned to the privileged users.

The IdentityIQ Lieberman Integration is built upon IdentityIQ Active Directory application which must be configured to use this integration. After configuring IdentityIQ Active Directory, Lieberman Target Collector must be configured on the same. The Lieberman Target Collector aggregates permissions assigned to Domain Users and Domain Groups (Active Directory User/Groups) on different management sets in Lieberman ERPM. The aggregated permissions are associated with the Accounts and Groups associated with the Active Directory application.

This document assumes IdentityIQ Active Directory application has been configured, and lists the steps to configure Lieberman Target Collector. For more information on configuring IdentityIQ Active Directory application, see *SailPoint Direct Connectors Administrator and Configuration Guide*.

Lieberman Target Collector

Lieberman target collector requires **msDS-PrincipalName** field of Active Directory account and group to be marked as Correlation Key.

The unstructured target defined on **Unstructured Targets** tab of application would be used by the Target Aggregation Task to correlate targets with permissions assigned to Identities and Account Groups.

These permissions can be revoked as part of certification remediation process, the certifier of the Identity or Entitlement can revoke these permission.

The Lieberman Target Collector configuration parameters are as follows:

Field	Description
Authenticator	The Authenticator of the user. The different Authenticators supported by IdentityIQ are: • **Explicit User***:* The user is Lieberman user and will be authenticated by Lieberman. • **Domain User***:* The user is Domain user and will be authenticated by the Active Directory.
URL*	URL to the web service.
Username*	Username of the Lieberman Administrator. It should be prefixed with domain name as (Domain\Username) in case of Domain User authenticator.
Password*	Password of the Lieberman Administrator.

Field	Description
Rules: Specify the rule used to correlate the targets with appropriate AD accounts. **Note: Click the "..." icon to launch the Rule Editor to make changes to your rules if needed.**	
Correlation Rule	Used to determine how to correlate target permission information to Active Directory users in IdentityIQ. Default correlation rule for Lieberman can be found in **targetRuleLibraries.xml** with **Lieberman Default Target Correlation For Active Directory Application** name**.**

Troubleshooting

1 - Target Aggregation displays the time out exception error for larger data.

Target Aggregation displays the following timeout exception error for larger data:

```
Target Source Scan failed. Reason: openconnector.ConnectorException: Exception
occurred while processing permissions,check the configuration: Read timed out
```

Resolution: Add the following in Target Source on debug page and set the timeout value.

```
<entry key="clientOptions">
  <value>
    <Map>
      <entry key="CONNECTION_TIMEOUT">
        <value>
          <Integer>180000</Integer>
        </value>
      </entry>
      <entry key="SO_TIMEOUT">
        <value>
          <Integer>180000</Integer>
        </value>
      </entry>
    </Map>
  </value>
</entry>
```

Note: Above Time Out value can be change as per requirement(s).

Chapter 17: SailPoint STEALTHbits Integration

Note: SailPoint will provide assistance during the deployment of this Integration Module. Additional troubleshooting, diagnostic, and best practice information beyond what is contained in this document will be provided in the Connector and Integration Deployment Center on Compass.

StealthBits Integration can be used to read the effective permissions for SharePoint and Windows FileShare. The effective permissions on SharePoint Resources and Windows Fileshare can be aggregated and associated to respective Account and Group in IdentityIQ.

The IdentityIQ StealthBits Integration is built upon IdentityIQ application. An IdentityIQ application must be configured to use this integration. This integration supports the following applications:

- **Active Directory application**: to aggregate the effective permissions assigned to Accounts associated with the IdentityIQ Active Directory application on SharePoint Folders, Files and Windows Fileshares.
- **SharePoint application**: to aggregate the effective permissions assigned to Accounts and SharePoint Groups associated with the IdentityIQ SharePoint application on SharePoint Folders and Files.
- **Windows Local application**: to aggregate the effective permissions assigned to Accounts and Groups associated with the IdentityIQ WinLocal application on Windows Fileshares.

After an IdentityIQ application has been configured, SteathBits Target Collector must be configured.
This document assumes one of the above mentioned three IdentityIQ applications has been configured, and lists the steps to configure StealthBits Target Collector on the configured IdentityIQ application. For more information on configuring any of the above mentioned IdentityIQ applications, see *SailPoint Direct Connectors Administrator and Configuration Guide*.

STEALTHbits Target Collector

STEALTHbits Target Collector is designed to read unstructured data from STEALTHbits Technologies' StealthAUDIT Management Platform. StealthAUDIT Management Platform (SMP) is an IT application which provides enterprise wide data collection capability for data repositories such as Microsoft SharePoint and Windows File System.

STEALTHbits target collector configuration

The following table lists the STEALTHbits target collector configuration parameters:

Field	Description
Managed System*	Managed system for which the data needs to be read from STEALTHbits.
URL*	Database url of STEALTHbits storage database for the selected system. For JDBC driver the url format is as follows: jdbc:sqlserver://<*serverName*>\<*instanceName*>:<*portNumber*>;databaseName=<*databaseName*>
UserName*	Login user name with read permissions on STEALTHbits database.
Password*	Password of above mentioned user.

Field	Description
Driver Class	JDBC driver class. Default: com.microsoft.sqlserver.jdbc.SQLServerDriver
Filter	Filter is basically **Where** clause that is appended to default queries to skip undesired data. For example, while configuring STEALTHbits collector for Windows Local application, the filter should be set to **[HOST]='HostName'**.
SharePoint Site Collection URL	Applicable only when selected Managed System is a 'Microsoft SharePoint'. Takes URL of the SharePoint Site Collection to be managed.
MSSQL User Query	Microsoft SQL query to get the data for users. If not provided, default query for selected managed system is used.
MSSQL Group Query	Microsoft SQL query to get data for groups. If not provided, default query of selected managed system type is used.

Note: In the above table the * sign indicates the mandatory attributes.

Supported Managed System

STEALTHbits System	Windows FileShare
IdentityIQ Application	• Active Directory • Windows Local-Direct
Correlation Key	objectSID for users and groups
Correlation Rule	WindowsSID Target Correlator
The collector aggregates Effective Permissions. **Note: While configuring this collector with the Windows Local Connector, setting 'filter' value to "[HOST]='Hostname'" may be considered where hostname is the server that the application is managing.**	

STEALTHbits System	Microsoft SharePoint
IdentityIQ Application	• Active Directory • Microsoft SharePoint
Correlation Key	• For Active Directory application **For Users**: msDS-PrincipalName **For Groups**: msDS-PrincipalName • For Microsoft SharePoint application **For Users**: AccountName **For Groups**: DisplayName
Correlation Rule	• For Active Directory application: AD STEALTHbits Correlator for SharePoint • For Microsoft SharePoint application: SharePoint Default Target Correlation

STEALTHbits System	Microsoft SharePoint
Note: The collector aggregates Effective Permissions. Automatic revocation is not supported for Effective Permissions. To create a work item when a permission is revoked, ensure that the application has "NO_PERMISSIONS_PROVISIONING" and "NO_GROUP_PERMISSIONS_PROVISIONING" feature string.	

Configuring custom queries

The Microsoft SQL User Query and Microsoft SQL Group Query attributes provide ability to customize queries that are used to read permissions from STEALTHbits system. The collector expects TargetName, Rights and NativeID as column names from the query result where:

- **TargetName** describes the name of the resource on the end system.
- **NativeID** describes the user identity which is also marked as the co-relation key in IdentityIQ application.
- **Rights** describes the permissions the user or group have on target.

Rights Mapping

The STEALTHbits Target Collector maps the rights read from STEALTHbits to appropriate rights to achieve permissions revocation functionality supported by IdentityIQ connectors. This mapping can be overwritten by assigning custom mapping to the **rightsMapping** attribute in target source attributes.

For example,

```
<entry key="rightsMapping">
        <value>
           <Map>
              <entry key="A" value="Admin"/>
              <entry key="D" value="Delete"/>
              <entry key="DM" value="Delete,Manage"/>
              <entry key="WDM" value="Write,Delete,Manage"/>
              <entry key="WM" value="Write,Manage"/>
           </Map>
        </value>
</entry>
```

In above example, the entry key represents rights in STEALTHbits application and value represents corresponding mapping in IdentityIQ.

If permissions are desired to be displayed on IdentityIQ as they appear in STEALTHbits system, set the **useSTEALTHbitsRights** attribute value to **True** in the target source attributes as follows:

```
<entry key="useSTEALTHbitsRights" value="True"/>
```

Rights mapping configured using **rightsMapping** or **useSTEALTHbitsRights** might cause connector failure while processing revocation of target permissions.

Note: **The** `rightsMapping` **and** `useSTEALTHbitsRights` **flag are not supported for the managed system of type Microsoft SharePoint.**

Appendix

This section contains information on the following:

- "A: Common Identity Management Integration Configuration" on page 113

Appendix A: Common Identity Management Integration Configuration

This appendix describes the following information.

Overview 113
Creating the IntegrationConfig Object 113
Provisioning 118

Overview

This appendix describes configuration process for integrations with identity management (IDM) systems and the places in IdentityIQ that use the integrations. It does not describe the details of a specific integration only the general framework common to all integrations.

Creating the IntegrationConfig Object

The first step in configuring an integration is designing an instance of the IntegrationConfig object. There is currently no user interface for editing these objects, you must write them in XML and import them. The IntegrationConfig defines the following things:

- Java class that handles communication with the IDM system
- Connection parameters such as host name, user name, and password
- IdentityIQ Application object that represents the IDM system in aggregations
- List of the applications that are managed by the IDM system
- Resource and attribute name mappings
- Role synchronization style
- Methods for selecting roles to synchronize

Here is an example of `IntegrationConfig` file that has all of the options:

```
<IntegrationConfig name='Example Integration'
   executor='sailpoint.integration.ExampleIntegration'
   roleSyncStyle='it'>

<!--
  Application representing the IDM system in IIQ
-->
<ApplicationRef>
  <Reference class='Application' name='Example Integration'/>
</ApplicationRef>

<!--
  Connection parameters needed by the executor.
-->

<Attributes>
  <Map>
```

```
    <entry key='url' value='http://somehost:8080/rest/iiq'/>
    <entry key='username' value='jlarson'/>
    <entry key='password' value='1:987zxd9872970293874'/>
  </Map>
</Attributes>

<!--
  Definitions of managed resources and name mappings.
-->
<ManagedResources>
 <ManagedResource name='LDAP 42'>
  <ApplicationRef>
   <Reference class='Application' name='Corporate Directory'/>
  </ApplicationRef>
  <ResourceAttributes>
   <ResourceAttribute name='memberOf' localName='groups'/>
  </ResourceAttributes>
 </ManagedResource>
</ManagedResources>

<!--
  Synchronized role list.
  In practice you will never have a SynchronizedRoles with
  the RoleSyncFilter or RoleSyncContainer elements. All three
  are shown only as an example.
-->
<SynchronizedRoles>
  <Reference class='Bundle' name='role1'/>
  <Reference class='Bundle' name='role2'/>
</SynchronizedRoles>
```

The executor attribute has the name of a class that implements the sailpoint.object.IntegrationExecutor interface. This class is conceptually similar to a Connector class in that it does the work specific to a particular integration. Each integration package will come with an example IntegrationConfig that contains the executor class name.

The roleSyncStyle attribute defines how roles are synchronized between IdentityIQ and the IDM system. The possible values are:

- **none**: roles are not synchronized
- **detectable**: detectable (IT) roles are synchronized
- **assignable**: assignable (business) roles are synchronized
- **dual**: both detectable and assignable roles are synchronized

If this attribute has no value the default is none. More information on the role synchronization process is found in the Role Synchronization section.

ApplicationRef

Some integrations support identity aggregation. In these cases there is a **sailpoint.object.Application** object defined to represent the IDM system and an implementation of the **sailpoint.connector.Connector** interface that handles communication with the IDM system. This is normally a multiplexed connector that returns objects representing the IDM system account as well as accounts on managed resources. Links in the identity cube are created for the managed resource accounts as well as the IDM system account.

```
<ApplicationRef>
  <Reference class='Application' name='Example Integration'/>
```

```
</ApplicationRef>
```

The documentation of each integration must describe the supported configuration attributes.

The following attributes are reserved and can only be used for the purposes defined here.

- **roleSyncHistory**: list of objects containing a history of previous synchronizations
- **universalManager**: enables the integration as a manager of all applications

The **roleSyncHistory** attribute contains a list of **sailpoint.object.IntegrationConfig.RoleSyncHistory** objects that have information about roles previously synchronized with this integration. This includes the name of the role and the date it was synchronized. This list can be used by the role synchronization task to optimize communication with the IDM system by sending only the roles that have changed since the last synchronization.

The **universalManager** attribute is set to the string true to enable this integration as a manager for all IdentityIQ applications without a ManagedResources list. This can be helpful in test environments to validate deployment configuration as well as environments where all provisioning must be fulfilled by a single integration.

ManagedResources

If the integration supports provisioning, it must define a list of managed resources that corresponding to applications defined in IdentityIQ. This determines how provisioning plans created during certification or role assignment are divided and sent to each integration.

```
<!--
    Definitions of managed resources and name mappings.
  -->
  <ManagedResources>
   <ManagedResource name='LDAP 42'>
    <ApplicationRef>
      <Reference class='Application' name='Corporate Directory'/>
    </ApplicationRef>
    <ResourceAttributes>
     <ResourceAttribute name='memberOf' localName='groups'/>
    </ResourceAttributes>
   </ManagedResource>
  </ManagedResource>
```

The ManagedResources element contains a list of ManagedResource elements. A ManagedResource element must contain an ApplicationRef that defines the associated IdentityIQ application. The ManagedResource element might have an optional name attribute that defines the name of the resource within the IDM system. If the name is not specified it is assumed that the resource name is the same as the IdentityIQ application name.

The ManagedResource element might also contain a ResourceAttributes element that contains one or more ResourceAttribute elements. ResourceAttribute is used to define mappings between attribute names in the IDM system and IdentityIQ. ResourceAttribute has the following XML attributes.

- **name**: attribute name in the IDM system
- **localName**: attribute name in the IdentityIQ application schema

If a provisioning plan is sent to this integration with attributes that are not in the ResourceAttributes list it is assumed that the name in IdentityIQ is the same as the name in the IDM system.

The ResourceAttributes list does not define a filter for attributes sent to the IDM system it only defines name mappings. When an integration has an IdentityIQ application in the ManagedResource list it is assumed that all attribute requests for that application are sent to that integration. You cannot have more than one integration managing different sets of attributes for the same application.

There is a special attribute that can be defined in Attributes that declares the integration as the manager of all applications in IdentityIQ regardless of the content of the ManagedResources element. You can still use ManagedResources to define name mappings for certain applications when necessary.

SynchronizedRoles

The SynchronizedRoles element is used to define a concrete list of roles to consider when roles are synchronized. When this is included in an IntegrationConfig object it has priority over both RoleSyncFiler and RoleSyncContainer elements if they are also included.

```
<SynchronizedRoles>
  <Reference class='Bundle' name='role1'/>
  <Reference class='Bundle' name='role2'/>
</SynchronizedRoles>
```

This can be used to synchronize simple integrations with a small set of roles that does not change often. If the set of roles is large or frequently changing, it is better to use RoleSyncFilter or RoleSyncContainer.

RoleSyncFilter

The RoleSyncFilter element contains a filter that is used to identify the roles to consider for synchronization.

```
<RoleSyncFilter>
  <Filter property='syncFlag' operation='EQ' value='true'/>
</RoleSyncFilter>

<RoleSyncFilter>
  <Filter property='name' operation='LIKE' matchMode='START'
value='Sync'/>
</RoleSyncFilter>
```

Synchronization filters typically used extended role attributes. In the first example an extended attribute syncFlag must be configured and have a value of true for the role to be synchronized.

Naming conventions can also be used to identify synchronized roles. In the second example any role whose name starts with Sync is synchronized.

A RoleSyncFilter can be combined with a RoleSyncContainer. If both are specified the intersection of the two role sets is considered for synchronization.

RoleSyncContainer

The RoleSyncContainer element defines the set of roles to be considered for synchronization by identifying an inherited role.

In this example, any role that directly or indirectly inherits the role named Roles To Synchronize is synchronized. This is typically a container role that has no function other than organizing other roles.

Specifying roles with inheritance has the advantage of creating a node in the modeler tree for Roles To Synchronize that you can expand to quickly see all of the synchronized roles.

A RoleSyncContainer can be combined with a RoleSyncFilter. If both are specified the intersection of the two role sets are synchronized.

Note: If an IntegrationConfig does not have any SynchronizedRoles, RoleSyncFilter, or RoleSyncContainer elements and the roleSyncStyle element has a value other than none, it is assumed that all roles are considered for synchronization.

Aggregation

Some integrations support feeds of identity information through the normal aggregation process. In these cases the integration package will have a SailPoint.connector.Connector implementation class and an example SailPoint.object.Application object in XML.

IDM connectors are usually multiplexed connectors that return objects representing the IDM system account as well as accounts on all managed resources.

When an aggregation application is defined a reference to it should be placed in the IntegrationConfig. This enables provisioning operations to obtain the account name in the IDM system that corresponds to an identity in IdentityIQ.

Role Synchronization

Role synchronization is performed by running the standard system task named Synchronize Roles. This task is defined in the file `tasksRunnable.xml` and is created during the normal initialization and upgrade processes.

The task normally attempts synchronization with every **IntegrationConfig** stored in the repository. The task has one hidden input argument, integrations, that can be set to a CSV of names of IntegrationConfig objects. Use this to restrict the synchronization to a particular set of integrations.

Each **IntegrationConfig** has a **roleSyncStyle** attribute that determines how roles are synchronized. If this attribute is missing or set to none, role synchronization is disabled.

When synchronization is enabled, the task first determines the set of candidate roles by evaluating the filtering options defined in the IntegrationConfig. If there is a SynchronizedRoles element it defines the concrete list of candidate roles. Otherwise the RoleSyncFilter and RoleSyncContainer are evaluated and intersected to produce the set of candidate roles.

Note: Candidate roles are not necessarily the ones that are sent to the IDM system. The roles sent are further constrained by the synchronization style.

roleSyncStyle=detectable

When the synchronization style is detectable the candidate role list is filtered to contain only detectable roles as defined by the role type.

For each candidate role a simplified representation of the role is built using the SailPoint`.integration.RoleDefinition` class, this is called the target role.

Entitlements for the target role are extracted from the candidate in one of two ways. If the candidate role has a provisioning plan, the plan defines the resources and attribute values that are included in the target role. If the candidate role has no provisioning plan, a set of resource attributes is derived by analyzing the profile filters in the candidate role.

To give a role a provisioning plan, design an instance of the SailPoint**.object.ProvisioningPlan** as an XML and place it inside the XML for the SailPoint`.object.Bundle` object representing the role.

Using provisioning plans gives you more control over the contents of the target role. Profile filters might be ambiguous or result in more attribute values for the role than are necessary, but for relatively simple filters it can be easier than defining provisioning plans.

To derive target roles, first build a list of candidate profiles. If the role option or Profiles is false, than all profiles defined in the role become candidates. If the or Profiles option is true, then only the first profile in the role is a candidate. Then iterate over each filter in each candidate profile applying this algorithm:

```
if the filter is EQ or CONTAINS_ALL
```

```
  add the values for this attribute comparison to the role
if the filter is OR
  recurs for the first child filter term
if the filter is AND
  recurs for all child filter terms
```

If the role inherits other roles, the hierarchy is flattened and inherited entitlements are merged into the target role. The same process described above is applied to every role in the inheritance hierarchy.

roleSyncStyle=assignable

When the synchronization style is assignable the candidate role list is filtered to contain only assignable roles as defined by the role type.

For each candidate role, build a simplified representation of the role using the SailPoint.integration.RoleDefinition class, this is called the target role.

Entitlements for the target role are extracted from the candidate by first applying the process described in roleSyncStyle=detectable to the candidate role. This might not have any effect since assignable roles do not normally have provisioning plans or profiles.

Next the roleSyncStyle=detectable process is applied to each of the required roles referenced by the candidate role. This is typically where most of the entitlements are found.

roleSyncStyle=dual

This is a hybrid of the assignable and detectable styles used only by the IBM Tivoli Identity Manager integration.

First detectable roles are synchronized as defined in the <ZBlueXref>“roleSyncStyle=detectable” on page 117 section.

Next assignable roles are synchronized. Instead of entitlements the roles have an extended attribute containing the names of all roles that were on the required list. The integration executor might further annotate the role definition with rules for automated assignment.

Provisioning

Provisioning can be performed in several ways.

- After role assignment from the IdentityIQ identity edit page
- After role assignment from the Access Request Manager
- During certification to handle revocations and role completions
- In a background reconciliation task
- During aggregation

Both IdentityIQ and the Access Request Manager (ARM) launch workflows with provisioning being done at the end. This provides the opportunity to insert an approval step before provisioning. The default workflow for IdentityIQ identity edits is named Identity Update. By default it has no approvals but does attempt provisioning. The example workflow for ARM requests is named ARM Role Approval Example.

Certifications can do provisioning to remove entitlements and roles that were revoked as well as add missing entitlements that are necessary to satisfy a role assignment.

A reconciliation task is an instance of the Identity Refresh task template with the provisioning argument set to true. This argument is visible in the configuration page for the refresh task. Reconciliation compares the assigned

roles with the detected entitlements and automatically provisioning any missing entitlements. Entitlements might be missing due to either changes in role assignments for an identity, or changes to the definition of roles already assigned to an identity.

Reconciliation is intended to replace the IdentityIQ Provisioning. The old provisioning page was role oriented, monitored changes to roles, and sent provisioning requests for users assigned to modified roles. It did not detect changes to the assigned roles list of identities, however. The reconciliation task is identity oriented and calculates all changes necessary to make an identity's entitlements match the currently assigned roles.

Since reconciliation is now part of the core set of identity refresh options, it can also be done during aggregation. This is less common, but aggregation could change account attributes that are used by role assignment rules resulting in changes to the assigned and detected role lists. With provisioning enabled, the aggregation could trigger the provisioning of missing entitlements for the assigned roles. A common use case for this would be aggregating from an application representing a HR system with HR attributes determining assigned business roles.

Note: Automated provisioning done by the reconciliation task or within workflows typically does not remove entitlements, it only adds missing entitlements. Removal of unnecessary entitlements is expected to be done in a certification where a user has more control. While it is possible to enable removals during automated provisioning, it is potentially dangerous and should not be done without careful consideration.

Provisioning with Synchronized Roles

The provisioning sub-system works with the role synchronization sub-system to determine how role assignments are provisioned. If roles are not being synchronized, the raw entitlements needed by that role are compiled and sent to the integration executors. If an integration supports role synchronization, requests are compiled to the IDM system itself to add or remove native role assignments.

There might be median cases where an integration does role synchronization, but only manages a subset of the possible applications. In these cases plans are compile with both IDM system role assignments and as raw entitlements for the entitlements that are not covered by a native role assignment.