Fundamentals of IdentityIQ Implementation

Training for SailPoint IdentityIQ Version 7.0

11305 Four Points Drive
Bldg 2, Suite 100
Austin, TX 78726
www.sailpoint.com

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

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

Section One: Installation, Identity Cubes, and Onboarding Applications

Fundamentals of IdentityIQ Implementation

Training for SailPoint IdentityIQ Version 7.0

11305 Four Points Drive
Bldg 2, Suite 100
Austin, TX 78726
www.sailpoint.com

Contents

Course Overview

Introduction

The exercises contained in this document are meant to accompany the Fundamentals of IdentityIQ Implementation training lecture materials.

These exercises are run within a Virtual Machine environment, which contains the following software:

- Oracle/Sun JDK (Version 1.7)
- Tomcat Application Server (Version 7.0.59)
- MySQL Database Server (Version 5.5.42)
- OpenLDAP Server (Version 2.4.40)
- Apache Directory Studio (Version 2.0.0)

During these Implementer Training exercises, we will be installing and configuring the following:

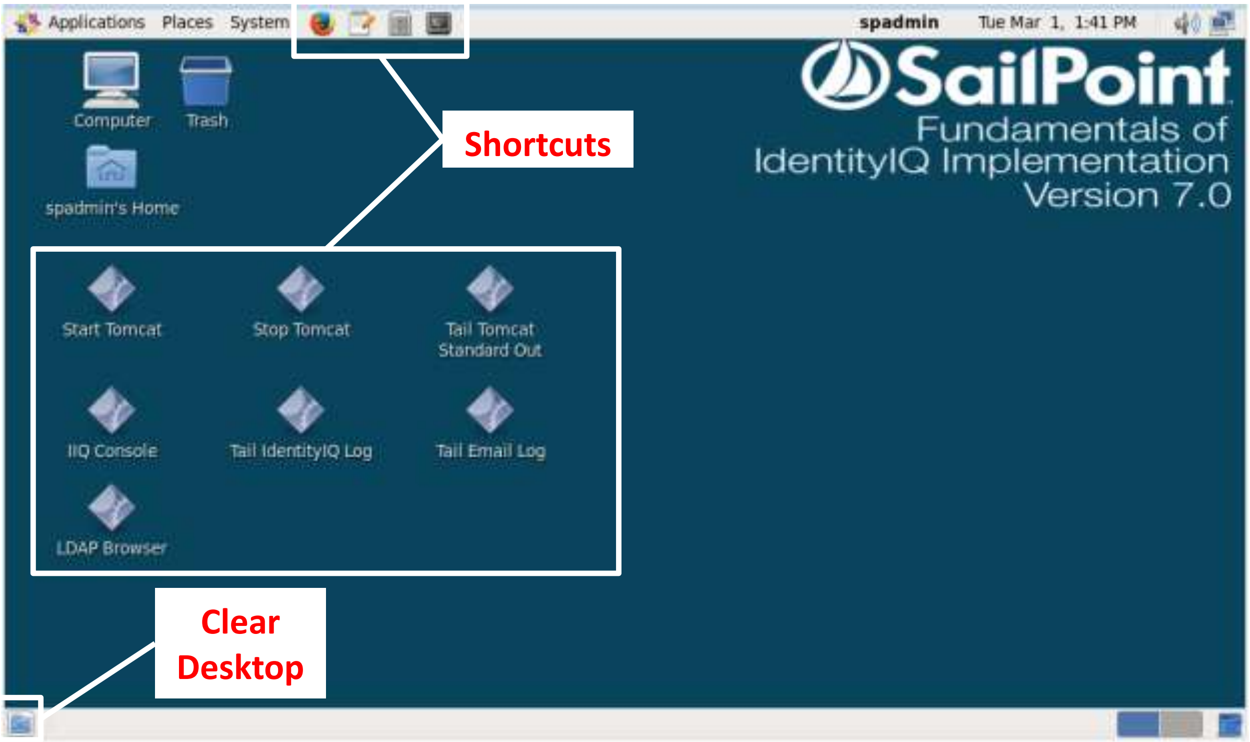
- IdentityIQ Version 7.0

The Virtual Machine Environment

Logins	
Linux Username/Password	spadmin/admin
IdentityIQ Username/Password	spadmin/admin
MySQL Administrator Login/Password	root/root
OpenLDAP Login	User: cn=Manager,dc=training,dc=sailpoint,dc=com Password: password
Install Directories	
IIQ Install Directory	/home/spadmin/tomcat/identityiq
Installer File Location	/home/spadmin/InstallImages
Implementer Training Files	/home/spadmin/ImplementerTraining
LDAP Install Location	/home/spadmin/openldap
MySQL Details	
MySQL Database Name	identityiq

Shortcuts/Applications Provided

The Virtual Machine environment includes several useful shortcuts.



- Application Shortcuts
  - File Browser (Linux utility to browse the file system)
  - Editor (gedit – A common Linux text editor)
  - Firefox (Web browser)
  - Terminal (Launches a command line terminal)
- Launcher Shortcuts on the Desktop
  - Launchers to Start/Stop Tomcat
  - Launcher to start the IdentityIQ Console
  - Launchers to observe the IdentityIQ Logs, IdentityIQ Email Logs, Standard Out Logs
  - Launcher to start ApacheDirectoryStudio LDAP Browser
- Clear Desktop – Use this to minimize windows to see the Desktop

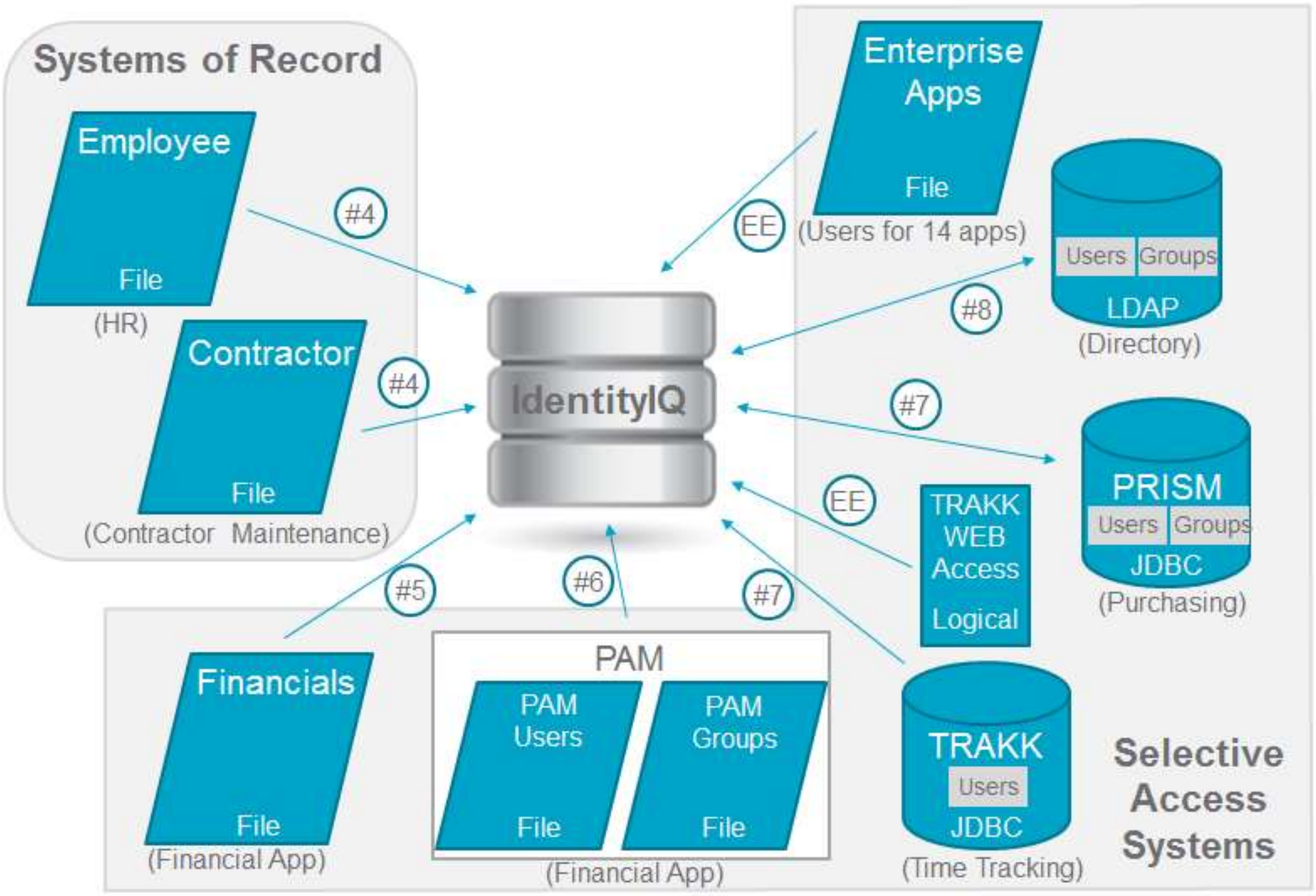
Section 1: Installing, Identity Cubes, Onboarding Applications

In this section, we will be setting up our system, loading Identity data into the system and onboarding account and group data from different applications.

- Install IdentityIQ
- Patch IdentityIQ
- Configure certain IdentityIQ administrative features
    - Redirect Emails to a file
    - Enable Logging
    - Configure certain audit events
- Onboard Identity information from authoritative (systems of record) application sources
    - Employees
    - Contractors
- Onboard additional account and group data from additional non-authoritative (systems of interest) application sources
    - Flat File (CSV) data feeds containing user accounts and group data
    - JDBC data feeds containing users accounts and group data
    - LDAP system containing user accounts and group data
    - Logical Application
    - Mulitplexed Application

The diagram on the following page provides a visual representation of the systems which will be onboarded in this section and utilized throughout this course. The numbers correspond to the Application onboarding exercises.

Exercise Environment – Systems



Exercise #1: Installing IdentityIQ

Objective

In this exercise, we will install and configure IdentityIQ.

Overview

Our training scenario represents a typical implementation cycle with a customer. The client has provided us with the following:

- A running database server with host, port and login information provided
- A pre-configured Tomcat Application Server instance

We need to install IdentityIQ with the following requirements.

- Install IdentityIQ into the **/identityiq** directory in Tomcat.
- Adjust the IdentityIQ Hibernate files to support our installation. Our installation needs to support the following:
    - 2 named extended identity attributes
    - 10 searchable and indexed placeholder extended identity attributes
- Generate the IdentityIQ database schema files and use these to create the IdentityIQ database within the MySQL database instance
- Initialize IdentityIQ
- Start the application server
- Confirm that everything is running okay

Assess Your Implementation Role

1. Will you be *responsible for installing IdentityIQ* in a development, test, or production environment? Yes / No (Circle one)

If No:

- Use the scripts provided in this virtual machine to complete the installation. See Appendix, page 2, **Scripted IdentityIQ Installation** for instructions.

If Yes:

- Continue to next step: **Prepare Application Server and Install IdentityIQ War File**.

Prepare Application Server and Install IdentityIQ War File

1. Stop the Tomcat Application Server

   a. From the desktop, run the shortcut labeled **Stop Tomcat** or

   b. Alternatively, you can open a Linux terminal window and type the following command:

   ```
   StopTomcat
   ```

   Note: If this is the first time this VM has been used, Tomcat should already be stopped.

2. Unzip and extract the IdentityIQ war file

   a. Open a Linux terminal window and navigate to the directory: **/home/spadmin/InstallImages.** At the $ command prompt, enter:

   ```
   cd InstallImages
   ```

   Note: For help navigating in Linux, see Basic Linux Commands, Appendix-1.

   b. Confirm that the IdentityIQ zip file is in the directory. Enter the following command to view the contents of the directory.

   ```
   ls
   ```

   c. Unzip the **IdentityIQ-7.0.zip** file:

   ```
   unzip identityiq-7.0.zip
   ```

   d. Within the InstallImages directory, locate the **identityiq.war** file and copy it to the installation directory for IdentityIQ: **/home/spadmin/tomcat/webapps/identityiq**

   Copy Options:

   Option 1: Use the file browser to copy and paste the file.

   Option 2: Use the Linux copy command. At the $ command prompt, enter:

   ```
   cp identityiq.war /home/spadmin/tomcat/webapps/identityiq
   ```

   e. In a command window, navigate to the **/home/spadmin/tomcat/webapps/identityiq** directory and extract the war file.

   ```
   jar –xvf  identityiq.war
   ```

Configure Extended Searchable Attributes

1. For our implementation, we have documented requirements for two extended identity attributes that need to be searchable: *empId* and *status*.

   In a standard implementation, the implementation team is responsible for adding the extended attributes to the Hibernate XML file. In our training environment, the Hibernate XML file has already been configured with these two named extended attributes.

   In this exercise you will replace the default Hibernate XML file with the pre-configured Hibernate XML file, and you will review the configuration to ensure it meets requirements.

   a. Navigate to the directory **/home/spadmin/ImplementerTraining/config**

   b. Copy the file **IdentityExtended.hbm.xml** to **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/classes/sailpoint/object**

      Copy Options:

      Option 1:
      Use the file browser to copy and paste the file. When prompted, click **Replace**.

      Option 2:
      Use the Linux copy command. At the $ command prompt, specify the file name and directory listed above in the following command:

      ```
      cp <insert file name> <insert directory name>
      ```

   c. Navigate to the directory: **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/classes/sailpoint/object**

   d. In the **object** directory, open the **IdentityExtended.hbm.xml** file using any editor; gedit is provided in the VM and is a good editor for viewing and editing XML files.

      Note: You will not be making changes to the file.

      About gedit: If red highlights are displayed, there is a syntax error in the XML. This may be a sign that you've accidentally changed the file.

   e. Find the entries for the named extended attributes, *empId* and *status*. Complete the missing components of the definitions below.

      ```
      <property name="empId" type="_______" length="450"
            access="sailpoint.persistence.ExtendedPropertyAccessor"
            index="spt_identity_empId_ci"/>
      <property name="status" type="string" length="_______"
            access="sailpoint.persistence.ExtendedPropertyAccessor"
            index="spt_identity__________________"/>
      ```

f. This exercise confirms that the *empId* and *status* extended attributes will be created in the database, with indexes. Later in this training course, after IdentityIQ is installed, you will create these extended attributes within IdentityIQ. Until they are created within IdentityIQ, **you will see errors** that they are not defined in the Identity object configuration.

2. Project requirements also tell us that we will need additional searchable and indexed extended identity attributes, but we don't yet know which ones. For these, we will use the 10 default placeholder attributes, but we want all 10 to be indexed, rather than the default of 5 indexed.

   a. Confirm that the Hibernate XML file is configured to support 10 searchable *and indexed* placeholder attributes per identity. *Complete the missing components of the definitions below*:

```
<property name="extended1" type="string" length="450"
          index="spt_identity_extended1_ci"/>
<property name="extended2" type="string" length="450"
          index="spt_identity_extended2_ci"/>
<property name="extended3" type="string" length="450"
          index="spt_identity_extended3_ci"/>
<property name="extended4" type="string" length="450"
          index="spt_identity_extended4_ci"/>
<property name="extended5" type="string" length="450"
          index="spt_identity_extended5_ci"/>
<property name="extended6" type="string" length="450"

          ________________________________ />
<property name="extended7" type="string" length="450"
          index="spt_identity_          _ci"/>


                              ___________
<property name="extended8" type="string" length="450"
          index="spt_identity_          _ci"/>

                              ___________
<property name="extended9" type="string" length="450"
          index="spt_identity_          _ci"/>

                              ___________
<property name="extended10" type="string" length="450"
          index="spt_identity_          _ci"/>
                              ___________
```

   b. How many identity extended attribute placeholders are configured by default? (Hint: check the in-line documentation at the top of the file)

   ____

c. How many identity extended attributes may be configured in total?

____

d. How many extended attribute placeholders are configured *with* a database index by default?

____

e. How many extended attribute placeholders in this Hibernate file are configured to have a database index?

____

3. Close the file and do not save any changes.

Configure the Database

1. Configure permissions on the iiq command so that we may execute it.

   a. Using a Linux terminal window, navigate to the **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/bin** directory

   b. Run the following command to mark the iiq command as executable:

   ```
   chmod +x iiq
   ```

2. Generate IdentityIQ Schema files

   a. Run the following command from the Linux terminal to generate the database schema files:

   ```
   ./iiq schema
   ```

3. Load the MySQL Schema file into MySQL to create the IdentityIQ database

   a. Using the command prompt, navigate to the **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/database** directory and run the following commands to log in to MySQL:

   ```
   mysql –u root –p
   Enter password: root
   ```

   b. Within the MySQL command line utility, type the following in order to load the schema into MySQL:

   ```
   mysql> source create_identityiq_tables.mysql
   ```

c. When the command finishes running, type the following to confirm that the **identityiq** database has been created properly. The other databases are not important. Make sure that **identityiq** is in the list of databases.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| identityiq         |
| mysql              |
| performance_schema |
| trakk              |
| prism              |
+--------------------+
6 rows in set (0.00 sec)
```

d. Type **quit** to exit the MySQL command line utility.

4. Analyze the Database Settings that IdentityIQ will use to connect to the database. You will not edit this file for the training environment. The default values will work in our environment with no modifications.

    a. Navigate to and open the configuration file for the IdentityIQ database: **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/classes/ iiq.properties**

    b. The table below lists the most commonly edited values. View the iiq.properties file and fill in the blanks in the table for the three missing default values.

Database Type	Determined by which database section is uncommented. **Default**: mysql 5
dataSource.username	Username to use when connecting to the database **Default**: identityiq
dataSource.password	Encrypted password to use when connecting to the database. (default: identityiq) **Note:** generated using the **iiq encrypt <password>** command
dataSource.url	Defines the host name, port and database to connect to. (default: use the standard port on localhost, database name = identityiq)
dataSource.driverClassName	Defines the driver to use when connecting to the database **Default**: oracle.jdbc.driver.OracleDriver

Note: For best performance, it is ***VERY*** important to update the default JDBC driver supplied with IdentityIQ to the most current driver supplied by your database vendor.

Initialize IdentityIQ and Verify your Installation

1. Using the IdentityIQ Console, import the default IdentityIQ objects to initialize the system. The console starts an instance of IdentityIQ and may take a few moments to start. You will know that it is running when you see the > prompt.

   a. Using a Linux terminal, navigate to: **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/bin**

   b. Run the following command:

   ```
   ./iiq console
   ```

   Note: In the training environment, the console can also be run from the desktop shortcut.

   c. Upon console start-up, you will see two error messages. What is causing the error messages? **Hint**: See page 1-13, number 1f.

   attributes are not defined in the Identity object configuration.

   d. At the console command prompt, load the default IdentityIQ objects using the following command:

   ```
   > import init.xml
   ```

   e. When the import is complete, **quit** the console.

2. Start the Tomcat Application Server and wait 30 to 60 seconds while the application server starts.

   Options to start Tomcat:

   Option 1: From the desktop, run the "**Start Tomcat**" shortcut

   Option 2: Type the following command in a Linux terminal:

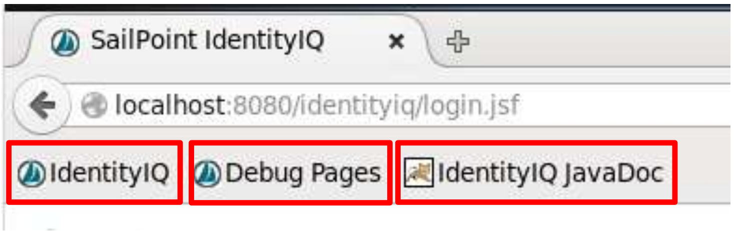
   ```
   StartTomcat
   ```

   To monitor the start process in the log file, use the desktop shortcut, **Tail Tomcat Standard Out**. The server has started when you see the phrase: **INFO: Server startup in *xxxxx* ms**.

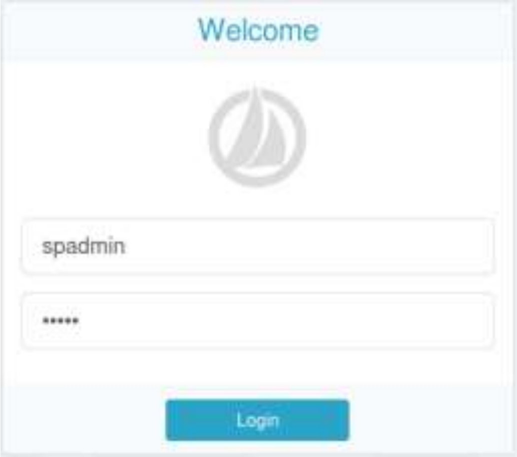
3. When Tomcat has started, log in to IdentityIQ using Firefox

   a. Click the Firefox book mark in the VM and go to: **http://localhost:8080/identityiq/**

Note: there are Bookmarks provided within the Firefox Browser for the IdentityIQ login page and others that we will use throughout this course.



b. Log in to IdentityIQ as **spadmin/admin**



c. If you can successfully log in and see the IdentityIQ application, then your installation was successful. If not, let your instructor know.

Exercise #2: Patching IdentityIQ

Objective

In this exercise, we will patch the product code to the latest patch level.

Overview

The patch process involves three major steps. Note that each patch install may not require all three steps. Always read the release notes for any patch in their entirety before patching a system.

- Deploy new product code (deploy the patch jar file install our install directory)
- Upgrade the database tables to support any changes required by the patch
    - New Tables
    - Deprecated Tables
- Run the patch script in order to convert any data as required by the new patch

Patch Installation

1. Stop the Tomcat Application Server.

   Options to stop Tomcat:

   Option 1: From the desktop, run the shortcut labeled "**Stop Tomcat**" or

   Option 2: Type the following command at a Linux terminal window: `StopTomcat`

2. Extract the IdentityIQ Patch file.

   a. Use the File Browser to locate the **identityiq-7.0p1.jar** (where '1' is the numeral one) file under **/home/spadmin/InstallImages** and copy it to the installation directory for IdentityIQ: **/home/spadmin/tomcat/webapps/identityiq**

   b. Using a Linux terminal window, navigate to the **/home/spadmin/tomcat/webapps/identityiq** directory and run the following command to extract the patch jar file.

   ```
   jar –xvf identityiq-7.0p1.jar
   ```

3. Patch the IdentityIQ Database

    a. Using the command prompt, navigate to the **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/database** directory and run the following commands to log in to MySQL:

    ```
    mysql –u root –p
    Enter password: root
    ```

    b. Within the MySQL command line utility, type the following in order to upgrade the IdentityIQ schema:

    ```
    mysql> source upgrade_identityiq_tables-7.0p1.mysql
    ```

    c. Type **quit** to exit the MySQL command line utility

4. Apply the Patch

    a. Using your Linux terminal, navigate to: **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/bin**

    b. Run the following command

    ```
    ./iiq patch 7.0p1
    ```

    c. Wait for the patch command to finish and watch for any errors. You should see two errors regarding the extended attributes *Employee ID* and *Status* ; we have not yet completed defining these attributes to IdentityIQ. If you also see a *Pool not open* error, ignore it

5. Confirm the installation

    d. Run the IdentityIQ Console either through the provided desktop shortcut or through a Linux terminal.

    e. Run the following command and confirm that the version and patch are 7.0p1.

    ```
    >about
    ```

    f. Quit the console.

4. Start the Tomcat Application Server and wait while the application server starts.

    Options to start Tomcat:

    Option 1: From the desktop, run the "**Start Tomcat**" shortcut

    Option 2: Type the following command in a Linux terminal: `StartTomcat`

Exercise #3: Configuring IdentityIQ

Objective

In this exercise, we will configure features of IdentityIQ that will assist us in our implementation efforts.

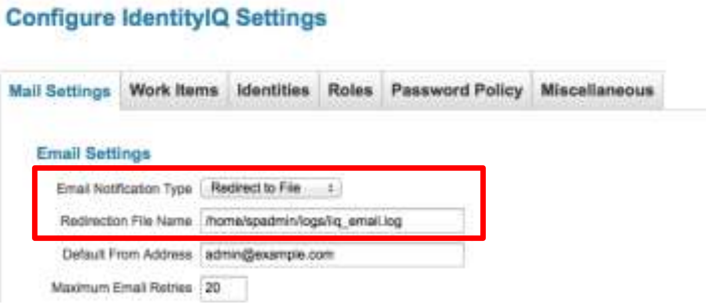
Overview

In order to support our client's needs, we will be turning on some troubleshooting/debugging/auditing features of IdentityIQ to support our development efforts:

- Configure the Email Redirector to send all system-generated emails to a local file instead of an SMTP Mail Server. This file is useful for debugging email notifications without sending real emails to users.
- Configure Auditing to log certain audit events into the Audit Table
- Configure Logging to send IdentityIQ log messages to a local file

Configure the Email Redirector

1. Double click on the Firefox icon to start IdentityIQ and log in with credentials **spadmin/admin.**
2. Within IdentityIQ, from the system setup gear, , select **Global Settings** and select **IdentityIQ Configuration**. Configure the following two options under **Email Settings**.
    a. Email Notification Type = **Redirect to File**
    b. Redirection File Name = **/home/spadmin/logs/iiq_email.log**



Email Notification Type: Redirect to File

Redirection File Name: /home/spadmin/logs/iiq_email.log

Default From Address: admin@example.com

Maximum Email Retries: 20

Note: This is the location in the UI where you can also configure the default Email Templates used for many notification types within the IdentityIQ application.

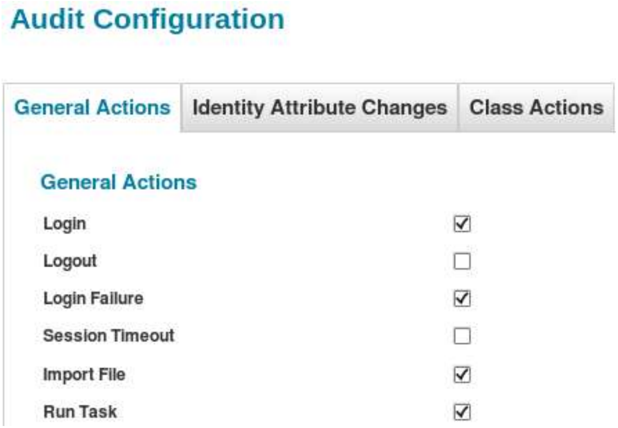
3. Scroll down to the bottom of the page and **Save**

Note: When you are ready to connect IdentityIQ to an SMTP mail server to send out real email notifications, make sure to change this configuration page to point to an SMTP mail server.

Configure IdentityIQ Auditing

1. Navigate to ⚙ → **Global Settings** → **Audit Configuration**

2. In the **General Actions** tab**,** configure the following four options by checking the box next to each:



Audit Configuration

General Actions

Login	☑
Logout	□
Login Failure	☑
Session Timeout	□
Import File	☑
Run Task	☑

3. Observe the other auditing options available.

   Note: You can turn on auditing for actions in the system, but can also turn on auditing for any changes to identity attributes or even the create/update/deletion of system objects. Also, it is possible to use the SailPoint API to audit additional items of your own choosing during rules or workflow steps.

4. Scroll to the bottom of the page and **Save**

5. Test the newly configured audit functions.

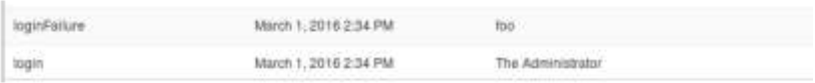
   a. Log out of IdentityIQ and attempt to log in using an incorrect username and password: example: **foo/foo**

   b. After this, log back in with the proper credentials: **spadmin/admin**

   c. Navigate to **Intelligence** → **Advanced Analytics** and select the **Audit Search** tab

   d. Don't adjust any parameters for the search; just choose **Run Search** from the bottom left.

   e. Confirm that you see entries showing the *login failure* and the successful *login*:



loginFailure	March 1, 2016 2:34 PM	foo
login	March 1, 2016 2:34 PM	The Administrator

6. While here, notice that Advanced Analytics provides detailed searching across IdentityIQ.

   a. How many different types of searches (tabs) are provided? 9 ______

b. Navigate to the **Identity Search** tab.

i. How many **Standard Attributes** are there for searching identities? 12

ii. How many Searchable Attributes are there? 3

iii. You increased the number of indexed searchable attributes defined in the IdentityIQ database. However, no searchable attributes have been defined to IdentityIQ. You will define attributes as searchable in upcoming exercises.

Configure IdentityIQ Logging

IdentityIQ uses log4J (a popular Java-based logging package), as its logging component. In this section, we will configure logging by configuring a log4j.properties file:

1. Copy the file: **log4j.properties** from **/home/spadmin/ImplementerTraining/config** and place it into **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/classes**

   Note: This will overwrite the default log4j.properties file with options added for the class environment.

2. This sample log configuration file will send all IdentityIQ logging output to the file: **/home/spadmin/logs/iiq_training_rolling.log**

3. Finish the configuration process by restarting the Tomcat application server to reload the log4j.properties file. Restart the application server from a terminal window (type the commands listed below) or use the shortcuts on the desktop.

   ```
   StopTomcat
   StartTomcat
   ```

   Note: You can view the success of both commands in the Tomcat Standard Out log.

4. Launch the desktop shortcut named: **Tail IdentityIQ Log** and leave this window running. This window will show any log messages generated by IdentityIQ as we work through the lab exercises.

5. You can edit the **log4j.properties** file to change the logging levels used by IdentityIQ. We will periodically adjust this file as we work through the exercises.

Exercise #4: Populating Identity Cubes – Loading Authoritative Data

Objective

The objective of this exercise is to become familiar with creating initial Identity Cubes from authoritative identity sources, creating new extended identity attributes, mapping identity attributes, and manipulating identity data as it is loaded into IdentityIQ.

Overview

Our client has authoritative identity data stored in two sources. One application, an HR application, stores employee data, and the other application stores contractor data. For our purposes, this data exists in two flat files in comma-separated-values format.

- AuthEmployees.csv
- AuthContractors.csv

Each file has the following data:

- employeeId (unique ID used for our Identity Attribute)
- firstName
- lastName
- managerId
- fullName (friendly name used for our Display Attribute)
- email
- department
- region
- location
- inactiveIdentity
- jobtitle (only present in the employee data)
- costcenter

They would like us to use these attributes from their Employee HR system and Contractor system to form new Identity Cubes within IdentityIQ.

When defining our applications, we will use the “employeeId” field as our Identity Attribute (unique value), and “fullName” as the Display Attribute (user friendly display name for the Identity.)

Additionally, the customer has a few more requirements:

- They want an additional Identity attribute called: **status**. This attribute will be set to “**Employee**” or “**Contractor**” depending on which authoritative source is used to create the identity. This attribute needs to be searchable and a group factory (so that we can easily create groupings of Identities that represent Employees and Contractors)
- For now, they want to set the default password for each identity to be “**xyzzy**” for testing purposes.
- They want the Cost Center attribute to be represented as a multi-value field. Currently, in the source data, Cost Center data is represented as a string such as: “R02, L04, L05, L06”. In order for individual Cost Centers to be searchable, we will need to mark this attribute as multi-valued so that the data is stored properly in IdentityIQ.

We will support these additional requirements by doing the following:

- Create an additional Identity Attribute called “Status” that will be sourced using rules that will determine if an Identity is an Employee or Contractor.
- Use a Creation Rule to set a default IdentityIQ password for each user as we create the Identity Cubes.

Define Employee and Contractor Applications

1. Create a new Application definition for the Employee Data

   a. Log in to IdentityIQ as **spadmin/admin**

   b. Navigate to **Applications → Application Definition** and select **Add New Application**

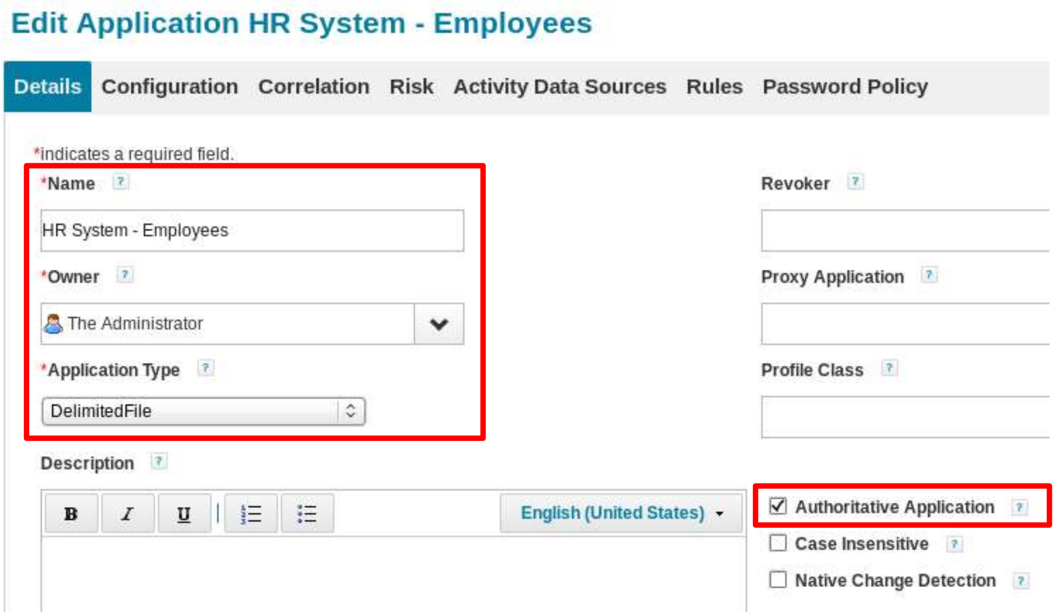
   c. Configure the Application as follows:

      i. Name: **HR System - Employees**

      ii. Owner: **spadmin (The Administrator)**

      iii. Application Type: **Delimited File**

      iv. Authoritative Application: **Checked**



Edit Application HR System - Employees

*indicates a required field.

*Name: HR System - Employees

*Owner: The Administrator

*Application Type: DelimitedFile

Revoker

Proxy Application

Profile Class

Description

☑ Authoritative Application

□ Case Insensitive

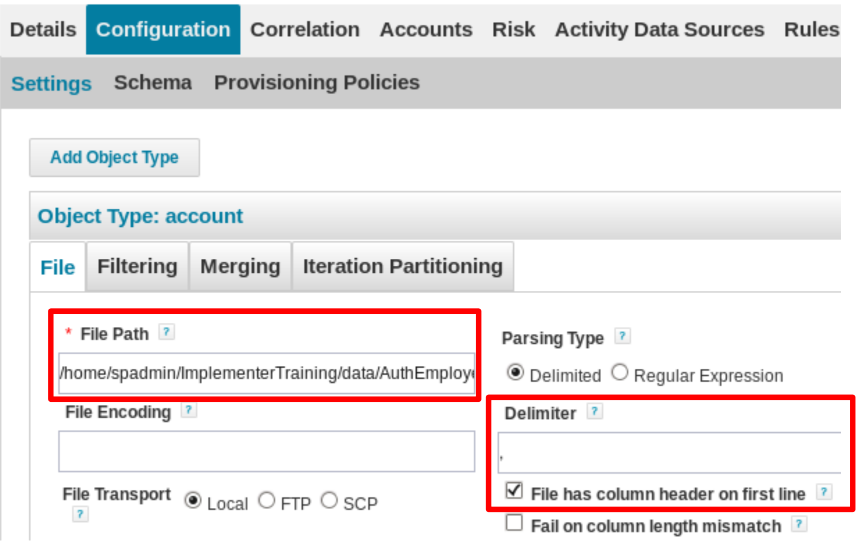
□ Native Change Detection

   d. Navigate to **Configuration → Settings** and configure as follows:

      i. File Path: **/home/spadmin/ImplementerTraining/data/AuthEmployees.csv**

      ii. Delimiter: **,**

      iii. File has column header on first line: **Checked**



Object Type: account

* File Path: /home/spadmin/ImplementerTraining/data/AuthEmploy

File Encoding:

File Transport: ● Local ○ FTP ○ SCP

Parsing Type: ● Delimited ○ Regular Expression

Delimiter: ,

☑ File has column header on first line

□ Fail on column length mismatch

2. At this point, we have defined the application, chosen the connector, and defined the connector attributes that define how to gather information about our Employees. We now need to tell the system what attributes we want to read from the file.

   a. Navigate to **Configuration → Schema**

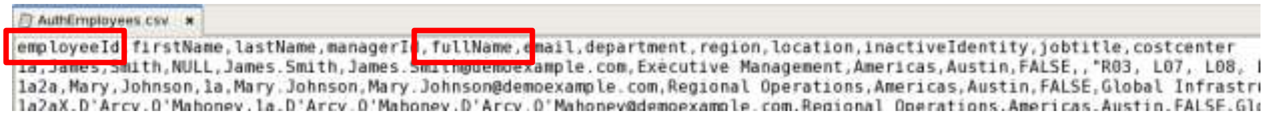
   b. Define the account schema as follows:

      i. Native Object Type: **account**

      ii. Identity Attribute: **employeeId**

      iii. Display Attribute: **fullName**

   c. These fields define which attributes that we are reading will be used to define uniqueness (“Identity Attribute”) and a friendly display name (“Display Attribute”). These attributes must match exactly (including case) with the actual schema attribute names from the file we are reading in.

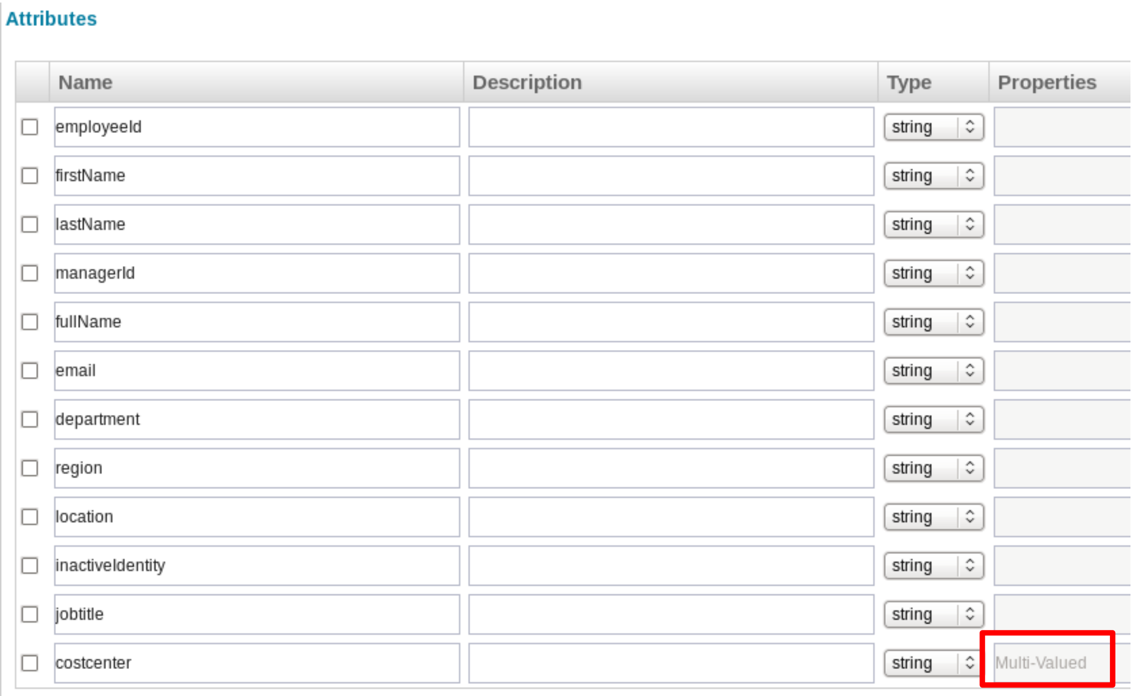


```
employeeId,firstName,lastName,managerId,fullName,email,department,region,location,inactiveIdentity,jobtitle,costcenter
1a,James,Smith,NULL,James.Smith,James.Smith@demoexample.com,Executive Management,Americas,Austin,FALSE,,"R03, L07, L08, 
1a2a,Mary,Johnson,1a,Mary.Johnson,Mary.Johnson@demoexample.com,Regional Operations,Americas,Austin,FALSE,Global Infrastr
1a2aX,D'Arcy,O'Mahoney,1a,D'Arcy.O'Mahoney,D'Arcy.O'Mahoney@demoexample.com,Regional Operations,Americas,Austin,FALSE,Gl
```

   d. Click the **Discover Schema Attributes** button. This will cause the connector to read just the header fields defining what data is present in the file. The schema attributes discovered should match what is in the actual raw file:

      `employeeId,firstName,lastName,managerId,fullName,email,department,region,location,inactiveIdentity,jobtitle,costcenter`

   e. For the **costcenter** attribute, select Edit (if necessary, scroll to the far right), and mark it as **Multi-valued.** Click **Save.**



Attributes

	Name	Description	Type	Properties
□	employeeId		string	
□	firstName		string	
□	lastName		string	
□	managerId		string	
□	fullName		string	
□	email		string	
□	department		string	
□	region		string	
□	location		string	
□	inactiveIdentity		string	
□	jobtitle		string	
□	costcenter		string	Multi-Valued

   f. Click the **Preview** button

   Preview iterates over the first 10 accounts and displays the results in a popup instead of loading it into IdentityIQ. This command is extremely useful when doing initial work onboarding applications to make sure that you are properly reading and manipulating the data and that your schema is correct. We will use this command extensively over the duration of this exercise.

3. We will now create a rule that will set the default password for each new Identity as we create them.

   a. At the top of the page, select **Rules**

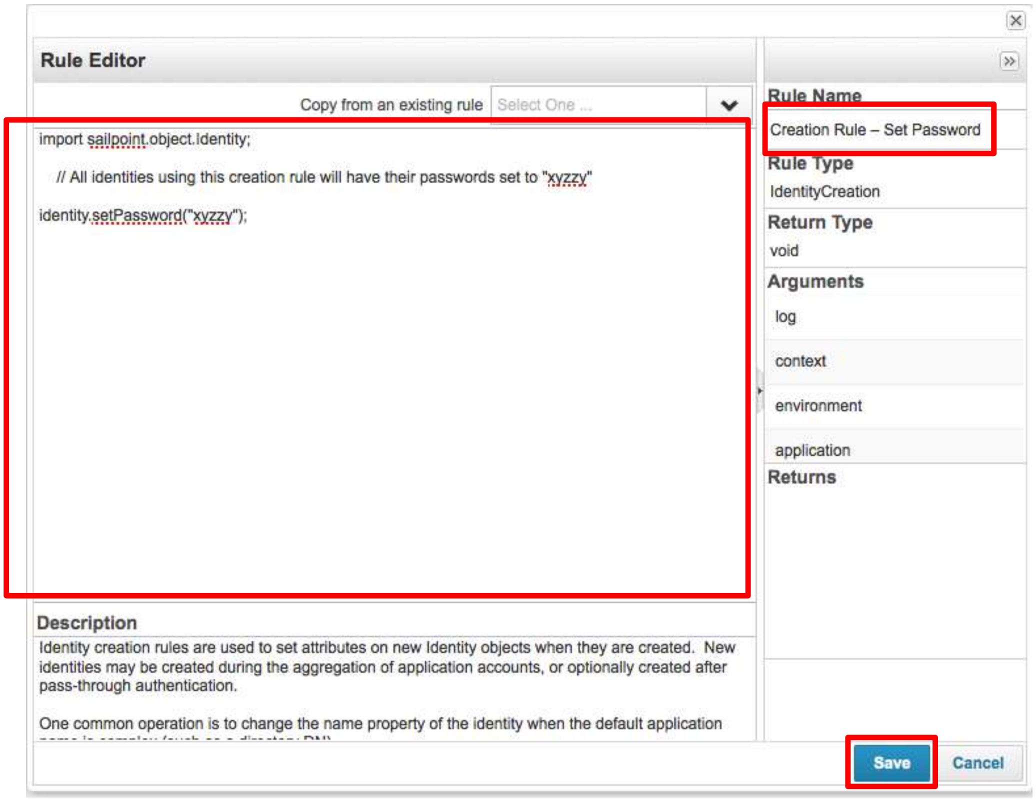
   b. To the right of the **Creation Rule**, select the button with the ellipsis (…)

   c. You will now see the **Rule Editor**

   d. Copy the beanshell code from the file **CreationRule-Set Default Passwords.txt** located in the **/home/spadmin/ImplementerTraining/beanshell** directory

   e. Paste the text into the **Rule Editor**

   f. Set the **Rule Name** to **Creation Rule - Set Password** as shown and then select **Save**



```
import sailpoint.object.Identity;

  // All identities using this creation rule will have their passwords set to "xyzzy"

identity.setPassword("xyzzy");
```

Rule Name: Creation Rule – Set Password

Rule Type: IdentityCreation

Return Type: void

Arguments: log, context, environment, application

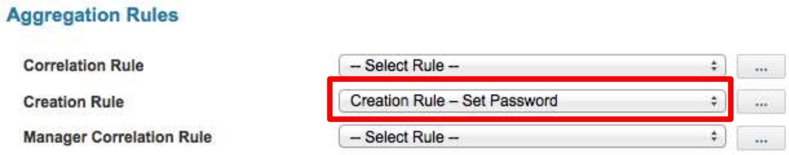
Returns

Description

Identity creation rules are used to set attributes on new Identity objects when they are created. New identities may be created during the aggregation of application accounts, or optionally created after pass-through authentication.

One common operation is to change the name property of the identity when the default application name is complex (such as a directory DN).

g. Select the rule you just created in the drop down



Correlation Rule: -- Select Rule --

Creation Rule: Creation Rule – Set Password

Manager Correlation Rule: -- Select Rule --

4. Scroll down and select **Save** to save the application.

5. Create a new Application definition for the Contractor Data

   a. Navigate to **Applications → Application Definition** and select **Add New Application**

   b. Configure the Application as follows:

      i. Name: **Contractor Feed**

      ii. Owner: **spadmin**

      iii. Application Type: **Delimited File**

      iv. Authoritative Application: **Checked**

c. Navigate to **Configuration → Settings** and configure as follows:

   i. File Path: **/home/spadmin/ImplementerTraining/data/AuthContractors.csv**

   ii. Delimiter: **,**

   iii. File has column header on first line: **Checked**

At this point, we have defined the application and where it will go to gather information about our Contractors. We now need to tell the system what attributes we want to read from the file.

d. Navigate to **Configuration → Schema**

e. Define the account schema as follows:

   i. Native Object Type: **account**

   ii. Identity Attribute: **employeeId**

   iii. Display Attribute: **fullName**

f. Click the **Discover Schema Attributes** button. The schema attributes discovered should match what is in the actual raw file:

   ```
   employeeId,firstName,lastName,managerId,fullName,email,depa
   rtment,region,location,inactiveIdentity,costcenter
   ```

g. For the **costcenter** attribute, mark it as **Multi-valued**

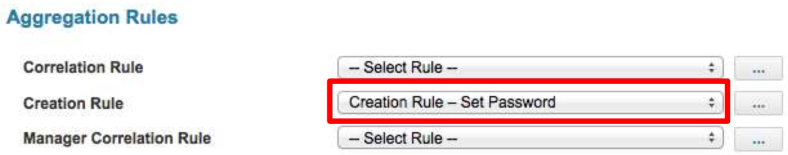
6. Click the **Preview** button.

   Remember that **Preview** does not write any information to the IdentityIQ database, but is very useful to ensure that you are properly reading and manipulating the data and that your schema is correct.

7. We will now use the same Creation Rule we defined before for setting the default password for each identity.

   a. At the top, select **Rules**

b. Select the rule we created earlier, **Creation Rule - Set Password**



Aggregation Rules

Correlation Rule	-- Select Rule --
Creation Rule	Creation Rule – Set Password
Manager Correlation Rule	-- Select Rule --

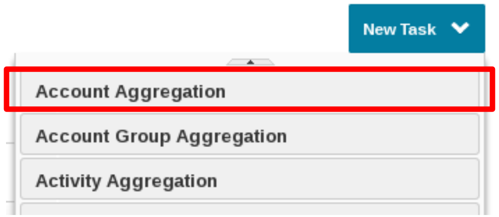
Note: We are using the same rule as we did for the previous application. It is common in IdentityIQ implementations to re-use rules. In this case, the rule applies to loading both Employees and Contractors.

8. Scroll down and select **Save** to save the application.

Aggregate the Employee and Contractor Data

1. Now, we will aggregate (or load) the Employee and Contractor data from the two delimited files by creating an Account Aggregation task. **Note:** We will use the same task for loading both files. *This process of loading account data from Authoritative Applications will generate our initial Identity Cubes.*

   a. Navigate to **Setup → Tasks** and under **New Task** choose **Account Aggregation**



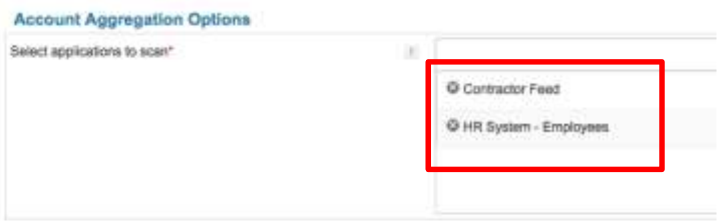
   b. Define the Task as follows:

      i. Name: **Aggregate Employees and Contractors**

      ii. Description: **Aggregate Employees from HR Data and Contractors from Contractor Feed.**

      iii. Select applications to scan:

      HR System – Employees
      Contractor Feed



Account Aggregation Options

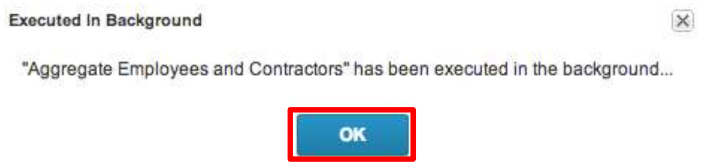
Select applications to scan*

Contractor Feed
HR System - Employees

iv. Select the **Detect Deleted Accounts** checkbox

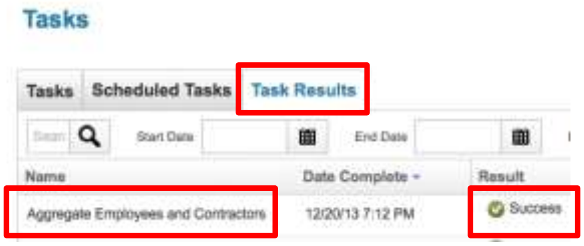
v. Select the **Disable optimization of unchanged accounts** checkbox

c. Scroll down and choose **Save and Execute** and choose **OK** when prompted.



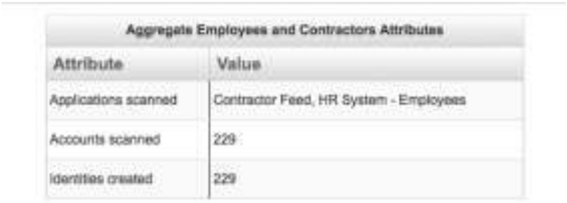
"Aggregate Employees and Contractors" has been executed in the background...

d. Once you are back on the main **Tasks** page, select **Task Results** as shown below. Click the **Aggregate Employees and Contractors** entry to open the task result. While it is pending, you can open it and view information about the processing. When it is complete, you will see the results of the aggregation.



Name	Date Complete	Result
Aggregate Employees and Contractors	12/20/13 7:12 PM	Success

e. The task output should show that two total applications were scanned, that a number of accounts were scanned, and that identities were created for each account.



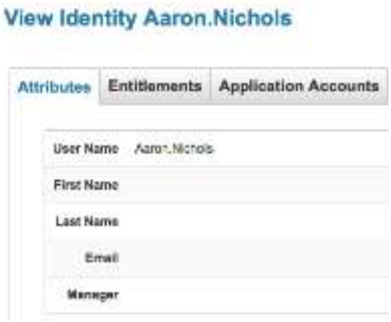
Aggregate Employees and Contractors Attributes

Attribute	Value
Applications scanned	Contractor Feed, HR System - Employees
Accounts scanned	229
Identities created	229

2. Confirm that the aggregation was successful.

a. View an Identity; navigate to **Identities → Identity Warehouse**

i. Click any user and confirm that an Identity Cube was created for this user. Note that all of the identity attributes are blank. This is because we haven't defined a mapping between the identity attributes and the applications that are feeding data into IdentityIQ



View Identity Aaron.Nichols

User Name	Aaron.Nichols
First Name	
Last Name	
Email	
Manager	

b. Confirm that the Creation Rule was successful

i. **Log out** of IdentityIQ

ii. Log in as the employee: **Aaron.Nichols/xyzzy**

iii. **Log out** of IdentityIQ

iv. Log in as the contractor: **Allen.Burton/xyzzy**

v. If you cannot log in to both accounts using the names and passwords, then you may have an issue with your Creation Rule. Double check that both applications have the Creation Rule defined properly.

vi. **Log out** of IdentityIQ

Understanding What We Just Did

There is an exact correspondence between the application schema and the Application Account data stored in IdentityIQ. *During aggregation, IdentityIQ gathers exactly what is specified in the application schema from the source application or calculated fields and stores it as Application Account data*, viewable on the Accounts Tab in the Application definition or on the Application Accounts tab on Identity Cubes.

1. Log in to IdentityIQ as **spadmin/admin**

2. Compare the ***application account*** for **HR System - Employees** with the ***schema attributes*** for the **HR System - Employees** application

a. Navigate to **Identities → Identity Warehouse** and view **Aaron.Nichols**

i. Which attribute is populated? user name

ii. What is the name of the field you set for IdentityIQ to populate this attribute? Hint: Look at the application schema.

____

b. On Aaron’s cube, select the **Application Account**s tab and view his account details for the **HR System - Employees** application

   i. How many attributes are listed on the application account? ____12____________

   ii. How many cost centers are associated with the account? _____4_______________
   Note: If your configuration is correct, there should be one cost center per line.

   iii. What was specified on the schema to include the cost centers as unique items?

   ____

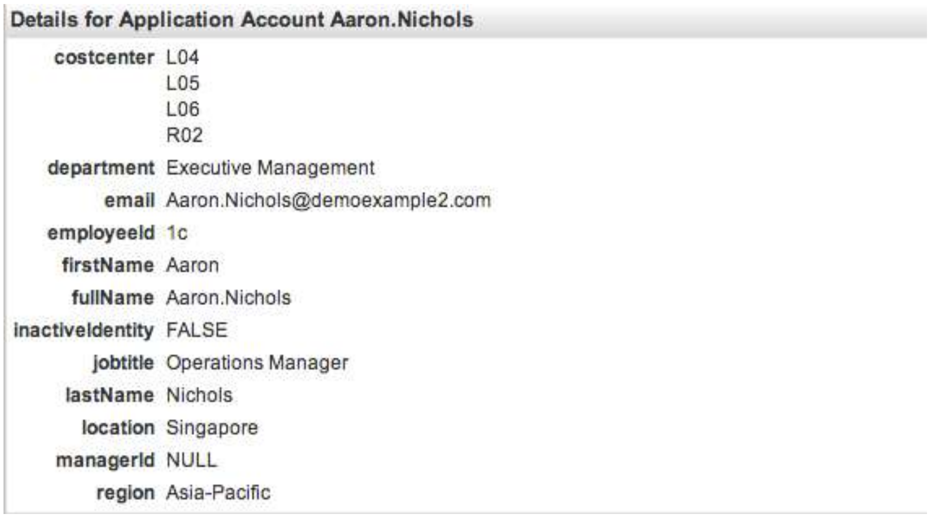
c. Navigate to **Applications → Application Definition**, select the **HR System - Employees** application and view the schema

   i. How many items are listed on the schema? ______________________________

   ii. Compare the schema attributes to the application account items and note that they are the same

   iii. Check your answer against those in 2b (the previous question). To simplify the comparison, the application account and the schema are included on the following page.

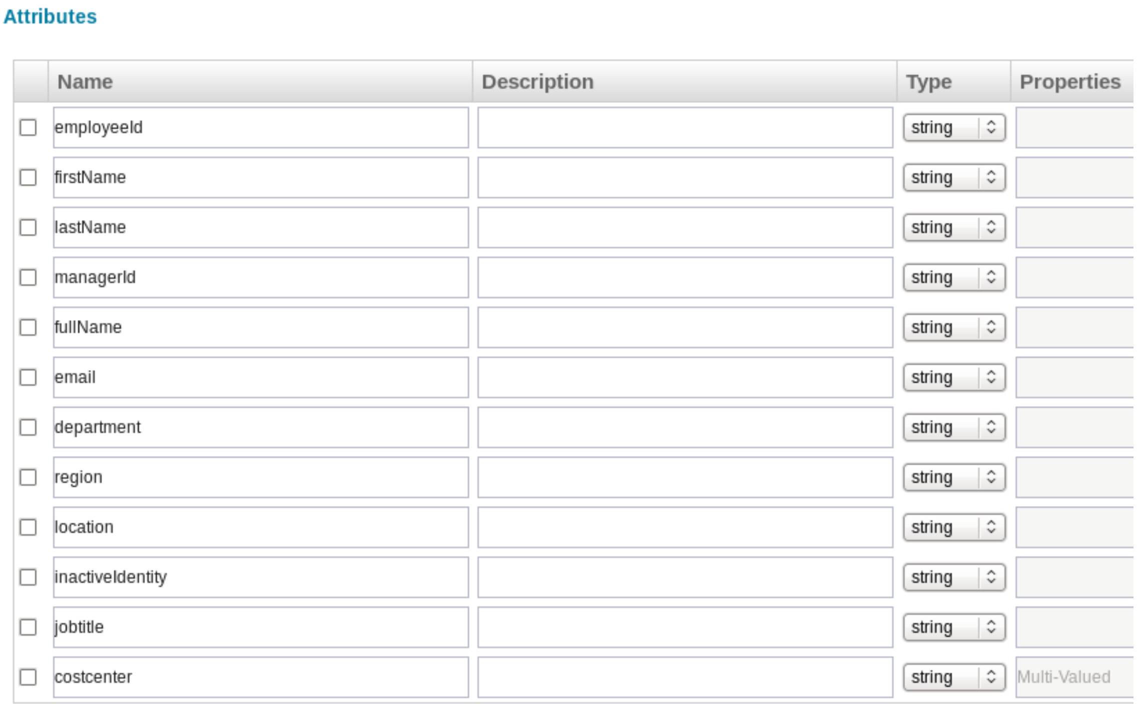
HR System - Employees *Application Account*:



Details for Application Account Aaron.Nichols

costcenter	L04 L05 L06 R02
department	Executive Management
email	Aaron.Nichols@demoexample2.com
employeeId	1c
firstName	Aaron
fullName	Aaron.Nichols
inactiveIdentity	FALSE
jobtitle	Operations Manager
lastName	Nichols
location	Singapore
managerId	NULL
region	Asia-Pacific

HR System - Employees *Application schema*:



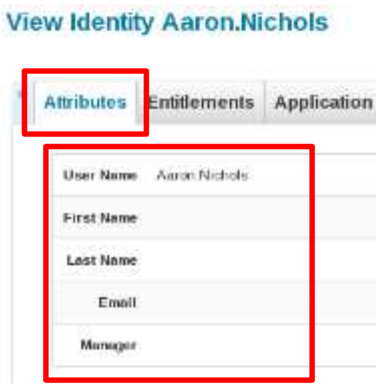
Attributes

	Name	Description	Type	Properties
☐	employeeId		string	
☐	firstName		string	
☐	lastName		string	
☐	managerId		string	
☐	fullName		string	
☐	email		string	
☐	department		string	
☐	region		string	
☐	location		string	
☐	inactiveIdentity		string	
☐	jobtitle		string	
☐	costcenter		string	Multi-Valued

Configure Identity Mappings for Standard Attributes

We now have two properly configured application definitions that load in account data from our Enterprise Directory. We now must define what data from these authoritative sources we will use to populate our identity data.

Previously we saw that so far only one Identity Attribute has been populated – User Name. User Name is populated by default from the Display Attribute in the schema header.



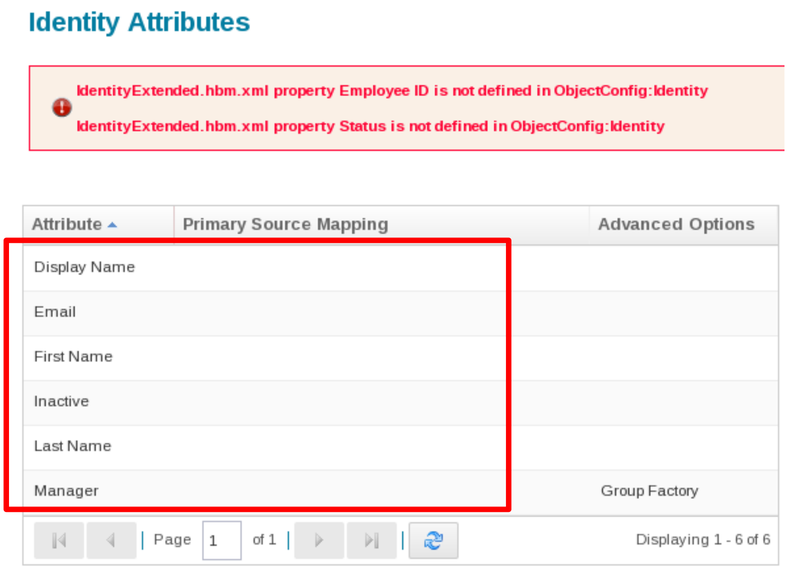
User Name Aaron Nichols
First Name
Last Name
Email
Manager

With the exception of User Name, Identity Attributes are only populated when they have an associated mapping. Typically Identity Attributes are populated from the application account attribute read in from a directory or HR application (an authoritative source), but they can also be populated by a rule. The source for an identity attribute is defined through Identity Mappings.

1. Navigate to ⚙ → **Global Settings** → **Identity Mappings**.
   This is the main interface for configuring Identity Attributes and how they are populated.

   a. Earlier you created two extended attributes in the IdentityIQ database. Notice the reminder to define those attributes in IdentityIQ. We will define them later in this exercise.

   b. Notice that the standard attributes already exist -- you do not create them. Also notice that they are created with no source mapping -- you will define the source from which they are populated.



Identity Attributes

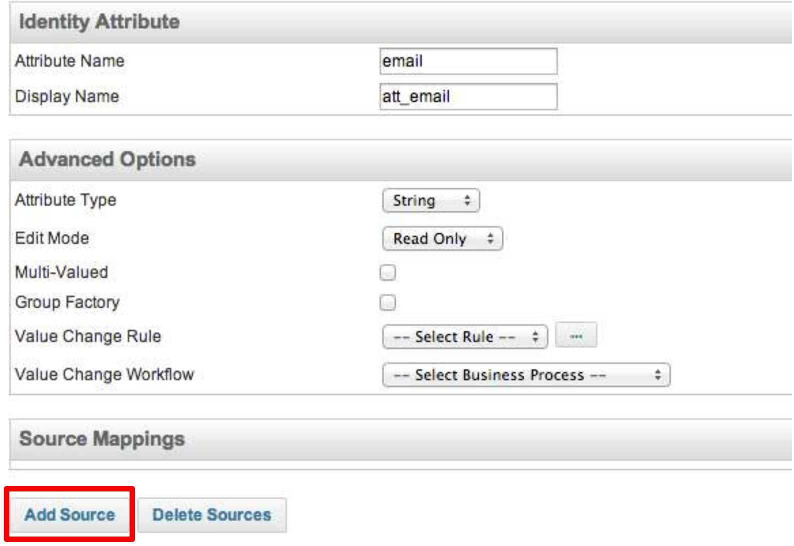
IdentityExtended.hbm.xml property Employee ID is not defined in ObjectConfig:Identity

IdentityExtended.hbm.xml property Status is not defined in ObjectConfig:Identity

Attribute ▲	Primary Source Mapping	Advanced Options
Display Name		
Email		
First Name		
Inactive		
Last Name		
Manager		Group Factory

2. Choose **Email** from the list of identity attributes.

   a. Scroll down to **Source Mappings** and click **Add Source** to configure the source of this Identity Attribute



Identity Attribute

Attribute Name	email
Display Name	att_email

Advanced Options

Attribute Type	String
Edit Mode	Read Only
Multi-Valued	□
Group Factory	□
Value Change Rule	-- Select Rule --
Value Change Workflow	-- Select Business Process --

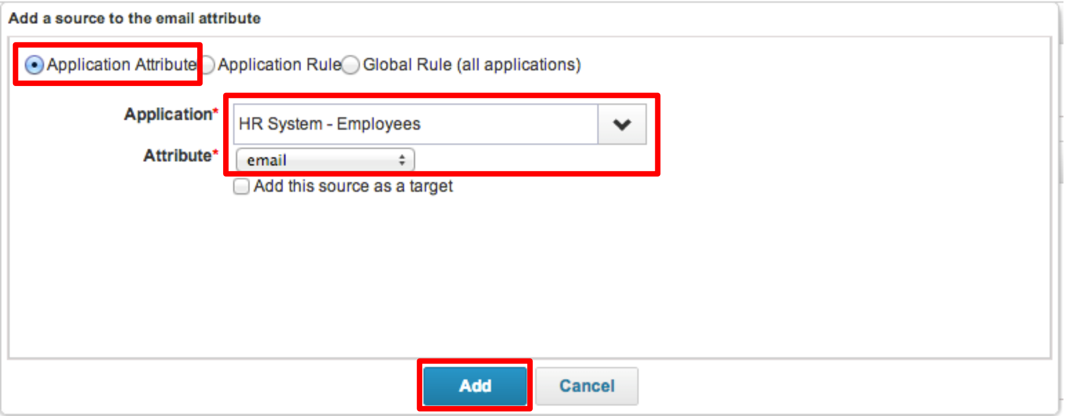
Source Mappings

   b. Choose **Application Attribute**

   c. Application: **HR System - Employees**

   d. Attribute: **email**

e. Click **Add** to add the source mapping



Add a source to the email attribute

● Application Attribute ○ Application Rule ○ Global Rule (all applications)

Application*: HR System - Employees

Attribute*: email

□ Add this source as a target

Note: Attributes can be populated by Application Attributes, or by a Rule. They can also be populated by multiple sources, as we are about to see in the next few steps.

f. Click **Add Source** again to configure where this attribute will come from for a contractor identity.

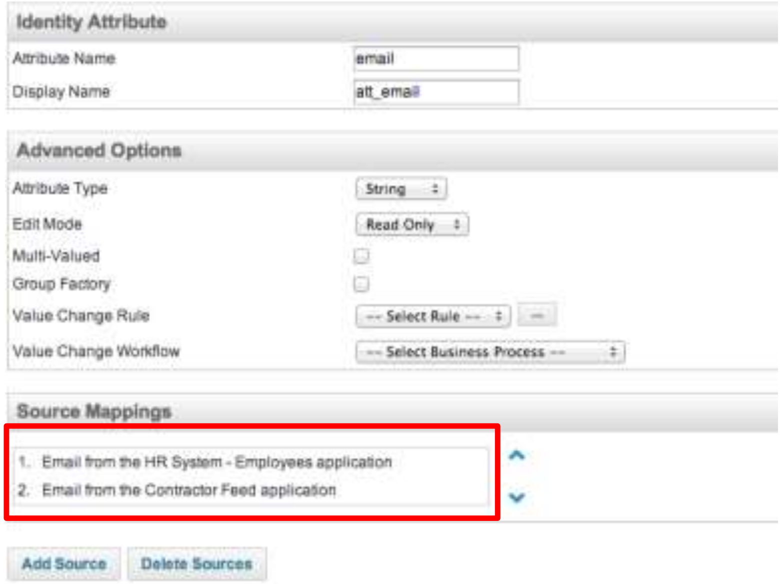
g. Choose **Application Attribute**

h. Application: **Contractor Feed**

i. Attribute: **email**

j. Click **Add** to add the source mapping

k. Your attribute mapping should look as shown, with two mappings for where the email Identity Attribute is sourced. For Employees, it will be sourced from the Employee data, and for Contractors, it will be sourced from the Contractor data.



Identity Attribute

Attribute Name: email

Display Name: att_email

Advanced Options

Attribute Type: String

Edit Mode: Read Only

Multi-Valued: □

Group Factory: □

Value Change Rule: -- Select Rule --

Value Change Workflow: -- Select Business Process --

Source Mappings

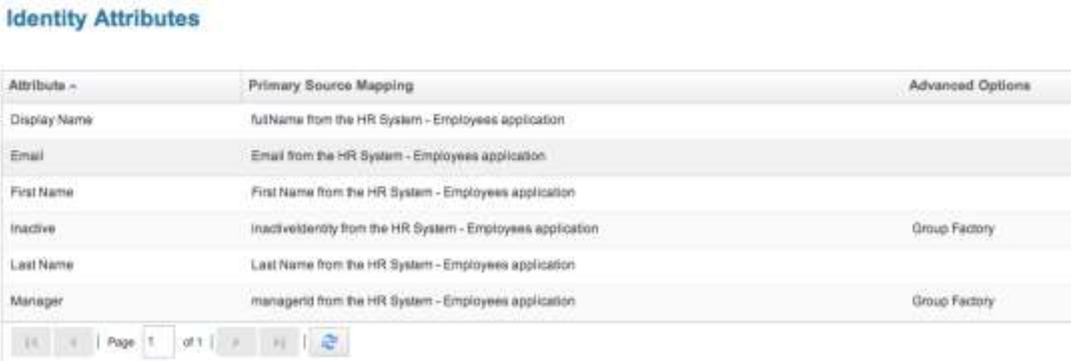
1. Email from the HR System - Employees application
2. Email from the Contractor Feed application

l. Click **Save** to complete the changes to the **email** attribute

m. Repeat the process for the following attributes as shown in the table. **Note:** make sure that you set the Advanced Options from the last column when defining the Identity attributes

Standard Attribute	Source Mappings	Advanced Options
Display Name	fullName from HR System - Employees fullName from Contractor Feed	
First Name	firstName from HR System - Employees firstName from Contractor Feed	
Inactive	inactiveIdentity from HR System - Employees inactiveIdentity from Contractor Feed	Group Factory checked
Last Name	lastName from HR System - Employees lastName from Contractor Feed	
Manager	managerId from HR System - Employees managerId from Contractor Feed	Group Factory checked

n. After editing the standard identity attributes, it should look like this:



Identity Attributes

Attribute	Primary Source Mapping	Advanced Options
Display Name	fullName from the HR System - Employees application	
Email	Email from the HR System - Employees application	
First Name	First Name from the HR System - Employees application	
Inactive	inactiveIdentity from the HR System - Employees application	Group Factory
Last Name	Last Name from the HR System - Employees application	
Manager	managerId from the HR System - Employees application	Group Factory

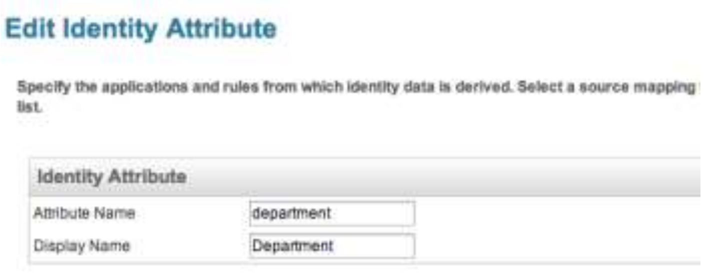
Define Extended (Custom) Identity Attributes

We will now define and configure additional Identity Attributes. These are attributes specific to the implementation that are additional to the standard attributes. These attributes are called Extended Identity Attributes.

1. Click the **Add New Attribute** button on the **Identity Attributes** page and enter the following:

   a. Attribute Name: **department**

   b. Display Name: **Department**



Edit Identity Attribute

Specify the applications and rules from which identity data is derived. Select a source mapping list.

Identity Attribute	
Attribute Name	department
Display Name	Department

c. Under **Source Mappings**, select **Add Source**

   i. Choose **Application Attribute**

   ii. Application: **HR System - Employees**

   iii. Attribute: **department**

   iv. Click **Add**

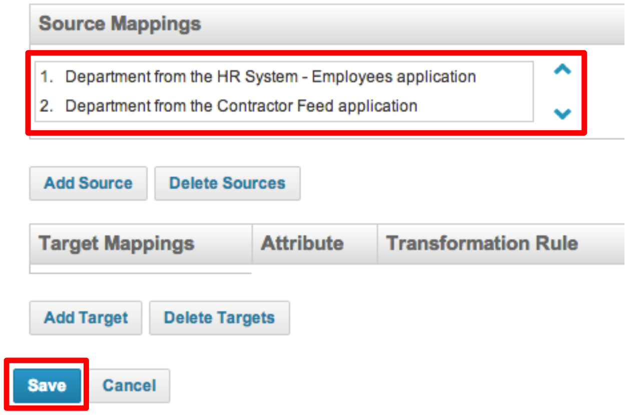
d. Click **Add Source**

   i. Choose **Application Attribute**

   ii. Application: **Contractor Feed**

   iii. Attribute: **department**

   iv. Click **Add**



Source Mappings

1. Department from the HR System - Employees application
2. Department from the Contractor Feed application

Target Mappings	Attribute	Transformation Rule

e. Click **Save** to save all changes to the **department** attribute

2. Repeat the steps above for the following additional Identity Attributes that we will add:

Attribute Name	Display Name	Source Mappings	Advanced Options
location	Location	location from HR System - Employees location from Contractor Feed	Group Factory, Searchable
empId	Employee ID	employeeId from HR System - Employees employeeId from Contractor Feed	Searchable
region	Region	region from HR System - Employees region from Contractor Feed	Group Factory, Searchable
jobtitle	Job Title	jobtitle from HR System - Employees	
costcenter	Cost Center	costcenter from HR System - Employees costcenter from Contractor Feed	Group Factory, Multi-valued (see note)

Note: When you select Group Factory, IdentityIQ automatically selects Searchable. Multi-valued attributes are automatically searchable. So, when you select both Group Factory and Multi-valued, when you save, IdentityIQ unchecks the searchable box for you.

3. Next, we will add an identity attribute that will be derived with a rule. The rule will set the identity's status attribute to either "Employee" or "Contractor", based on which source application file the identity comes from.

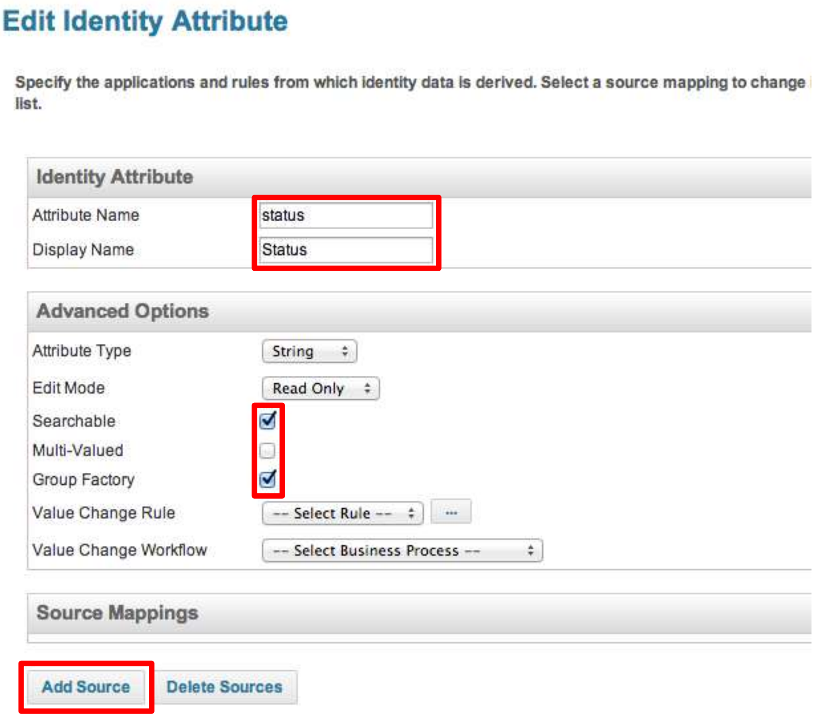
   a. Select **Add New Attribute** and configure the attribute as defined:

      i. Attribute Name: **status**

      ii. Display Name: **Status**

      iii. Searchable: **checked**

      iv. Group Factory: **checked**



Edit Identity Attribute

Specify the applications and rules from which identity data is derived. Select a source mapping to change list.

Identity Attribute

Attribute Name	status
Display Name	Status

Advanced Options

Attribute Type	String
Edit Mode	Read Only
Searchable	☑
Multi-Valued	□
Group Factory	☑
Value Change Rule	-- Select Rule -- ...
Value Change Workflow	-- Select Business Process --

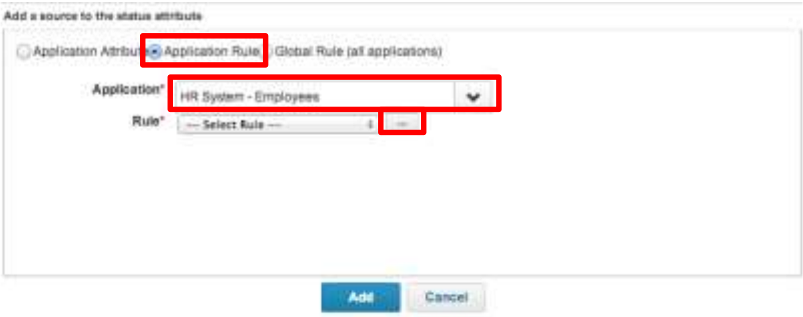
Source Mappings

v. Under **Source Mappings**, select **Add Source**

1. Configure the Source Mapping as shown

   a. Application Rule: **Selected**

   b. Application: **HR System – Employees**



Add a source to the status attribute

○ Application Attribute ● Application Rule ○ Global Rule (all applications)

Application* HR System - Employees

Rule* --- Select Rule --- ...

   c. Click the "…" to access the Rule Editor. Edit the Rule as shown here:

      i. Rule Name: **Identity Attribute: Employees**

      ii. In the Rule Editor, type the Java BeanShell script: **return "Employee";**

d. Click **Save** to save the rule

2. Choose the rule you just created and select **Add.**

vi. Under **Source Mappings**, select **Add Source** again

1. Configure the Source Mapping as shown

a. Application Rule: **Selected**

b. Application: **Contractor Feed**

c. Click the "…" to edit the Rule as shown here:

i. Rule Name: **Identity Attribute: Contractors**

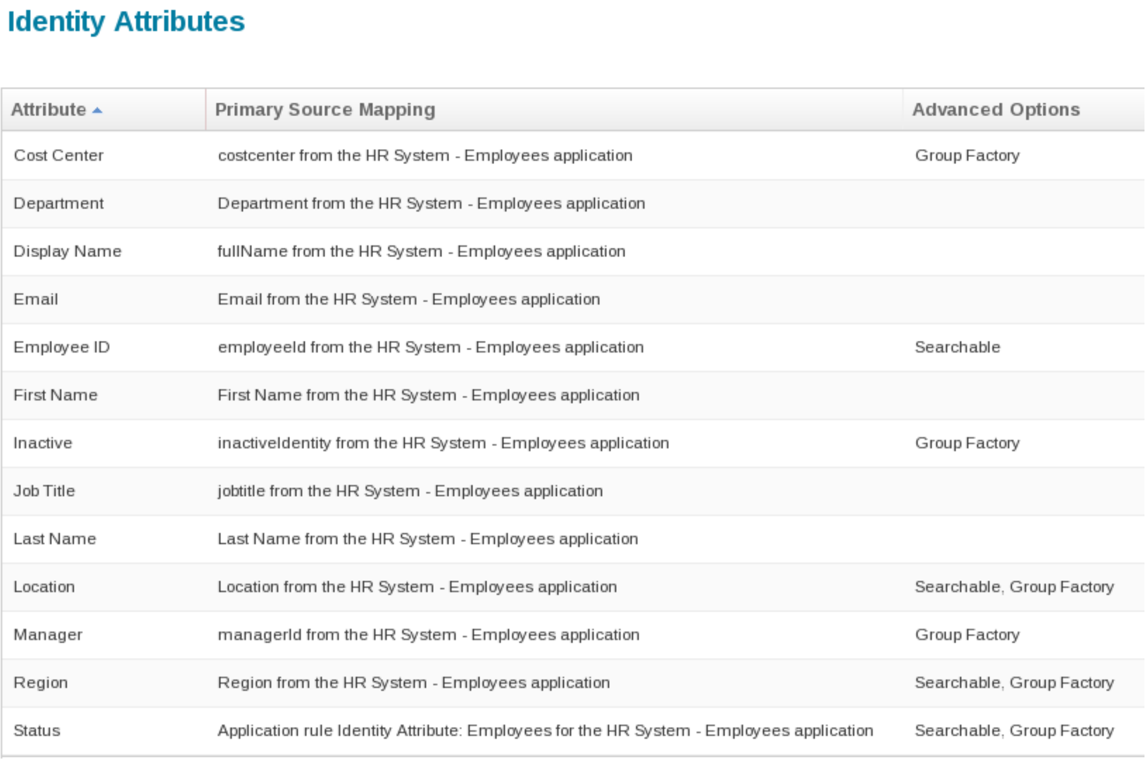
ii. Rule Script: **return "Contractor";**

d. Click **Save** to save the rule

2. Choose the rule you just created and select **Add.**

vii. Click **Save** to save all changes to the **status** attribute

4. After adding these 7 additional Identity Attributes, your Identity Attributes screen should look similar to the following:



Identity Attributes

Attribute ▲	Primary Source Mapping	Advanced Options
Cost Center	costcenter from the HR System - Employees application	Group Factory
Department	Department from the HR System - Employees application	
Display Name	fullName from the HR System - Employees application	
Email	Email from the HR System - Employees application	
Employee ID	employeeId from the HR System - Employees application	Searchable
First Name	First Name from the HR System - Employees application	
Inactive	inactiveIdentity from the HR System - Employees application	Group Factory
Job Title	jobtitle from the HR System - Employees application	
Last Name	Last Name from the HR System - Employees application	
Location	Location from the HR System - Employees application	Searchable, Group Factory
Manager	managerId from the HR System - Employees application	Group Factory
Region	Region from the HR System - Employees application	Searchable, Group Factory
Status	Application rule Identity Attribute: Employees for the HR System - Employees application	Searchable, Group Factory

Note: Certain fields are marked as "Searchable" and/or "Group Factory". A field should be marked as searchable if you will need to use it for account correlation (like Employee ID) or for Analytics (Location, Region). Group Factory identifies those fields from which groups of users may be created (for example, a group of inactive users). You will use these later.

Account Correlation: The act of matching accounts to an Identity Cube.

Update Manager Status

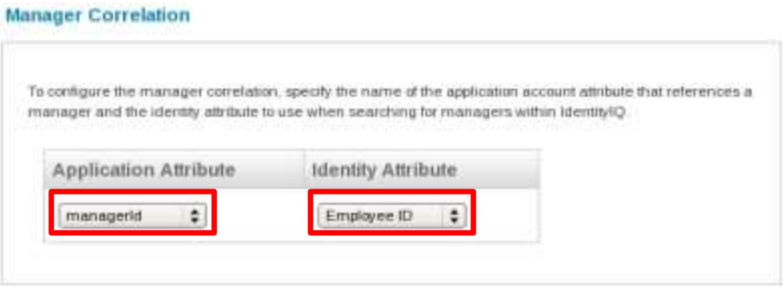
The data we are reading from the delimited files includes manager data. In order to get this manager data to be properly handled by IdentityIQ, we need to define a manager correlation. This correlation will describe which Application Attribute defines a user's manager, and which attribute to map this to in the Identity.

In our case, each managerId from the delimited files maps to a specific Employee ID.

1. Define Manager Correlation for the **HR System - Employees** application.

    a. Navigate to **Applications → Application Definition → HR System - Employees →Correlation**

b. Under **Manager Correlation**, set the **Application Attribute** to **“managerId”** and the **Identity Attribute** to **“Employee ID”**



Manager Correlation

To configure the manager correlation, specify the name of the application account attribute that references a manager and the identity attribute to use when searching for managers within IdentityIQ.

Application Attribute	Identity Attribute
managerId	Employee ID

c. Select **Save** to save the application definition.

2. Repeat the above steps for the **Contractor Feed** application.

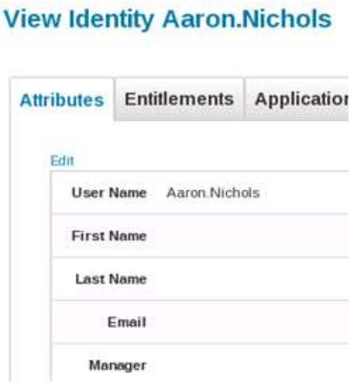
    a. Navigate to **Applications → Application Definition → Contractor Feed → Correlation**

    b. Under **Manager Correlation**, set the **Application Attribute** to **“managerId”** and the **Identity Attribute** to **“Employee ID”**

    c. Select **Save** to save your application definition

Configure the UI to Display new Identity Attributes

Another thing we need to do is configure the UI to display all these new identity attributes. By default, IdentityIQ will only display a certain set of attributes as shown here.



View Identity Aaron.Nichols

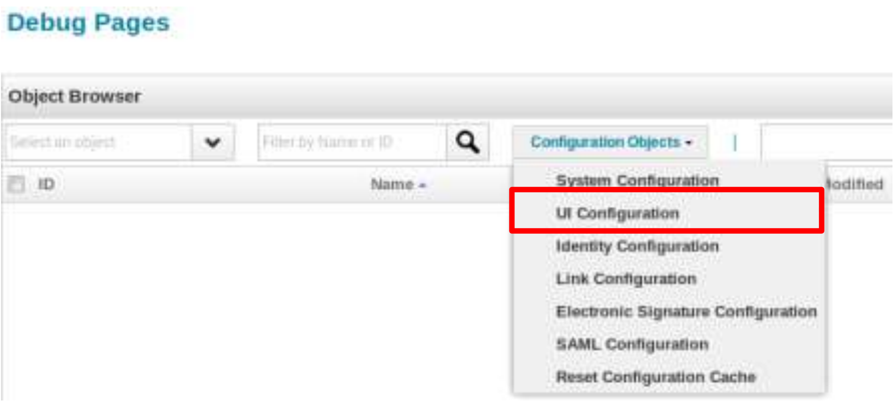
User Name	Aaron.Nichols
First Name	
Last Name	
Email	
Manager	

We want to extend the interface to show our new identity attributes: Status, Job Title, etc.

1. Configure IdentityIQ to display all Identity Attributes

    a. Using Firefox, open another tab and browse to **http://localhost:8080/identityiq/debug** or use the **Debug Pages** shortcut in you Firefox browser.

b. The IdentityIQ Debug Pages are used for advanced configuration and for debugging. Click the drop down arrow next to **Configuration Objects** and select **UI Configuration** as shown:



c. You will now see an XML representation of the UIConfig object within IdentityIQ. Search for entry key with the name: **identityViewAttributes**. The keys are listed alphabetically. It will look like this:

```
<entry key="identityViewAttributes" value="name,firstname,
lastname,email,manager"/>
```

d. Edit the entry and change it to reflect the additional fields that we want to display Take care to type the names of the attributes accurately:

```
<entry key="identityViewAttributes" value="name,firstname,
lastname,email,manager,department,location,empId,region,
jobtitle,costcenter,status"/>
```

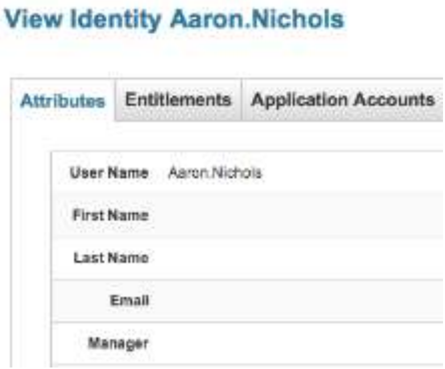
e. Scroll to the bottom of the page and **Save**

f. Navigate to **Identities → Identity Warehouse**, click any identity and confirm that new attributes are displayed. Notice that they are not yet populated with values.

Refresh and Populate the Identity Attributes

An **aggregation** task reads data from an external application, and a **refresh** task acts upon data within IdentityIQ. We aggregated the **HR System – Employees** and **Contractor Feed** applications, which read all of the information specified in each application schema and stored it as application account data. Now, based on our mappings, the application data will be promoted to (used to populate) the identity attributes.

1. Remember that when we aggregated the **HR System – Employees** and **Contractor Feed** applications, the identity attributes were not populated. This was because at the time of the aggregation, no mappings were defined. Once mappings are defined, aggregation will also populate the Identity Attributes.



View Identity Aaron.Nichols

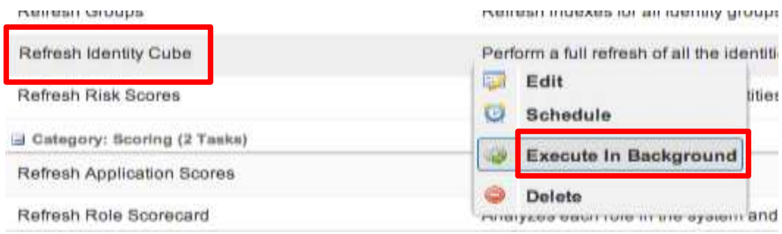
User Name: Aaron.Nichols
First Name:
Last Name:
Email:
Manager:

Note: When onboarding an application, aggregation without mappings is often done as part of the exploratory process to view the data prior to promoting it to identity attributes.

2. Refresh Identities and observe changes to the Identities

   a. Navigate to **Setup → Tasks** and scroll down looking for the **Refresh Identity Cube** task

   b. Right-Click and choose **Execute in Background**

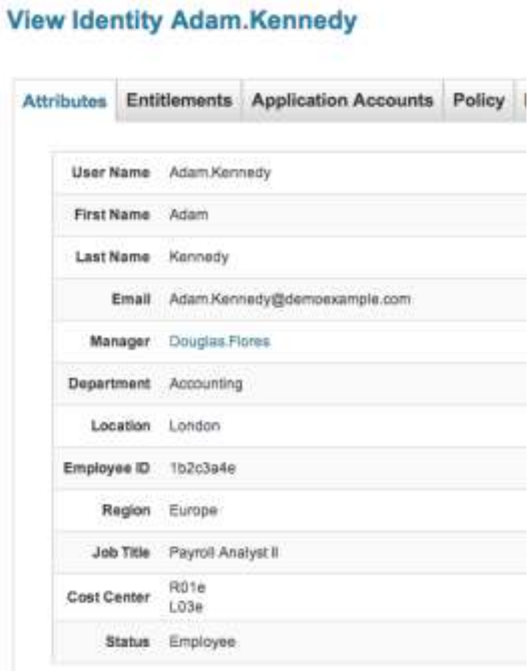


Refresh Groups	Refresh indexes for all identity groups
Refresh Identity Cube	Perform a full refresh of all the identiti
Refresh Risk Scores	...tities
Category: Scoring (2 Tasks)	
Refresh Application Scores	
Refresh Role Scorecard	Analyzes each role in the system and

   c. Visit **Task Results** and monitor the Refresh Identity Cube task.

   d. When the refresh is complete, Navigate to **Identities → Identity Warehouse**

e. Click **Adam.Kennedy** and see that this user now has populated values for all attributes, including the Manager and Status. Check a few other identities to see if they were loaded properly and that the Manager and Status fields are set.



View Identity Adam.Kennedy

User Name	Adam.Kennedy
First Name	Adam
Last Name	Kennedy
Email	Adam.Kennedy@demoexample.com
Manager	Douglas.Flores
Department	Accounting
Location	London
Employee ID	1b2c3a4e
Region	Europe
Job Title	Payroll Analyst II
Cost Center	R01e L03e
Status	Employee

i. Notice the manager name is a link to the manager's cube. IdentityIQ maintains the reporting hierarchy for use in approvals, escalations, etc.

ii. Notice Cost Center. Because we specified the identity attribute as multi-valued, it retains its multi-value representation that was first specified on the application schema.

3. One common thing to do is to customize the Identities summary page to include new Identity Attributes (like our new "Status" attribute) and whether or not the identity is "Authoritative" or not.

a. Go to the Debug page (use the shortcut in the Firefox browser.)

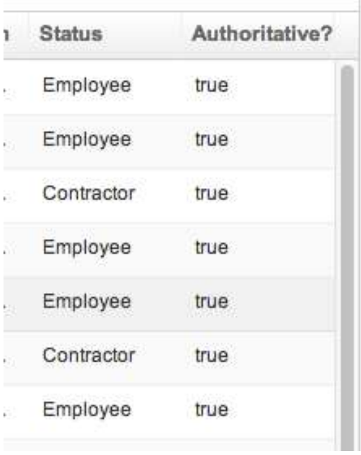
b. Click the drop down arrow next to **Configuration Objects** and select **UI Configuration**

c. Search for **identityTableColumns** and select **Next** until you find **<entry key="identityTableColumns">**

d. Add the following two lines to the end of the ColumnConfig entries for **identityTableColumns** and click **Save**

```
<ColumnConfig dataIndex="status" headerKey="Status" hideable="true"
property="status" sortProperty="status" sortable="true"/>
<ColumnConfig dataIndex="correlated" headerKey="Authoritative?"
hideable="true" property="correlated" sortProperty="correlated"
sortable="true"/>
```

e. Navigate to **Identities → Identity Warehouse** and confirm that your new columns show in the UI.



Status	Authoritative?
Employee	true
Employee	true
Contractor	true
Employee	true
Employee	true
Contractor	true
Employee	true

Note: UI Customizations like those we performed in this section can be useful for customizing the look and feel of the IdentityIQ product to reflect additional attributes that you add to the system. There is a Technical White Paper in the Compass portal that details all the UI Configuration options available to implementers.

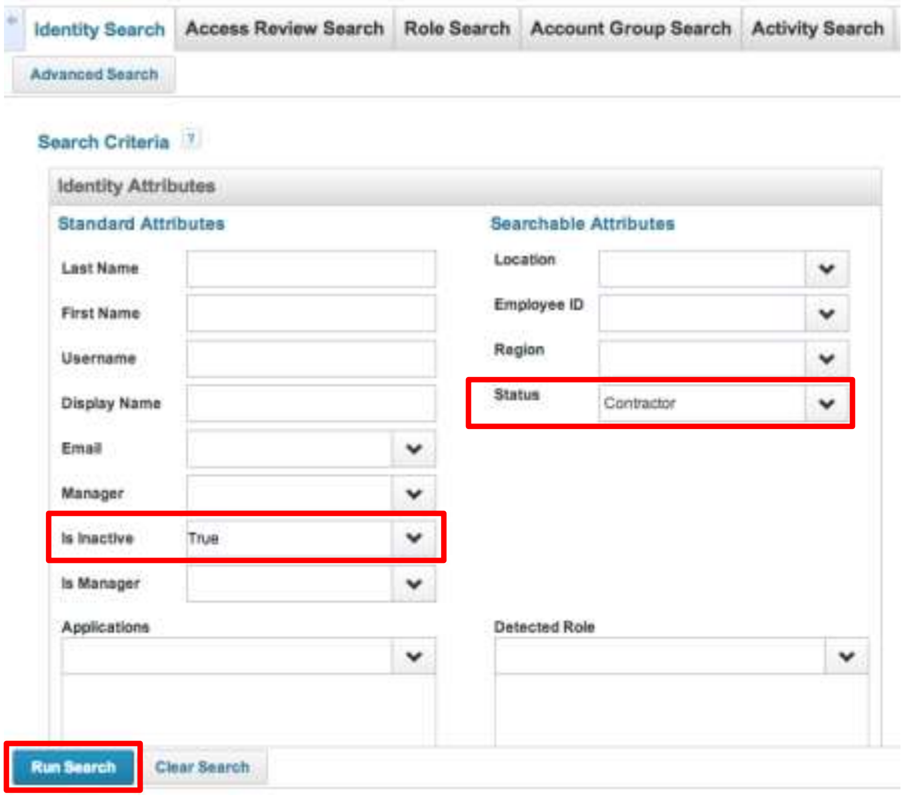
Investigate Your Data

Now that the account data specified in the Identity Mappings has been promoted to Identity Attributes, both the standard and the extended Identity Attributes are available for searching.

1. Navigate to **Intelligence → Advanced Analytics** and select the **Identity Search** tab.
   a. How many Searchable Attributes are now available? ______________
   b. Where did you specify that these attributes are searchable?

   ____

2. Using **Advanced Analytics**, **Identity Search** tab, answer the following questions. We will start with searching for inactive contractors.
   a. To ensure that no previously defined search parameters impact your search, a good practice is to clear the search parameters prior to searching. At the bottom left, select **Clear Search.**
      i. Set **Is Inactive** to **True**
      ii. Set **Status** to **Contractor**
      iii. Click **Run Search**

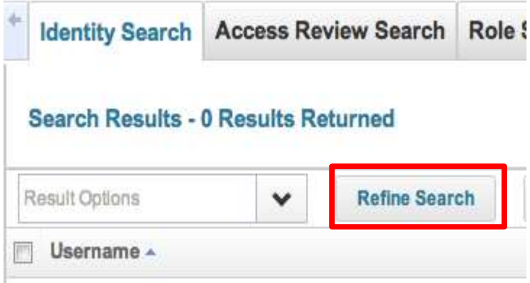


Is Inactive: True

Status: Contractor

   b. How many *inactive* contractors are there? ______________

c. Click **Refine Search** to return to the **Search Criteria** page, and then click **Clear Search** to reset everything.

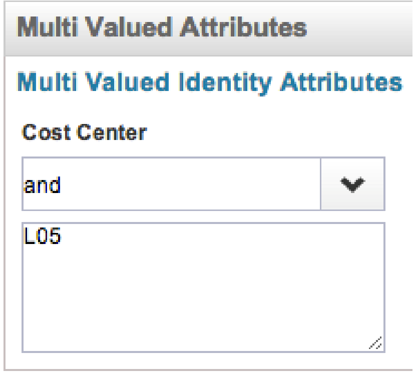


Search Results - 0 Results Returned

d. Now on your own, answer the following question:

How many *inactive employees* are there? ______________

3. You configured Cost Center as a multi-valued attribute and thus, because it is multi-valued, it is also searchable. Click **Refine Search**, and on the Identity Search tab, scroll down to **Multi Valued Attributes.**



Cost Center: and

L05

a. Click **Clear Search**, and then in the Cost Center **Value** box, enter **L05**

b. Select **Run Search**

c. How many identities contain the *L05 cost center*? ______________

Handy Tip: When working through these exercises, it is common to make some mistakes. You can always clear out all the identities in the system by using the IdentityIQ console.

Start the IIQ Console by using either method:

(1) **./iiq console -j** (from within the /WEB-INF/bin directory)
Note: The *-j* option enables using the arrow keys to page through commands entered during the session.

(2) Use the **IIQ Console** shortcut from the Desktop of the training environment

From within the console, run **delete identity *** to clear out all Identities from the system.

Note: use this option cautiously as this will remove all identities other than **spadmin**, which is a protected identity. If identities are used as values for other fields (such as an Application owner), the field will be emptied and must be reset.

Once you’ve cleared everything out, you can then re-run the aggregation and identity refresh tasks to reload the Identity Cubes.

Exercise #5: Loading and Correlating the Financials Application

Objective

In this exercise, we will load and correlate the Financials application. This application is used by a subset of people at the company (mainly those in the Finance department). This application data includes entitlement data (typically data that represents authorization -- for example, an attribute on an application that gives you “accounts payable” access to that application) as well as account data.

Overview

When loading a non-authoritative application, it is necessary to correlate user accounts from this new application to existing Identity Cubes. We will do this by defining an Account Correlation configuration when we configure each application. Account Correlation can be configured as a simple attribute mapping or, for more complicated examples, we can implement Account Correlation as a rule. In this section we will use an attribute mapping to correlate accounts.

As an example, the data for the Financial application looks like this:

```
employeeId,dbId,app2_privileged,acct_lastLogin,app2_service,app2_inactive,groupmbr,use
rName
1a2c3a,112,false,04/17/2008 21:26:43,false,false,AcctsPayable,RichardJackson
1a2c3a,112,true,04/17/2008 21:26:43,true,true,AcctsReceivable,RichardJackson
1a2c3a,112,true,04/17/2008 21:26:43,false,false,PayrollAnalyis,RichardJackson
1a2c3a,112,false,04/17/2008 21:26:43,false,false,PlanReview,RichardJackson
1a2c3b,113,false,04/17/2008 21:26:43,false,false,AcctsReceivable,MariaWhite
1a2c3c,114,true,04/17/2008 21:26:43,false,false,DPA,CharlesHarris
1a2c3c,114,false,04/17/2008 21:26:43,false,false,FinancialAnalyis,CharlesHarris
1a2c3c,114,false,04/17/2008 21:26:43,false,false,FinancialPlanning,CharlesHarris
1a2c3c,114,false,04/17/2008 21:26:43,false,false,PlanReview,CharlesHarris
1a2c3c,114,false,04/17/2008 21:26:43,false,false,Strategy&Planning,CharlesHarris
```

...

Notice that there are multiple rows for the same users. Charles Harris has 5 rows in the data because he is a member of five separate groups on the Financial application. Because of this, we will also implement merging of the data when we read in the accounts from this application.

Also, notice the employeeId field. We will use this value as our correlating value to link these accounts to our existing Identity Cubes.

After loading this application, we will see if we have any orphan accounts in the system (those accounts that cannot be linked to existing identity cubes) and will use manual correlation to deal with these accounts appropriately.

Define the Financials Application

1. For all new application definitions, we will not walk you through the process step-by-step, but will provide you with a table detailing the application settings.

    a. Create a new Application using information from the following table:

Application Applications → Application Definition →Add New Application	
Name:	**Financials**
Owner:	**The Administrator**
Description:	**Finance Application**
Application Type (Connector:)	**Delimited File**
Connector Attributes: Configuration → Settings → File	
File Path	**/home/spadmin/ImplementerTraining/data/Finance-users.csv**
Delimiter	**,**
File has column header on first line	**Checked**
Configuration→Settings→Merging	
Data needs to be Merged	**Checked**
Index Column	**dbId**
Data sorted by the indexColumn(s)?	**Checked**
Which Columns should be merged?	**groupmbr**

2. Next, we will configure the attributes that we will read from the **Financials** application.

    a. Navigate to **Configuration → Schema**

    b. For the account schema, select **Discover Schema Attributes.** Configure the schema settings as shown:

        i. Native Object Type: **account**

        ii. Identity Attribute: **dbId**

        iii. Display Attribute: **userName**

        iv. Under attributes, for the **groupmbr** attribute, on the far right, click **Edit** and check **Managed, Entitlement** and **Multi-Valued**

v. Complete the following sentences:

1. When ______________________________________ is specified, it indicates that the values of that item will also be included in the Entitlement Catalog.

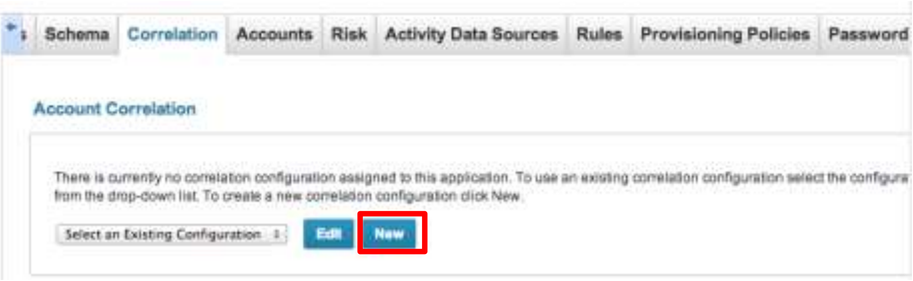
2. As a result of the ____ designation, the entitlements are also listed on the Entitlements tab on the Identity Cube and will ultimately be available for certification and other usage.

c. Select **Preview** and notice that the data has been merged (look at the **groupmbr** field). Preview displays the first ten resource object records.

3. Define an Account Correlation configuration to match accounts from this application to existing Identity Cubes

a. Navigate to **Correlation**

b. Click the **New** button



There is currently no correlation configuration assigned to this application. To use an existing correlation configuration select the configura
from the drop-down list. To create a new correlation configuration click New.

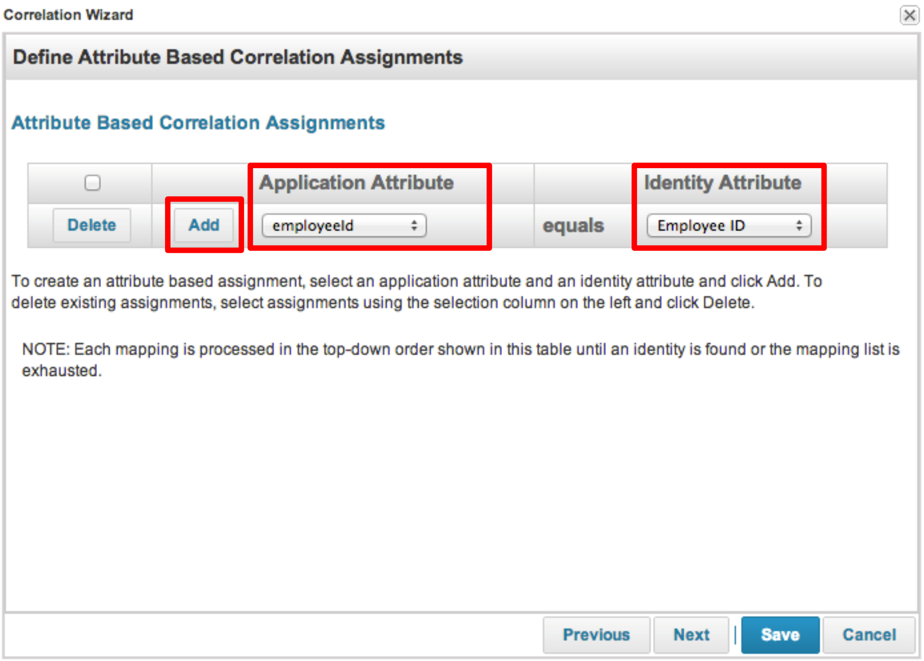
c. This will bring up the **Correlation Wizard**

d. Click **Next**

e. When prompted, enter a name for this configuration: **Financial Correlation**

f. Click **Next** twice

g. When prompted, configure the **Attribute Based Correlation Assignment** as shown, then click **Add**



Correlation Wizard

Define Attribute Based Correlation Assignments

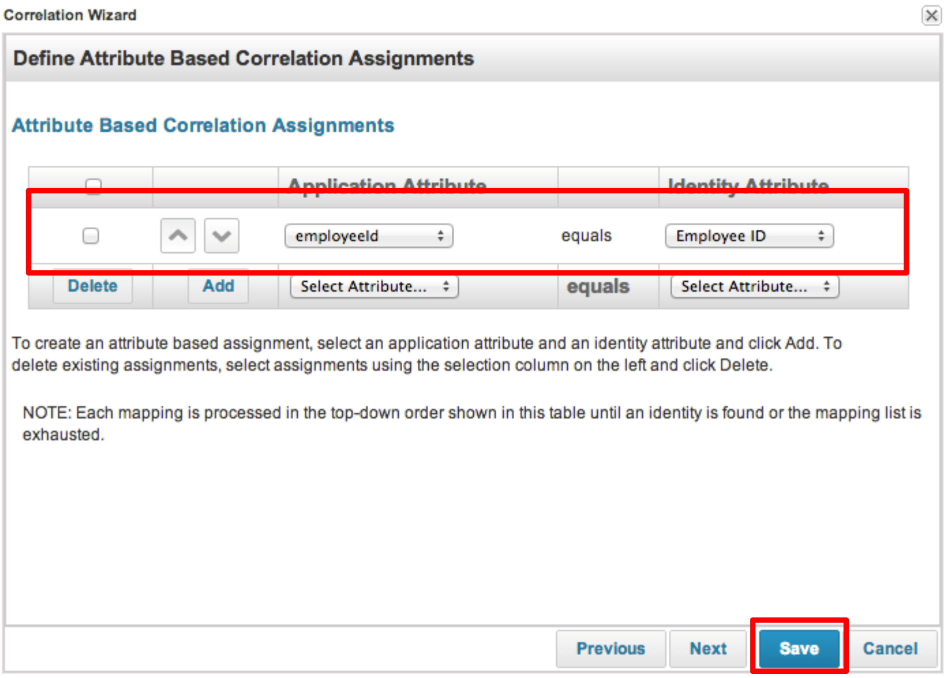
Attribute Based Correlation Assignments

□		Application Attribute		Identity Attribute
Delete	Add	employeeId	equals	Employee ID

To create an attribute based assignment, select an application attribute and an identity attribute and click Add. To delete existing assignments, select assignments using the selection column on the left and click Delete.

NOTE: Each mapping is processed in the top-down order shown in this table until an identity is found or the mapping list is exhausted.

h. Confirm that the mapping is configured as shown below, then click **Save**



Correlation Wizard

Define Attribute Based Correlation Assignments

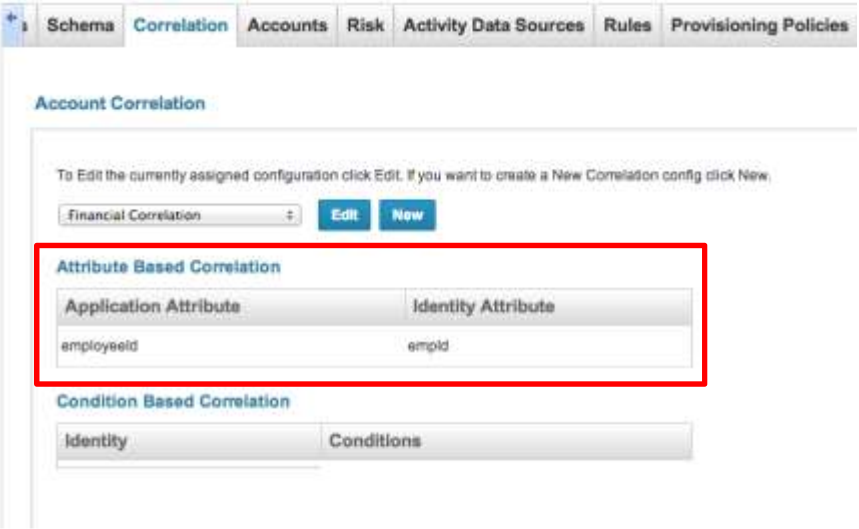
Attribute Based Correlation Assignments

□		Application Attribute		Identity Attribute
□	˄ ˅	employeeId	equals	Employee ID
Delete	Add	Select Attribute...	equals	Select Attribute...

To create an attribute based assignment, select an application attribute and an identity attribute and click Add. To delete existing assignments, select assignments using the selection column on the left and click Delete.

NOTE: Each mapping is processed in the top-down order shown in this table until an identity is found or the mapping list is exhausted.

i. At this point, your correlation should look like this:



Account Correlation

To Edit the currently assigned configuration click Edit. If you want to create a New Correlation config click New.

Financial Correlation

Attribute Based Correlation

Application Attribute	Identity Attribute
employeeId	empId

Condition Based Correlation

Identity	Conditions

j. Click **Save** to save the application definition

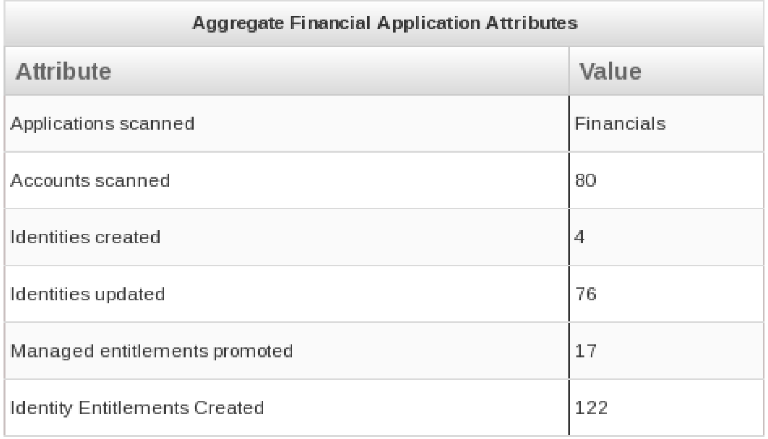
Aggregate from the Financials Application

1. We will now create a task to aggregate accounts from the Financials application. For all new task definitions, we will not walk you through the process step-by-step, but will provide you with a table detailing all the settings:

Task Setup → Tasks → New Task	
Type:	**Account Aggregation**
Name:	**Aggregate Financial Application**
Description:	**Task to aggregate accounts from the Financials application.**
Select applications to scan:	**Financials**
Detect deleted accounts:	**Checked**
Disable optimization of unchanged accounts:	**Checked**
Promote managed attributes:	**Checked**

2. Scroll down and select **Save and Execute** and click **OK**

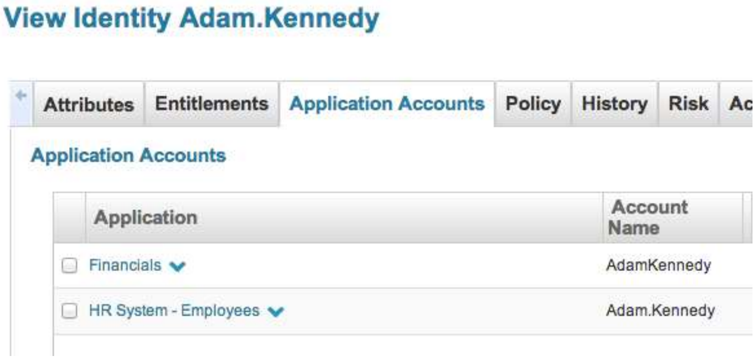
3. Go to **Task Results** and confirm that the results are shown:



Aggregate Financial Application Attributes	
Attribute	Value
Applications scanned	Financials
Accounts scanned	80
Identities created	4
Identities updated	76
Managed entitlements promoted	17
Identity Entitlements Created	122

Confirm that Accounts and managed entitlements were properly loaded

1. Navigate to **Identities → Identity Warehouse** and find **Adam.Kennedy**
2. Click **Application Accounts** and check to make sure that the Financials account shows up:

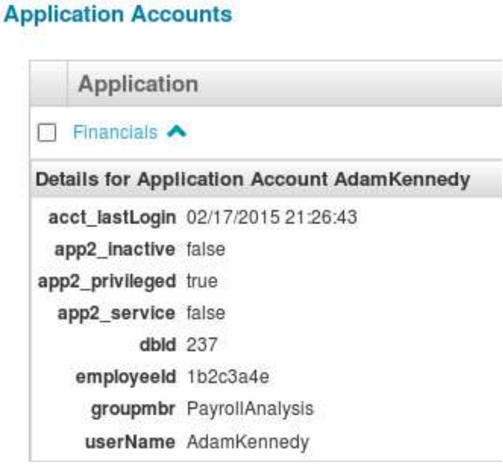


View Identity Adam.Kennedy

Application Accounts

	Application	Account Name
☐	Financials	AdamKennedy
☐	HR System - Employees	Adam.Kennedy

3. Click the **Financials** account and check the attributes for the **Financials** application:



Application Accounts

	Application
☐	Financials

Details for Application Account AdamKennedy

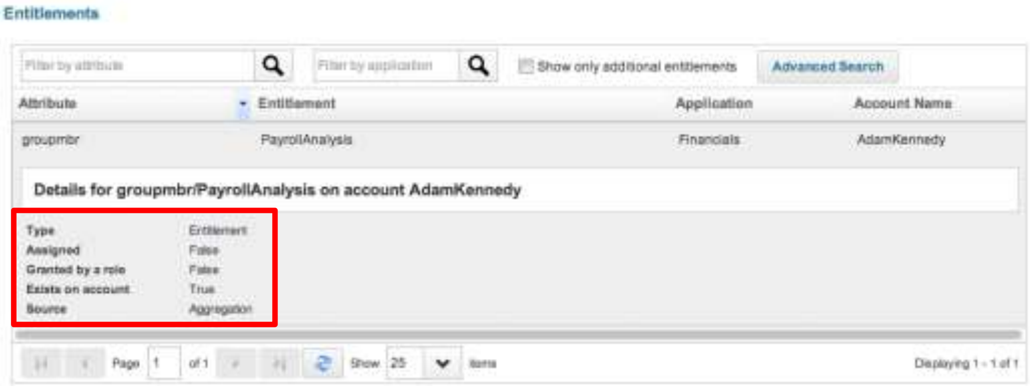
acct_lastLogin	02/17/2015 21:26:43
app2_inactive	false
app2_privileged	true
app2_service	false
dbId	237
employeeId	1b2c3a4e
groupmbr	PayrollAnalysis
userName	AdamKennedy

4. Within the same Identity (Adam.Kennedy), click **Entitlements** and then under **Entitlements,** select the **groupmbr** entitlement to expand the information related to the entitlement.

   a. What is the source of this entitlement?

   ____

   b. Was this entitlement assigned through IdentityIQ? (circle one) YES / NO



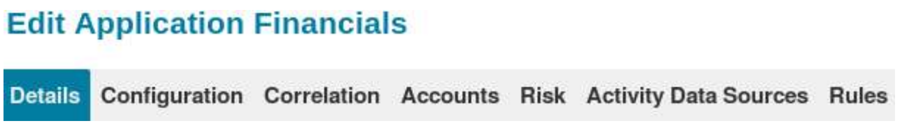
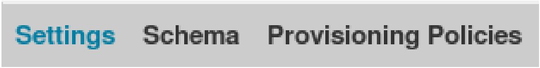
Attribute	Entitlement	Application	Account Name
groupmbr	PayrollAnalysis	Financials	AdamKennedy

Details for groupmbr/PayrollAnalysis on account AdamKennedy

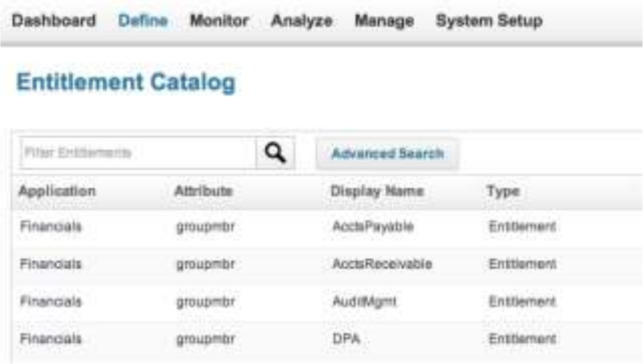
Type	Entitlement
Assigned	False
Granted by a role	False
Exists on account	True
Source	Aggregation

The **Source** is aggregation and **Assigned** is false. This means this is a detected entitlement that we discovered from the IT environment via **Aggregation.**

   c. Complete the following table. When you defined the Financials application, where did you configure IdentityIQ to track this entitlements information?

Which Application:	*Financials*
Circle the navigation path:	 Edit Application Financials Details Configuration Correlation Accounts Risk Activity Data Sources Rules
Circle the navigation path:	
Which Attribute:	
Option Selected:	

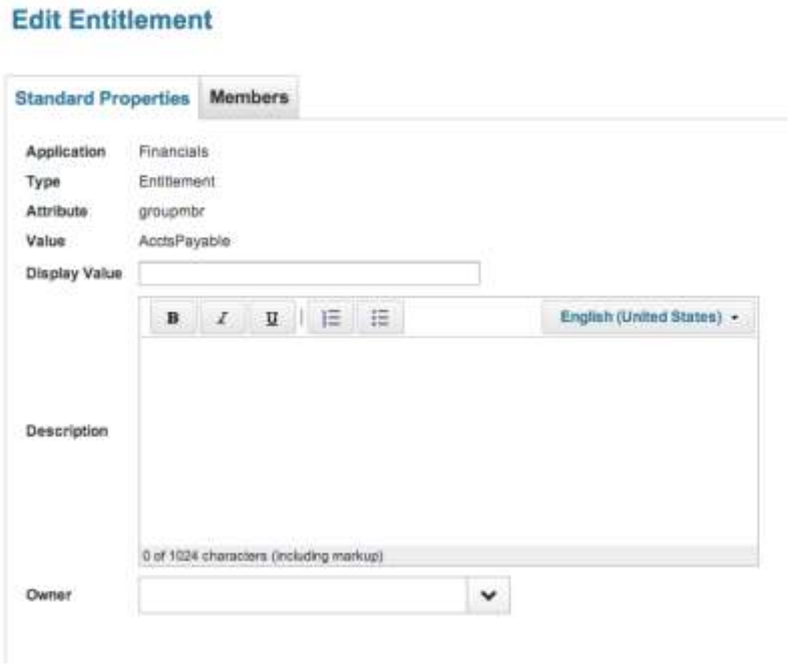
5. Navigate to **Applications → Entitlement Catalog** and notice that the entitlement values from the **Financials** application have been loaded. These values were loaded because we marked the **groupmbr** attribute as **Managed** in the application schema.



Entitlement Catalog

Application	Attribute	Display Name	Type
Financials	groupmbr	AcctsPayable	Entitlement
Financials	groupmbr	AcctsReceivable	Entitlement
Financials	groupmbr	AuditMgmt	Entitlement
Financials	groupmbr	DPA	Entitlement

6. In the entitlement catalog, we can assign owners to the entitlements (for approval purposes) and set multi-lingual descriptions. *Data in the Entitlement Catalog is foundational* for many other aspects of IdentityIQ including Policy, Risk, Certifications, Roles, and Lifecycle Manager.
   a. Select the **AcctsPayable** entitlement and view the options on the **Standard Properties** tab.
   b. Select the **Members** tab to view the identities with the Financials AcctsPayable entitlement.



Edit Entitlement

Application	Financials
Type	Entitlement
Attribute	groupmbr
Value	AcctsPayable
Display Value	
Description	0 of 1024 characters (including markup)
Owner	

Exercise #6: Loading and Correlating the PAM Application

Objective

The objective of this exercise is to load an application that includes a more complicated definition including Accounts, Account Groups and Permissions.

Overview

The client has requested that we load an application that maintains application permissions in account groups. An account group is an indirect method of defining access to a system. A user will have an account on a system with entitlement to a defined set of account groups. These account groups indirectly define the user's access to the application.

The PAM application data feed consists of two delimited files:

- PAM-users.csv – Contains user Account information for the PAM application. Parts of the account information that we will read are account groups (specifically Permission Groups).
- PAM-group-permissions.csv – Contains information about the Permission Groups themselves including the targets the Permission Group allows access to

When onboarding the PAM application, we will need to define two schemas, one for the accounts that we are aggregating and another for the account groups that we are aggregating.

Here is an example of what the data is that we will be modeling:

In the PAM-users.csv file, we have a user **Mary.Johnson** who has an account on the PAM application. Her account includes two **Permission Group** values

```
ACCOUNTING
FINANCE
```

In the permissions.csv file, we have permission groups defined as such:

```
“TEST01”,”20080211”,”ACCOUNTING”,”ACCTSYSTEM”
“TEST01”,”20080211”,”FINANCE”,”FINSYSTEM”
```

These lines define the specific permissions for both the **ACCOUNTING** and **FINANCE** permission groups.

Create the Base PAM Application

1. Create a new application definition based on the following table:

Application
Define → Applications → New Application

Name:	**PAM**
Owner:	**Patrick.Jenkins**
Revoker:	**Albert.Woods**
Description:	**Financially significant application**
Application Type (Connector:)	**Delimited File**
Connector Attributes: Configuration → Settings → File	
File Path	**/home/spadmin/ImplementerTraining/data/PAM-users.csv**
Delimiter	**,**
File has column header on first line	**Checked**
Configuration → Settings → Merging	
Data needs to be Merged	**Checked**
Index Column	**User ID**
Data sorted by the indexColumn(s)?	**Checked**
Which Columns should be merged?	**Permission Group**

Configure Account Schema for PAM Application

1. Navigate to **Configuration → Schema**

2. Fill in the configuration details:

Name	**Value**	**Description**
Native Object Type	account	This is predefined in most connectors. Ensures that the correct information is accessed. For Delimited File you will enter either “account” or “group” as appropriate.
Identity Attribute	User ID	The Identity Attribute defines which attribute will be used to determine the uniqueness of the account. You could think of this as the primary key for the application accounts. In this case, we are using the “User ID” which is unique for each user.
Display Attribute	User Name	A more “friendly” identifier for the account used in the GUI.

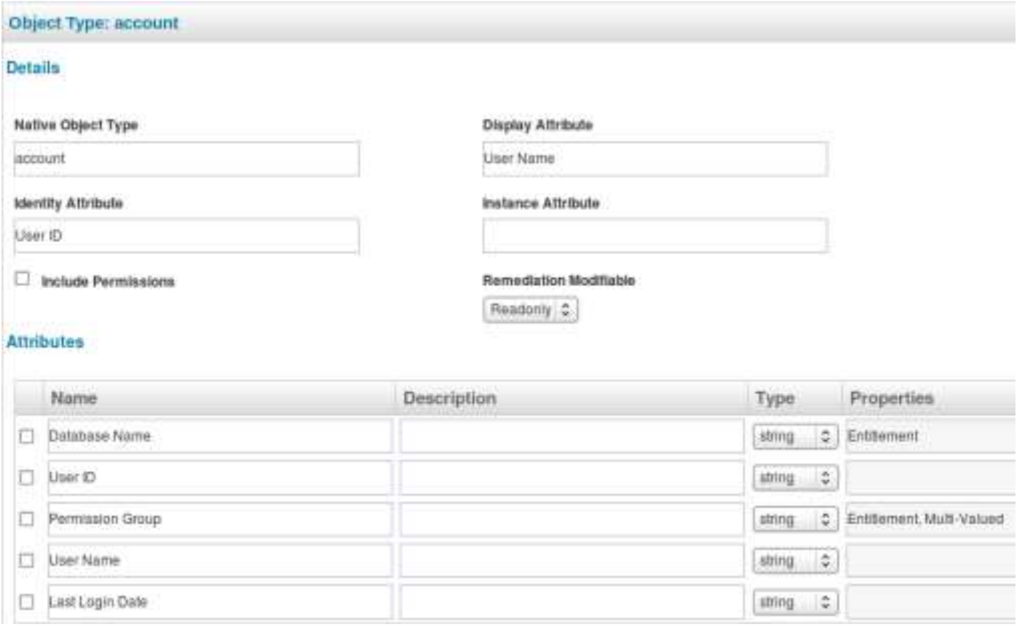
3. Fill in the following attributes for the account schema using the table below. To enter each new attribute, use the **Add New Schema Attribute** button.

Note: Manually specifying the schema attributes, as done here, is useful when only a small subset of the attributes contained in a data file are to be added to IdentityIQ.

Name	Type	Entitlement	Multi-Valued
Database Name	String	Checked	
User ID	String		
Permission Group	Group*	Checked	Checked
User Name	String		
Last Login Date	String		

*The Group Type is not available until we create the group object. Once the group schema has been created, configuration of the Permission Group attribute can be completed.

4. At this point, your account schema should look like this:



Object Type: account

Details

Native Object Type: account

Display Attribute: User Name

Identity Attribute: User ID

Instance Attribute:

□ Include Permissions

Remediation Modifiable: Readonly

Attributes

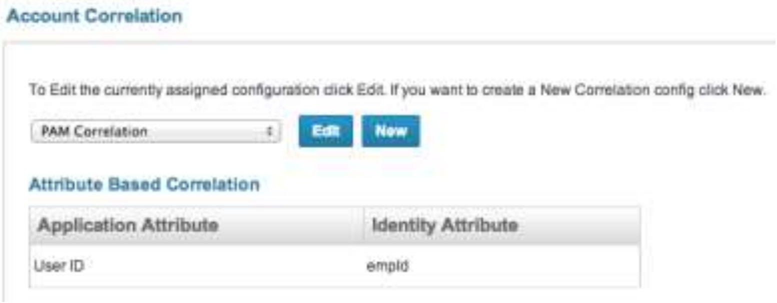
	Name	Description	Type	Properties
□	Database Name		string	Entitlement
□	User ID		string	
□	Permission Group		string	Entitlement, Multi-Valued
□	User Name		string	
□	Last Login Date		string	

   a. Preview the accounts and confirm that account data will be loaded properly.

5. Navigate to **Correlation** and configure a new **Account Correlation** based on the following attributes.

Account Correlation	
Name	**PAM Correlation**
User ID	**User ID** equals **Employee ID**

6. When done, your correlation should look like this.



Account Correlation

To Edit the currently assigned configuration click Edit. If you want to create a New Correlation config click New.

PAM Correlation

Attribute Based Correlation

Application Attribute	Identity Attribute
User ID	empId

7. Scroll to the bottom and **Save** the application definition.

Use connectorDebug to Confirm PAM Application Account Data

The console **connectorDebug** command iterates over all of the accounts in an application and displays the results of the iteration, resource objects, to the console screen instead of loading it into IdentityIQ. Like the Preview button, this command can be extremely useful when doing initial work onboarding applications to make sure that you are properly reading and manipulating the data and that your schema is correct.

1. Open up the IIQ console by one of the following two methods:

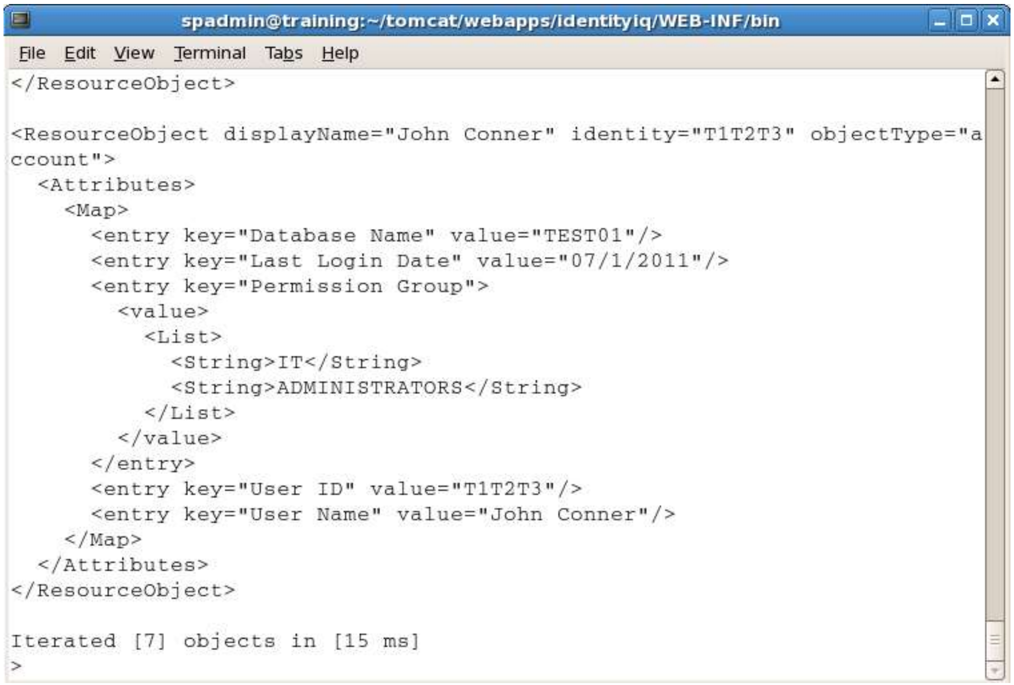
   Option 1: Click the **IIQ Console** shortcut on the desktop

   Option 2: Using a Linux terminal window, run ./**iiq console -j** from **/home/spadmin/tomcat/webapps/identityiq/WEB-INF/bin**

2. Within the IIQ Console, run the following command:

   ```
   >connectorDebug PAM iterate
   ```

3. In the screen shot, circle the multi-valued attribute and all of its values.


spadmin@training:~/tomcat/webapps/identityiq/WEB-INF/bin
File Edit View Terminal Tabs Help

```
</ResourceObject>

<ResourceObject displayName="John Conner" identity="T1T2T3" objectType="a
ccount">
  <Attributes>
    <Map>
      <entry key="Database Name" value="TEST01"/>
      <entry key="Last Login Date" value="07/1/2011"/>
      <entry key="Permission Group">
        <value>
          <List>
            <String>IT</String>
            <String>ADMINISTRATORS</String>
          </List>
        </value>
      </entry>
      <entry key="User ID" value="T1T2T3"/>
      <entry key="User Name" value="John Conner"/>
    </Map>
  </Attributes>
</ResourceObject>

Iterated [7] objects in [15 ms]
>
```

Note: Best practice is to start by using Preview (the only option for those who do not have console access). The connectorDebug command is useful for debugging problems with multi-valued attributes and when debugging build map rules (you see the output from the rule interspersed with the account information).

Caution: The connectorDebug command will display *all* accounts for the application, whether 200 or 200,000; whereas Preview displays the first 10 accounts.

Configure the Group Data Source for PAM Application

Now that we have successfully modeled the account schema, we need to model the group schema for the PAM application. This will involve defining the schema and attributes we will read in from the PAM-group-permissions.csv file.

1. Navigate to **Applications → Application Definition → PAM → Configuration**
2. Click **Add Object Type**. Name the object type **group**
3. Scroll down to **Object Type: group** and configure the connector as shown:

Name	Value	Description
File Path	/home/spadmin/ImplementerTraining/data /PAM-group-permissions.csv	Location of the file containing the groups
Delimiter	,	
File has column header on first line	Checked	

4. Scroll down and select **Save**

Configure Group Schema for PAM Application

1. Navigate to **PAM → Configuration → Schema** and scroll down to **Object Type: group**
2. Fill in the configuration details:

Name	Value	Description
Native Object Type	group	This is predefined in most connectors. Ensures that the correct information is accessed. For Delimited File you will enter either “account” or “group” as appropriate.
Identity Attribute	Permission Group	The Identity Attribute defines which attribute will be used to determine the uniqueness of the information. You could think of this as the primary key for the groups.
Display Attribute	Permission Group	In this case, the Permission Group name is acceptable for the Display Attribute

Note: Be mindful that you are editing the correct object type. After each entry, the user interface returns the focus to the top of the screen, where you may accidentally modify the *account* object type rather than the *group* object type.

3. Fill in the following attributes for the group schema using the table below. To enter each new attribute, use the **Add New Schema Attribute** button. Alternatively, if the number of unwanted attributes is small (as in this case), use **Discover Schema Attributes** and delete the unwanted attributes from the schema.

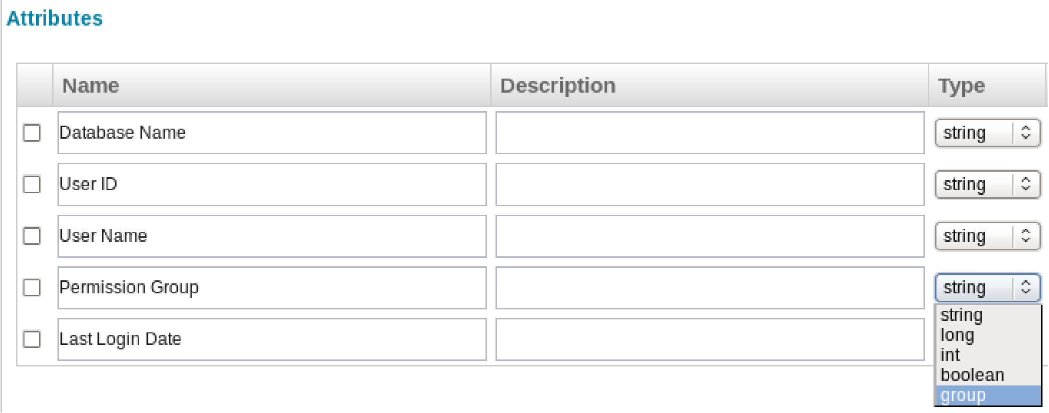
Name	Type	Entitlement	Multi-Valued
Permission Group	String		
Permission Rights	String	Checked	Checked
Description	String		

4. Select the **Preview** button. This command will iterate through the group information, and you can confirm that the Groups and Permissions are being extracted from the file correctly.

5. Now that the group objects exists, *group* will be available to configure as a type for the account attributes.

   a. Scroll up to the account schema

   b. Change the **Permission Group** attribute type to **group**



Attributes

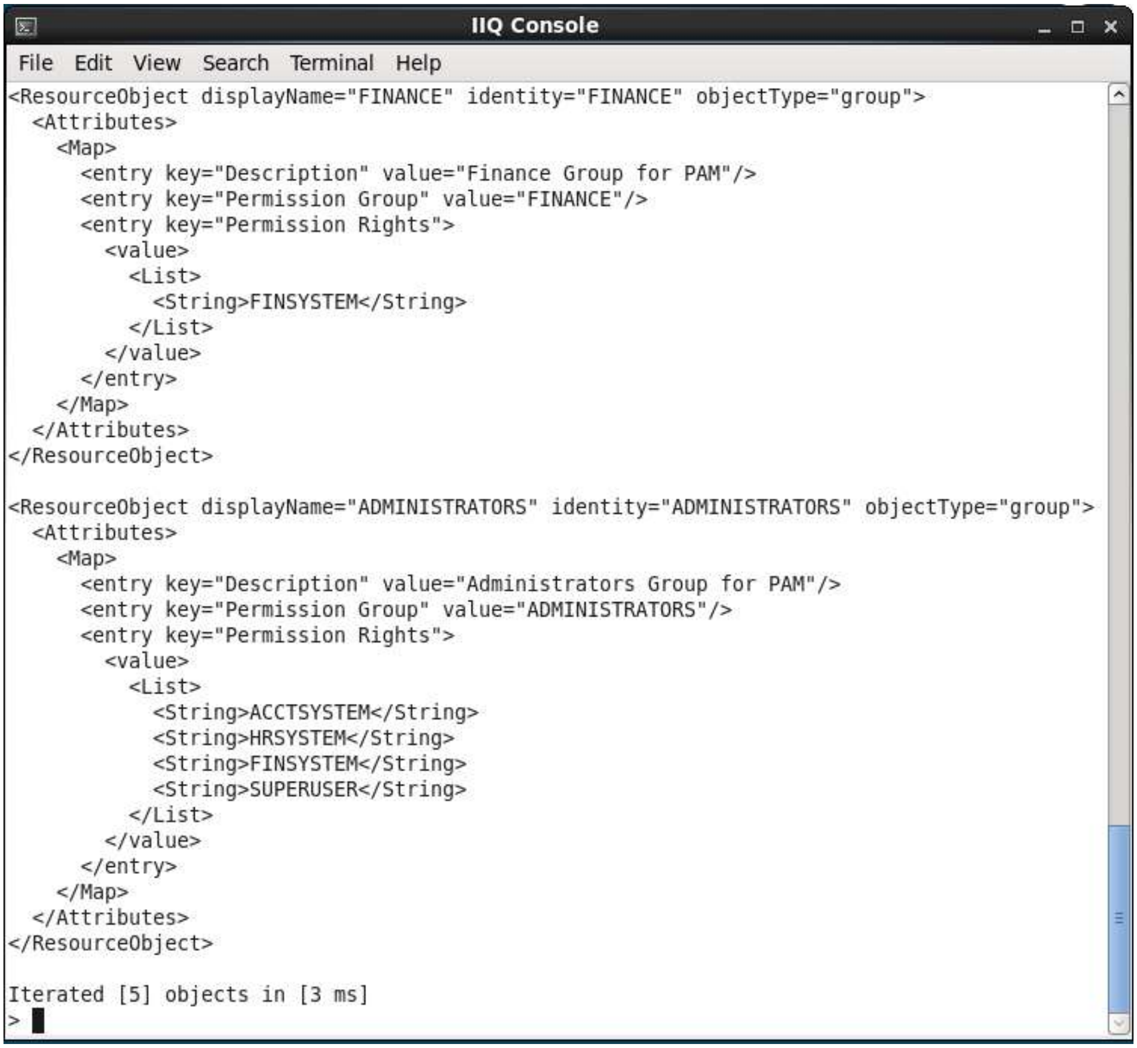
	Name	Description	Type
□	Database Name		string
□	User ID		string
□	User Name		string
□	Permission Group		string
□	Last Login Date		

6. Scroll to the bottom and **Save** the application definition.

7. Go to the IIQ Console and run the following command:

```
>connectorDebug PAM iterate group
```

8. Like **Preview Groups**, running the **connectorDebug** command with the group option will iterate through the group information being read from the input file.



```
<ResourceObject displayName="FINANCE" identity="FINANCE" objectType="group">
  <Attributes>
    <Map>
      <entry key="Description" value="Finance Group for PAM"/>
      <entry key="Permission Group" value="FINANCE"/>
      <entry key="Permission Rights">
        <value>
          <List>
            <String>FINSYSTEM</String>
          </List>
        </value>
      </entry>
    </Map>
  </Attributes>
</ResourceObject>

<ResourceObject displayName="ADMINISTRATORS" identity="ADMINISTRATORS" objectType="group">
  <Attributes>
    <Map>
      <entry key="Description" value="Administrators Group for PAM"/>
      <entry key="Permission Group" value="ADMINISTRATORS"/>
      <entry key="Permission Rights">
        <value>
          <List>
            <String>ACCTSYSTEM</String>
            <String>HRSYSTEM</String>
            <String>FINSYSTEM</String>
            <String>SUPERUSER</String>
          </List>
        </value>
      </entry>
    </Map>
  </Attributes>
</ResourceObject>

Iterated [5] objects in [3 ms]
>
```

9. Now, we need to aggregate this data into the system. Remember, that the **connectorDebug** command only shows us a debug view of the data; we haven't actually loaded any Account and Account Group data into the system.

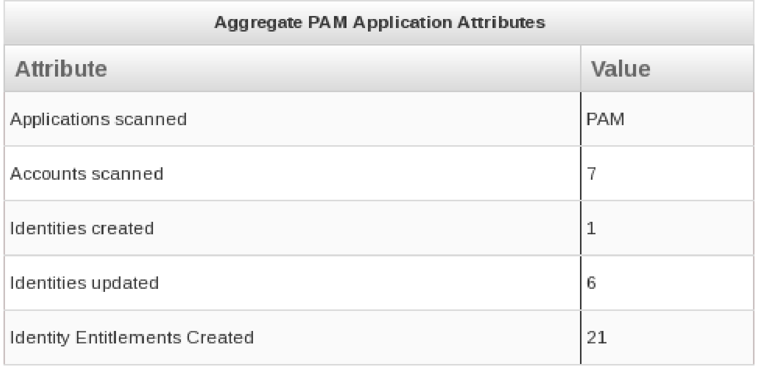
Aggregate PAM Accounts and Groups

1. In order to aggregate Accounts and Groups from the PAM application, we will need two tasks: one to aggregate accounts and the other to aggregate account groups.

2. Create a task to aggregate accounts from the PAM application:

Task Setup → Tasks → New Task	
Type:	**Account Aggregation**
Name:	**Aggregate PAM Application**
Description:	**Task to aggregate accounts from the PAM application.**
Select applications to scan:	**PAM**
Detect deleted accounts:	**Checked**
Disable optimization of unchanged accounts:	**Checked**

3. Scroll down and select **Save and Execute** and click **OK.** Go to **Task Results** and wait for the task to finish. Confirm the results:



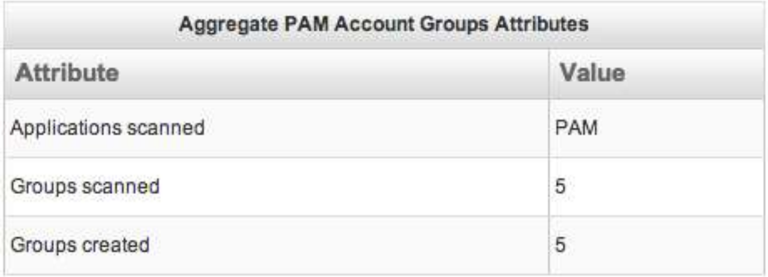
Aggregate PAM Application Attributes

Attribute	Value
Applications scanned	PAM
Accounts scanned	7
Identities created	1
Identities updated	6
Identity Entitlements Created	21

4. Create another task for group aggregation as shown:

Task Setup → Tasks → New Task	
Type:	**Account Group Aggregation**
Name:	**Aggregate PAM Account Groups**
Description:	**Task to aggregate account groups from the PAM application.**
Select applications to scan:	**PAM**
Detect deleted account groups	**Checked**
Automatically promote descriptions to this locale:	**en_US**
Description attribute (default "description")	**Description**

5. Scroll down and select **Save and Execute** and select **OK.** Go to the **Task Results** tab and confirm the results:



Aggregate PAM Account Groups Attributes

Attribute	Value
Applications scanned	PAM
Groups scanned	5
Groups created	5

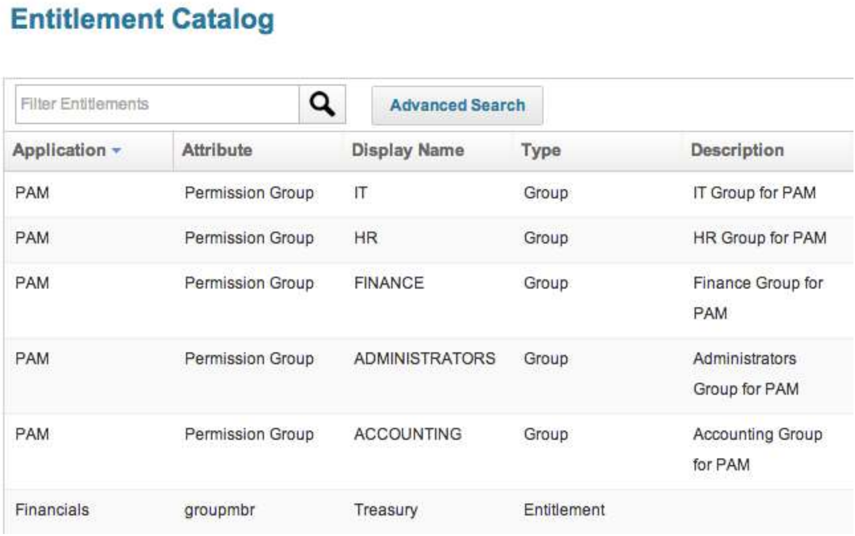
6. Confirm that your PAM accounts and groups were loaded properly

    a. Go to **Identities → Identity Warehouse** and look up the user: **Carl.Foster**

    b. Click **Application Accounts** and select the **PAM** application

    c. Select **<u>ACCOUNTING</u>** to see the rights for the **ACCOUNTING** group and all other members that share the same **ACCOUNTING group**

    d. Navigate to **Applications → Entitlement Catalog** and see all the Account Groups defined for the PAM application. You can sort on the application name column to see them grouped together.



Entitlement Catalog

Application	Attribute	Display Name	Type	Description
PAM	Permission Group	IT	Group	IT Group for PAM
PAM	Permission Group	HR	Group	HR Group for PAM
PAM	Permission Group	FINANCE	Group	Finance Group for PAM
PAM	Permission Group	ADMINISTRATORS	Group	Administrators Group for PAM
PAM	Permission Group	ACCOUNTING	Group	Accounting Group for PAM
Financials	groupmbr	Treasury	Entitlement	

    e. You can click any of the Account Groups to see the permissions and members for any of these Account Groups

Exercise #7: Onboarding JDBC Applications

Objective

The objective of this exercise is to onboard account data out of multiple JDBC resources.

Overview

For this application, we will onboard two JDBC applications:

- TRAKK – An application that we will configure completely by hand
- PRISM – An application that we will configure by loading a ready to go XML file. Loading the XML format is a common way to load applications in real environments, especially during migration from development to QA to production

Configure the Application and Connector for the TRAKK Application

1. Create a new application definition based on the following table:

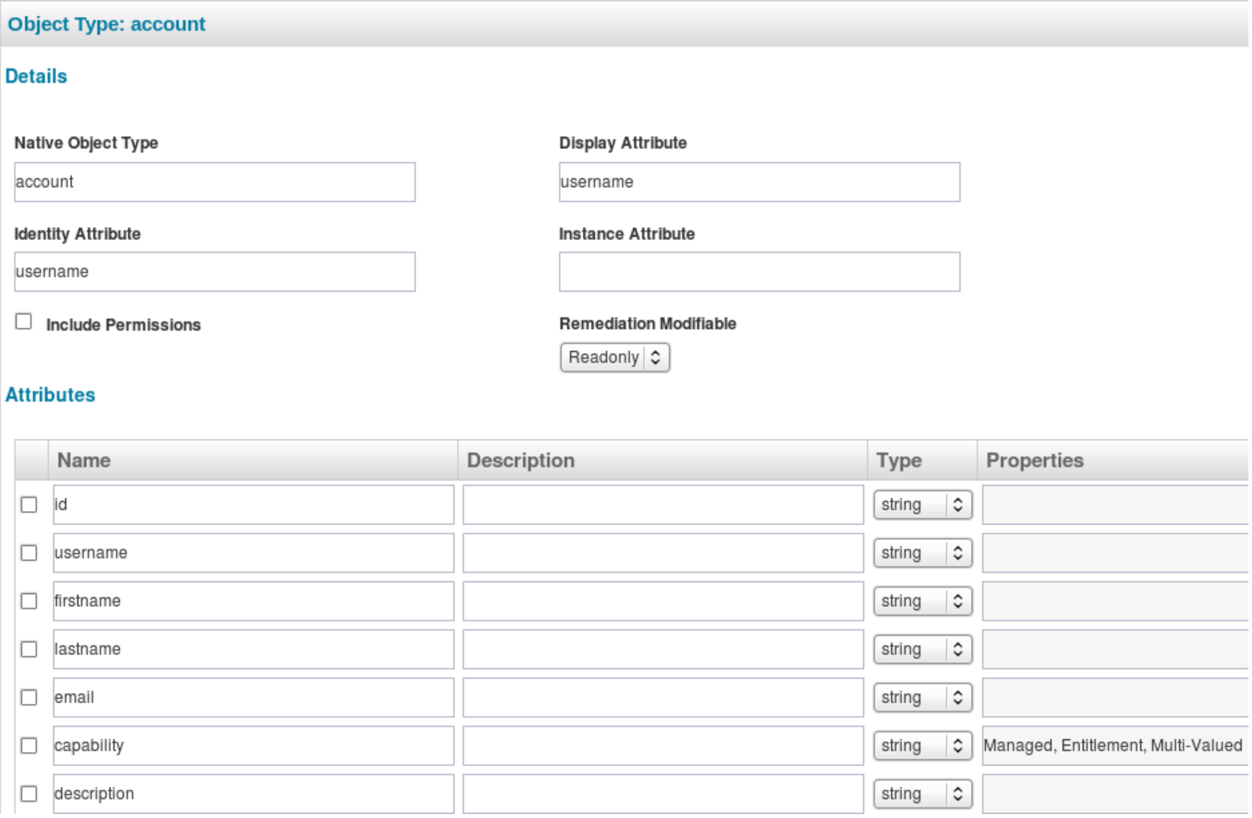
Application Applications → Application Definition → Add New Application	
Name:	**TRAKK**
Owner:	**The Administrator**
Application Type (Connector:)	**JDBC**
Description:	**The TRAKK Time Tracking Application**
Connector Attributes: Configuration→Settings→account→Settings	
JDBC Connection Settings	
Connection User	**root**
Connection Password	**root**
Database URL	**jdbc:mysql://localhost/trakk**
JDBC Driver	**com.mysql.jdbc.Driver**
Query Settings	
SQL Statement	**select * from users left outer join capabilities on users.id = capabilities.id order by users.username;**
getObjectSQL	**select * from users left outer join capabilities on users.id = capabilities.id where users.username = '$(identity)';**

2. Test the connection parameters: Click **Test Connection.** This option is useful to verify the connection between IdentityIQ and the application without performing an account preview.

Configure Account Schema for the TRAKK Application

1. Navigate to **Configuration → Schema,** for the account schema, click **Discover Schema Attributes**. Fill out the schema according to the table below:

Account Schema				
Native Object Type	**account**			
Identity Attribute	**username**			
Display Attribute	**username**			
Attributes	**Type**	**Managed**	**Entitlement**	**Multi-Valued**
id	**string**			
username	**string**			
firstname	**string**			
lastname	**string**			
email	**string**			
capability	**string**	**checked**	**checked**	**checked**
description	**string**			



Object Type: account

Details

Native Object Type: account

Display Attribute: username

Identity Attribute: username

Instance Attribute:

☐ Include Permissions

Remediation Modifiable: Readonly

Attributes

	Name	Description	Type	Properties
☐	id		string	
☐	username		string	
☐	firstname		string	
☐	lastname		string	
☐	email		string	
☐	capability		string	Managed, Entitlement, Multi-Valued
☐	description		string	

2. Select **Preview**.

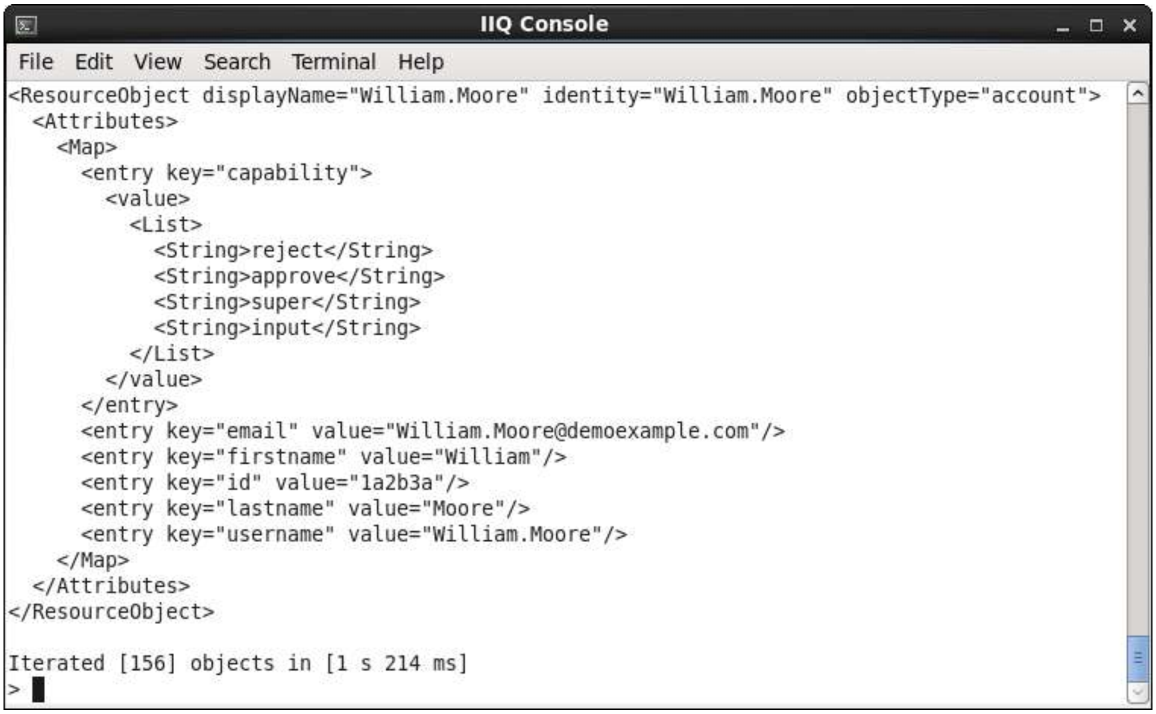
3. Navigate to **Configuration → Settings → account → Merging** and complete the merge options as follows:

Connector Attributes: Attributes Tab	
Data needs to be merged	**checked**
Index Column	**username**
Which Columns should be merged?	**capability**

4. Click **Save** to save your work for the TRAKK application.

5. Next you will test the SQL commands you entered in the JDBC Query Settings. Launch the **IIQ Console** and run the following commands:

```
>connectorDebug TRAKK iterate
```

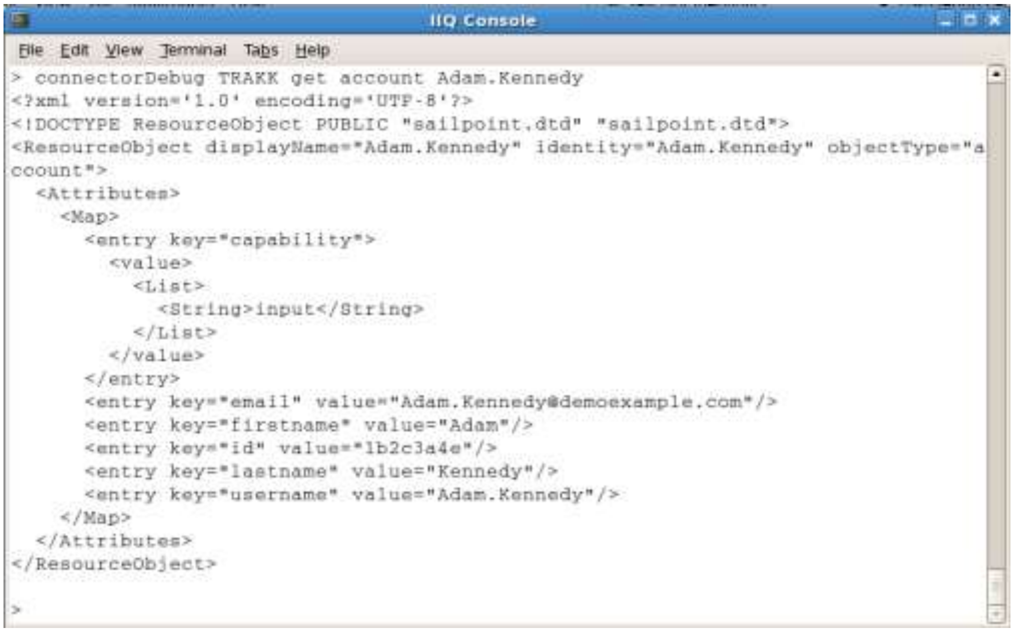


```
<ResourceObject displayName="William.Moore" identity="William.Moore" objectType="account">
  <Attributes>
    <Map>
      <entry key="capability">
        <value>
          <List>
            <String>reject</String>
            <String>approve</String>
            <String>super</String>
            <String>input</String>
          </List>
        </value>
      </entry>
      <entry key="email" value="William.Moore@demoexample.com"/>
      <entry key="firstname" value="William"/>
      <entry key="id" value="1a2b3a"/>
      <entry key="lastname" value="Moore"/>
      <entry key="username" value="William.Moore"/>
    </Map>
  </Attributes>
</ResourceObject>

Iterated [156] objects in [1 s 214 ms]
>
```

This connectorDebug command grabbed all the accounts out of the JDBC resource using the **SQL Statement** query that you configured in the TRAKK application.

```
>connectorDebug TRAKK get account Adam.Kennedy
```



```
> connectorDebug TRAKK get account Adam.Kennedy
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE ResourceObject PUBLIC "sailpoint.dtd" "sailpoint.dtd">
<ResourceObject displayName="Adam.Kennedy" identity="Adam.Kennedy" objectType="a
ccount">
  <Attributes>
    <Map>
      <entry key="capability">
        <value>
          <List>
            <String>input</String>
          </List>
        </value>
      </entry>
      <entry key="email" value="Adam.Kennedy@demoexample.com"/>
      <entry key="firstname" value="Adam"/>
      <entry key="id" value="1b2c3a4e"/>
      <entry key="lastname" value="Kennedy"/>
      <entry key="username" value="Adam.Kennedy"/>
    </Map>
  </Attributes>
</ResourceObject>

>
```

This connectorDebug command grabbed a single account from the JDBC resource using the **getObjectSQL** query that you configured in the TRAKK application. Some connectors can support the random access of a single record. For the JDBC connector, the **getObjectSQL** query supports this random access operation.

6. If you run both of the connectorDebug commands successfully, continue; otherwise re-check your configuration of the TRAKK application.

Configure Correlation Rule for the TRAKK Application

1. Configure Correlation for this new application

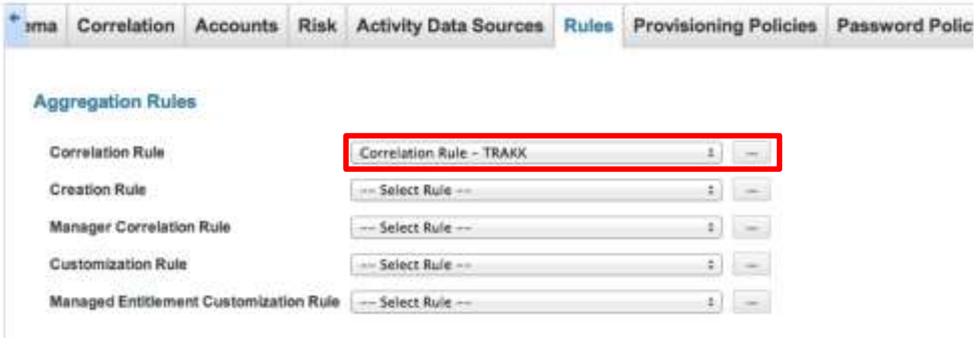
   a. Navigate to **Applications → Application Definition → TRAKK → Rules** and create a new Correlation Rule by clicking the **…** :

      i. Name: **Correlation Rule - TRAKK**

      ii. Rule: copy and paste beanshell from **/home/spadmin/ImplementerTraining/beanshell/Correlation-Rule-Trakk.txt**

      iii. Click **Save**

   b. Choose the correlation rule once you save it, by choosing it in the dropdown list.



Aggregation Rules

Correlation Rule	Correlation Rule - TRAKK
Creation Rule	-- Select Rule --
Manager Correlation Rule	-- Select Rule --
Customization Rule	-- Select Rule --
Managed Entitlement Customization Rule	-- Select Rule --

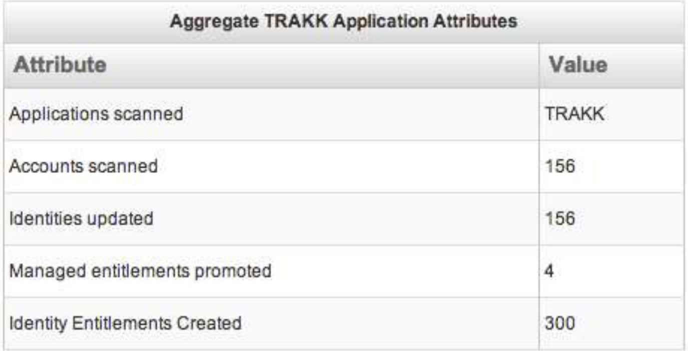
   c. Click **Save** to save the TRAKK application.

Aggregate Accounts from TRAKK

1. Configure a new task using the information from the table below

Task Setup → Tasks → New Task	
Type:	**Account Aggregation**
Name:	**Aggregate TRAKK Application**
Description:	**Task to aggregate accounts from the TRAKK application.**
Select applications to scan:	**TRAKK**
Detect deleted accounts:	**Checked**
Disable optimization of unchanged accounts:	**Checked**
Promote managed attributes:	**Checked**

2. Scroll down and select **Save and Execute.** Click **OK** and go to the **Task Results** tab and check the results:



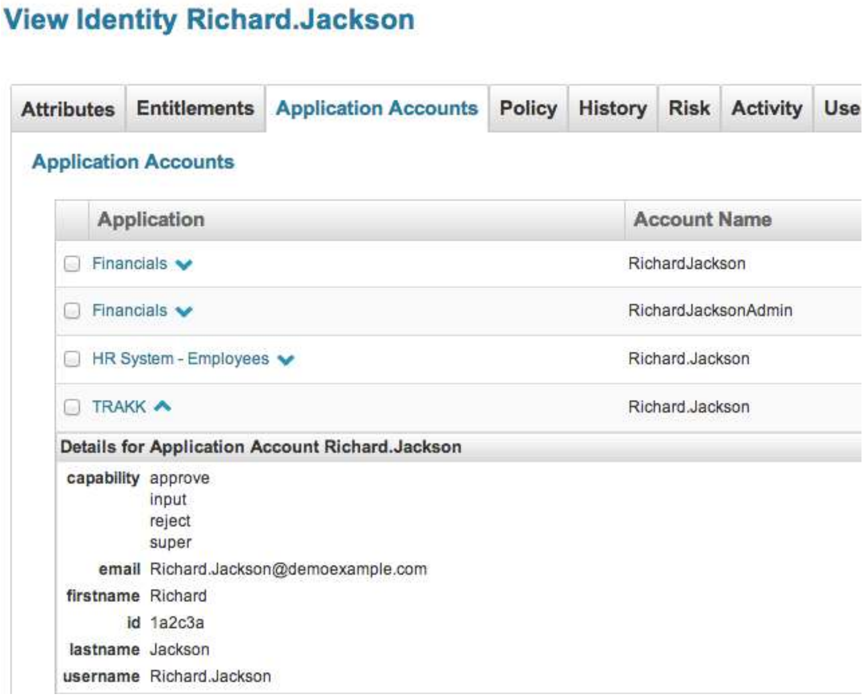
Aggregate TRAKK Application Attributes

Attribute	Value
Applications scanned	TRAKK
Accounts scanned	156
Identities updated	156
Managed entitlements promoted	4
Identity Entitlements Created	300

3. Verify Account Attributes and Entitlements

   a. Navigate to **Identities → Identity Warehouse** and look for Richard Jackson

   b. Click **Application Accounts** and **TRAKK** to verify that Richard has an account on TRAKK:



View Identity Richard.Jackson

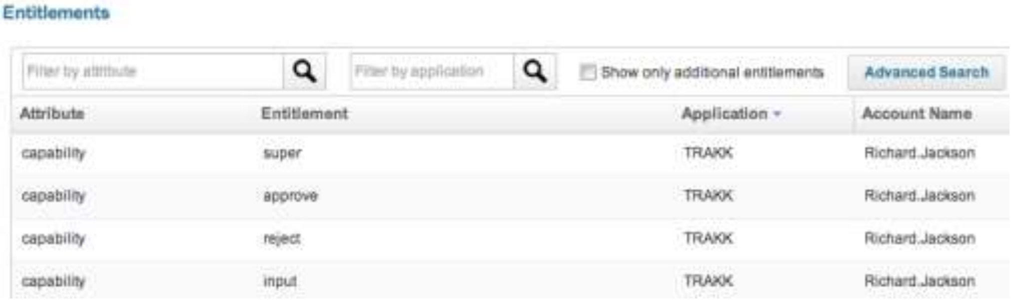
Application Accounts

Application	Account Name
□ Financials	RichardJackson
□ Financials	RichardJacksonAdmin
□ HR System - Employees	Richard.Jackson
□ TRAKK	Richard.Jackson

Details for Application Account Richard.Jackson

capability	approve input reject super
email	Richard.Jackson@demoexample.com
firstname	Richard
id	1a2c3a
lastname	Jackson
username	Richard.Jackson

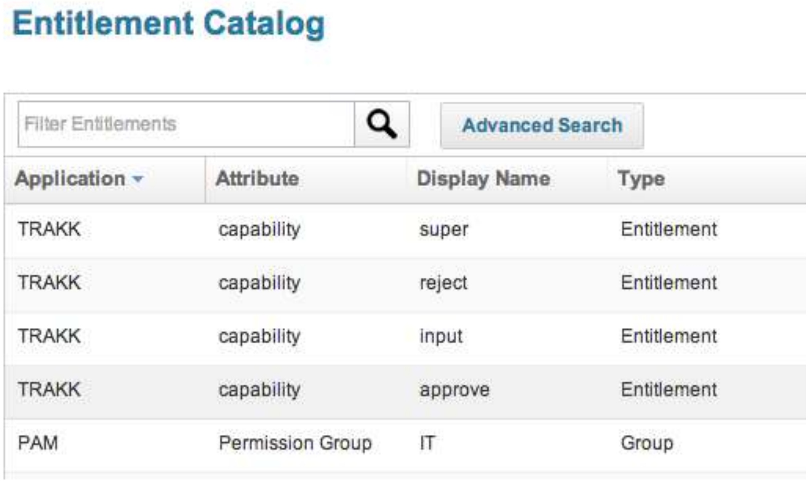
c. Click the **Entitlements** tab to verify that the entitlements were properly created for **Richard.Jackson**



Entitlements

Attribute	Entitlement	Application	Account Name
capability	super	TRAKK	Richard.Jackson
capability	approve	TRAKK	Richard.Jackson
capability	reject	TRAKK	Richard.Jackson
capability	input	TRAKK	Richard.Jackson

d. Navigate to **Applications → Entitlement Catalog** to confirm that the managed entitlements for the **TRAKK** application were loaded:



Entitlement Catalog

Application	Attribute	Display Name	Type
TRAKK	capability	super	Entitlement
TRAKK	capability	reject	Entitlement
TRAKK	capability	input	Entitlement
TRAKK	capability	approve	Entitlement
PAM	Permission Group	IT	Group

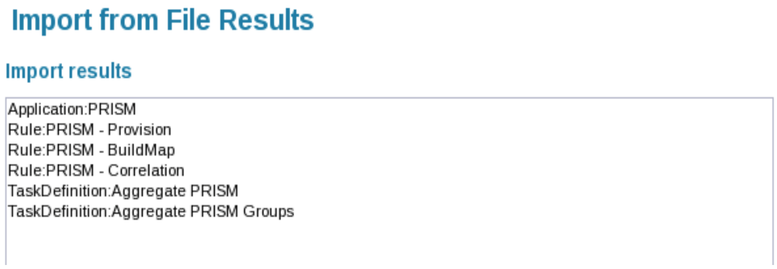
Loading the PRISM Application

The PRISM application will be loaded as a pre-defined XML file that contains the entire application definition for the PRISM application. Take a look at the XML file using the editor provided in the VM. Note that the XML file is one large XML file containing 6 individual SailPoint objects.

1. Navigate to ⚙ → **Global Settings → Import from File** and in the **Import Objects** section, click **Browse…** and import the following file:

   /home/spadmin/ImplementerTraining/config/PRISM/PRISM.xml

2. Once the file is done loading, check the output to confirm that everything loaded ok.



Import from File Results

Import results

```
Application:PRISM
Rule:PRISM - Provision
Rule:PRISM - BuildMap
Rule:PRISM - Correlation
TaskDefinition:Aggregate PRISM
TaskDefinition:Aggregate PRISM Groups
```

3. This single XML file contained the following:

   a. An Application definition

   b. Rules for Correlation, Buildmap and Provisioning

   c. Tasks to Aggregate Accounts and Account Groups

4. Navigate to **Applications → Application Definition** and spot check the PRISM application.

   a. Who is the owner for the PRISM application?

   ____

   We also want IdentityIQ administrators to be able to act as owners for the PRISM application. We'll note that we need to create a workgroup to use for ownership.

   b. Which connector is used by the PRISM application?

   ____

   c. Is PRISM an authoritative application?   ____Yes ___No

   d. How are we correlating the accounts? (circle one)

   Correlation Configuration   Correlation Rule

   e. Look at the schema. Which attribute will be managed in the Entitlement Catalog?

   ____

5. Check the **Entitlement Catalog**.

   a. Are there any PRISM entitlements listed?   ____Yes ___No
      As you continue with this exercise, think about why or why not PRISM entitlements are listed. It will become clear as you work through the next few instructions.

6. Create a workgroup for PRISM ownership.

   a. Navigate to **Setup → Groups**, select the **Workgroups** tab, and click **Create Workgroup**.

   b. Define the workgroup as follows:

      i. Name: **PRISM Application Owners**

      ii. Owner: **spadmin**

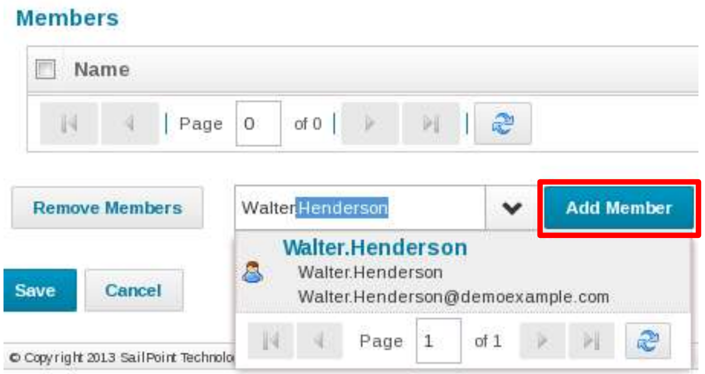
      iii. Description: **Group for all users with ownership for the PRISM application.**

      iv. Group Email**: prism_owners@example.com**

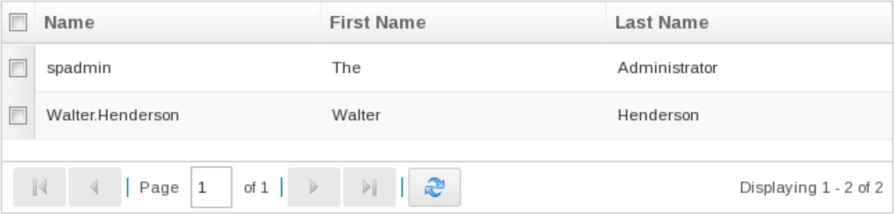
      v. Capabilities: **Application Administrator**

c. Add members to the workgroup:

  i. Search for **Walter Henderson** and click **Add Member**



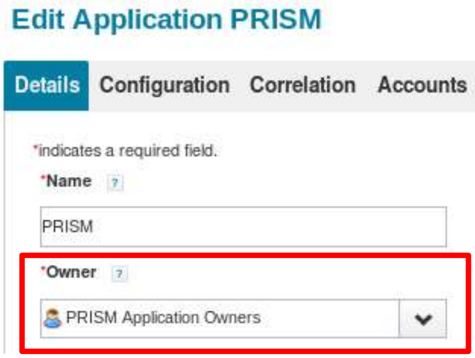
  ii. Add one more member: **spadmin**



□	Name	First Name	Last Name
□	spadmin	The	Administrator
□	Walter.Henderson	Walter	Henderson

d. **Save** the workgroup.

e. Navigate to the PRISM application, change the owner to the **PRISM Application Owners** workgroup, and **Save**.



Edit Application PRISM

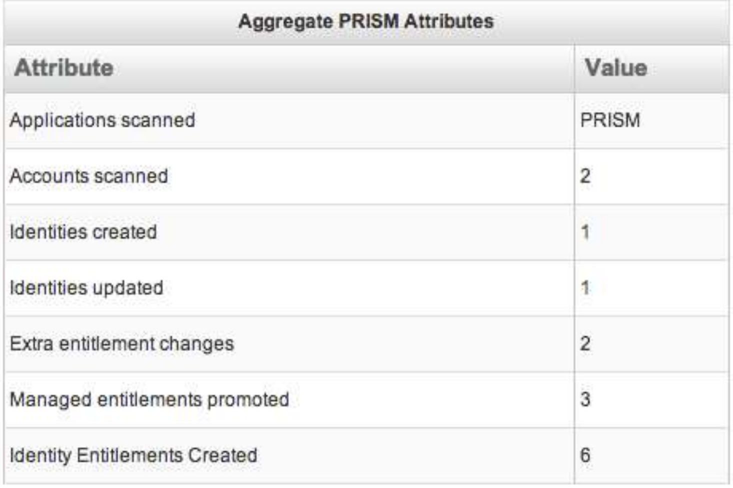
*indicates a required field.

*Name: PRISM

*Owner: PRISM Application Owners

7. Next, aggregate the PRISM application by running the following tasks in order. Wait for each to finish before running the next:

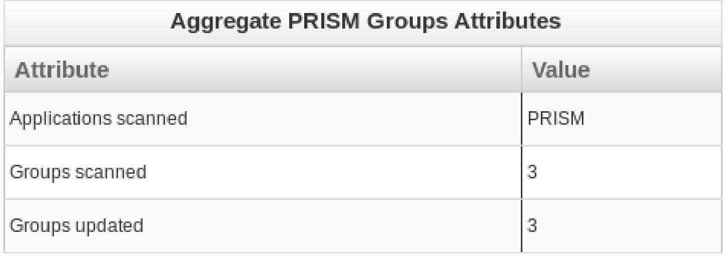
   a. **Aggregate PRISM**



Aggregate PRISM Attributes

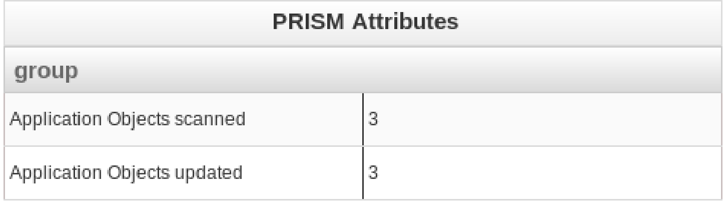
Attribute	Value
Applications scanned	PRISM
Accounts scanned	2
Identities created	1
Identities updated	1
Extra entitlement changes	2
Managed entitlements promoted	3
Identity Entitlements Created	6

   b. **Aggregate PRISM Groups**



Aggregate PRISM Groups Attributes

Attribute	Value
Applications scanned	PRISM
Groups scanned	3
Groups updated	3

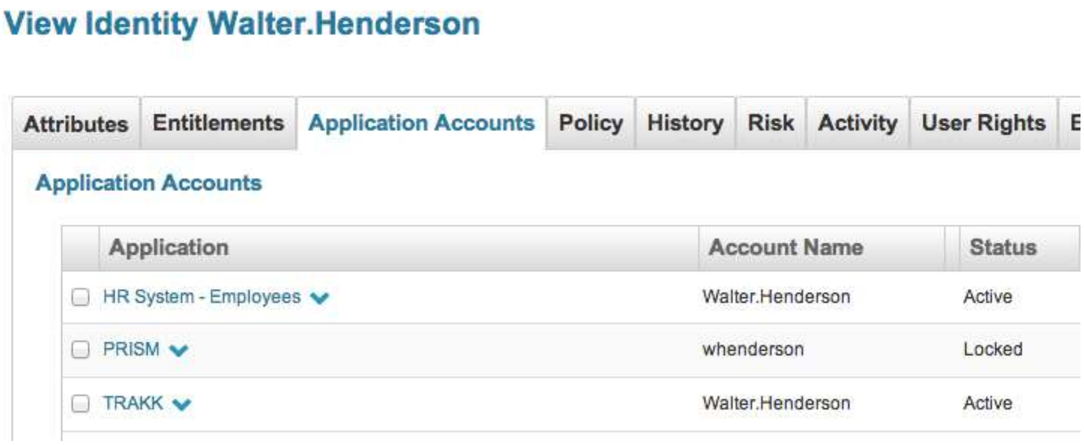


PRISM Attributes

group	
Application Objects scanned	3
Application Objects updated	3

8. Once the Aggregations are complete, check the following to make sure everything went smoothly

    a. Check **Walter.Henderson** to make sure he has an account on PRISM.

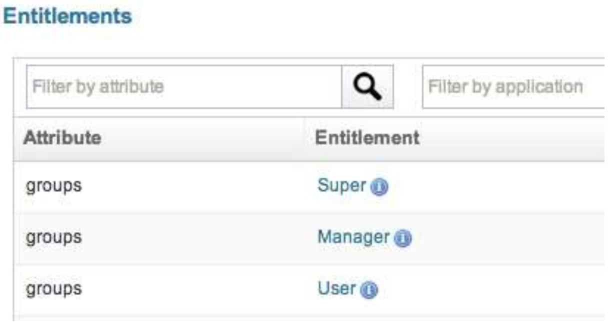


View Identity Walter.Henderson

Application Accounts

Application	Account Name	Status
□ HR System - Employees	Walter.Henderson	Active
□ PRISM	whenderson	Locked
□ TRAKK	Walter.Henderson	Active

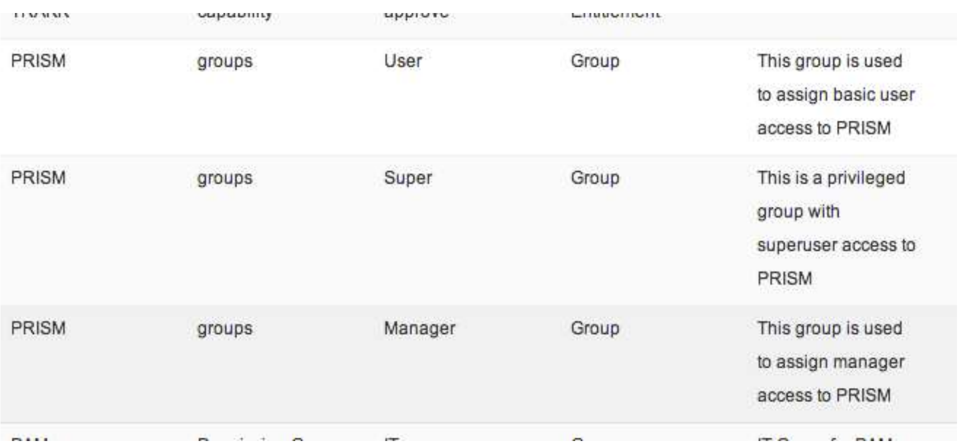
    b. Check **Walter.Henderson** to make sure he has entitlements for PRISM.



Entitlements

Attribute	Entitlement
groups	Super
groups	Manager
groups	User

    c. Check the **Entitlement Catalog** to make sure that the PRISM account groups were loaded properly.



PRISM	groups	User	Group	This group is used to assign basic user access to PRISM
PRISM	groups	Super	Group	This is a privileged group with superuser access to PRISM
PRISM	groups	Manager	Group	This group is used to assign manager access to PRISM

Exercise #8: Onboarding an LDAP Application

Objective

The objective of this exercise is to onboard account and group data out of an LDAP application.

Overview

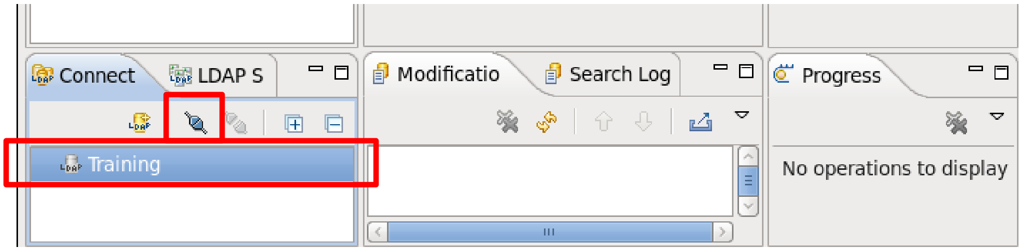
For this application, we will onboard an LDAP application using the LDAP direct connector.

Start the local LDAP Server

1. From a Linux command window, enter **StartLDAP**

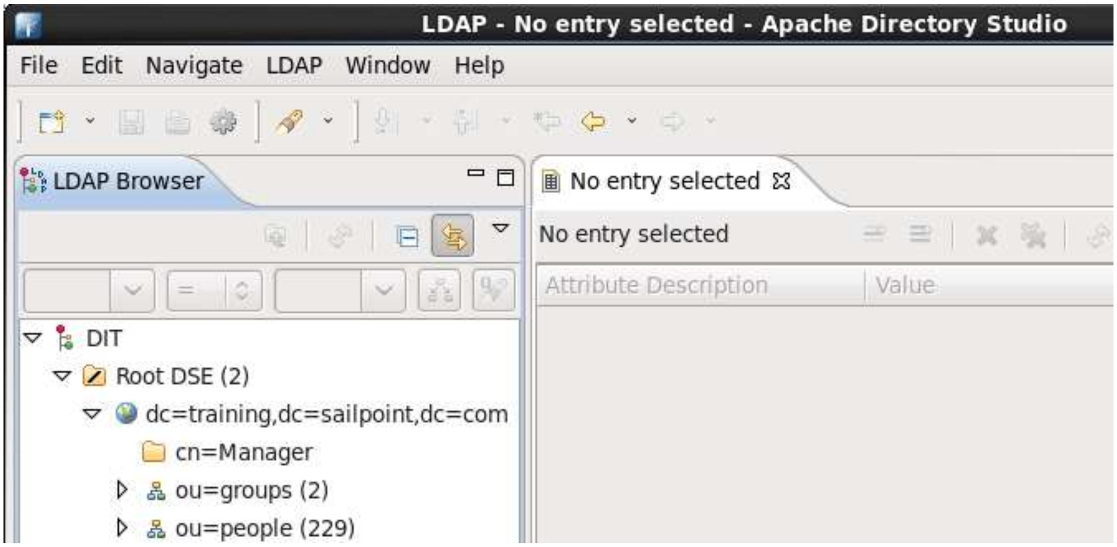
2. Use the desktop shortcut to launch the LDAP Browser

3. At the bottom of the **LDAP - Apache Directory Studio** window, click on **Training**, and then click on the **Open Connection button**



4. View the current **people** and **groups** in LDAP.

   a. Click on the arrow next to **dc=training,dc=sailpoint,dc=com**



   b. Expand and view the **people**.

c. Expand and list the **groups**: ____________________ ____________________

d. Close **LDAP - Apache Directory Studio**.

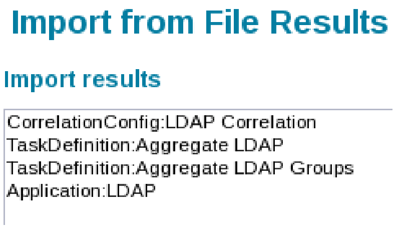
Loading the LDAP Application

The LDAP application will be loaded as a pre-defined XML file that contains the entire application definition for the LDAP application. Take a look at the XML file using the editor provided in the VM. Note that the XML file is one large XML file containing 4 individual SailPoint objects.

1. Navigate to ⚙ → **Global Settings** → **Import from File** and in the **Import File** section, click **Browse…** and import the following file:

   /home/spadmin/ImplementerTraining/config/LDAP/LDAP.xml

2. Once the file is done loading, check the output to confirm that everything loaded ok



Import from File Results

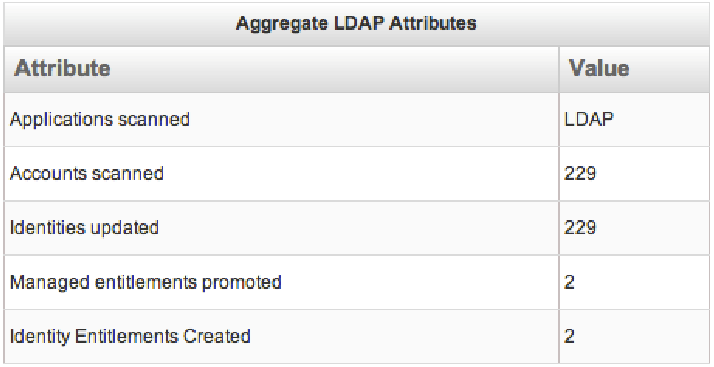
Import results

```
CorrelationConfig:LDAP Correlation
TaskDefinition:Aggregate LDAP
TaskDefinition:Aggregate LDAP Groups
Application:LDAP
```

3. This single XML file contained the LDAP application definition, Correlation Configuration, and Tasks to aggregate both LDAP Accounts and Groups.

4. Next, aggregate the LDAP application by running the following tasks in order and confirming the output:

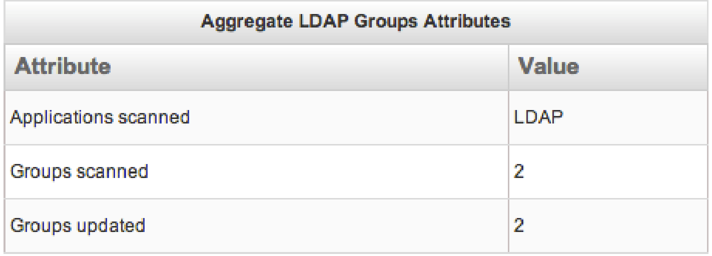
   a. **Aggregate LDAP**



Aggregate LDAP Attributes

Attribute	Value
Applications scanned	LDAP
Accounts scanned	229
Identities updated	229
Managed entitlements promoted	2
Identity Entitlements Created	2

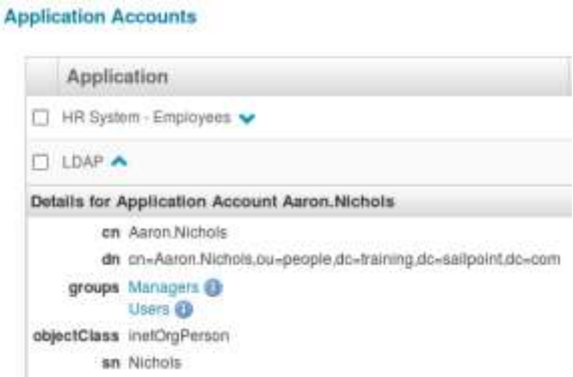
   b. **Aggregate LDAP Groups**



Aggregate LDAP Groups Attributes	
Attribute	**Value**
Applications scanned	LDAP
Groups scanned	2
Groups updated	2

5. Once the Aggregations are complete, check the following to make sure everything went smoothly

   a. Check **Aaron.Nichols** to make sure he has an account and the appropriate entitlements for LDAP

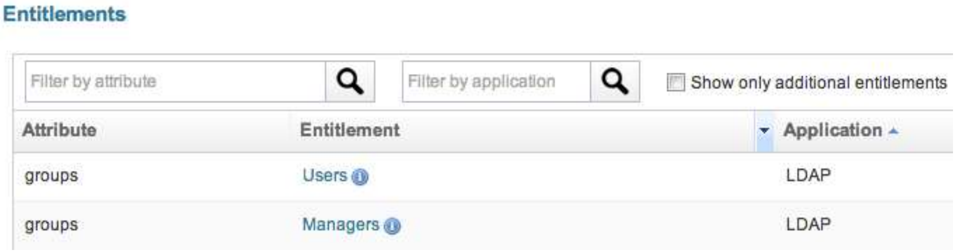


Application Accounts

	Application
□	HR System - Employees
□	LDAP

Details for Application Account Aaron.Nichols

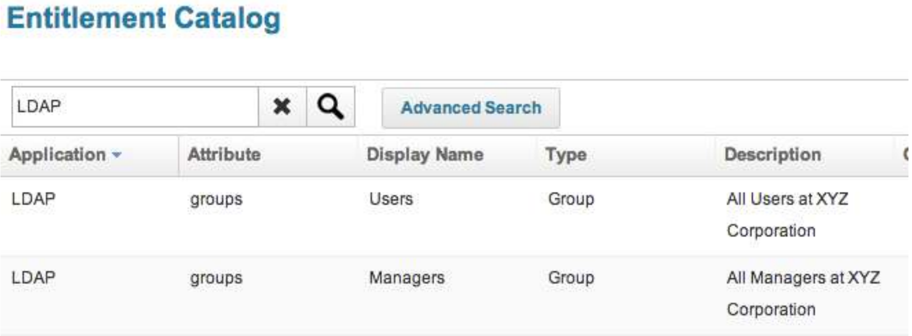
cn	Aaron.Nichols
dn	cn=Aaron.Nichols,ou=people,dc=training,dc=sailpoint,dc=com
groups	Managers Users
objectClass	inetOrgPerson
sn	Nichols



Entitlements

Attribute	Entitlement	Application
groups	Users	LDAP
groups	Managers	LDAP

   b. Check the **Entitlement Catalog** to make sure that the LDAP account groups were loaded properly along with descriptions.



Entitlement Catalog

Application	Attribute	Display Name	Type	Description
LDAP	groups	Users	Group	All Users at XYZ Corporation
LDAP	groups	Managers	Group	All Managers at XYZ Corporation

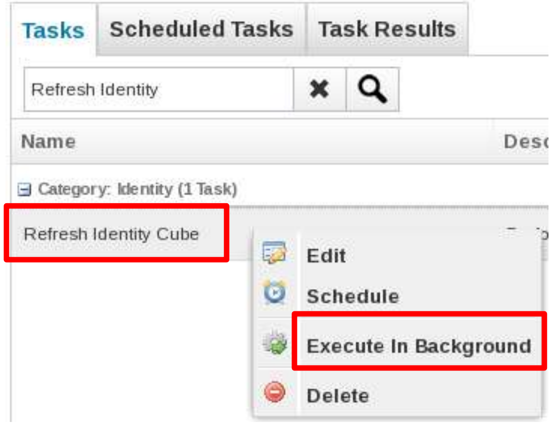
Refresh Identities

Once all aggregations are complete, an identity refresh is required to complete processing of the identity data. Though aggregations result in entitlement attributes appearing on the Identity Cube Application Accounts and Entitlements tabs, one more step (a refresh,) is required to fully promote entitlements and make them usable by other processes.

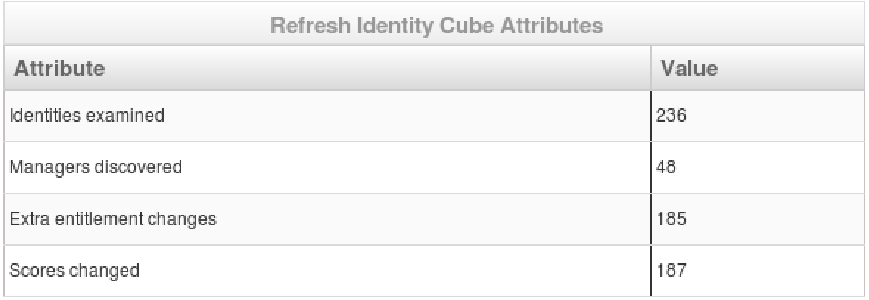
1. Run the task: **Refresh Identity Cube**.

   a. Navigate to **Setup → Tasks;** search for and right click the **Refresh Identity Cube** task.

   b. Select **Execute In Background**



   c. Confirm the results of the identity refresh



Refresh Identity Cube Attributes

Attribute	Value
Identities examined	236
Managers discovered	48
Extra entitlement changes	185
Scores changed	187

Exercise #9: Exploring the Identity Refresh Task

The Refresh Task is critical to finalizing data on the Identity Cubes. For example, all entitlements are promoted from the Account Data to the Identity Cubes by the Refresh Task. Another example, though we have yet to speak in detail about them: policy violations and risk scores are calculated by the Refresh Task.

Typically Aggregation Tasks are followed by a Refresh Task. You just performed a Refresh Task that refreshed *all of the identities* in the system (the default) and completed all configured promotions to the cube.

Investigate the Default Refresh Identity Cube Task

1. Navigate to **Setup → Tasks → Task Results** tab. View the **Refresh Identity Cube** *results*.

   a. How many identities were examined? ________________

   b. How many identities are there in your IdentityIQ instance? ________________
   Hint: Navigate to Identities → Identity Warehouse

   c. Notice that the default Refresh task ran against all identities in the environment.

2. Navigate to **Setup → Tasks** and view the **Refresh Identity Cube** task.

   a. View the default options. From the following list, draw lines through the options that are not one of the defaults (that are not checked):

      - Refresh identity attributes
      - Refresh the identity risk scorecards
      - Check active policies
      - Process events
      - Refresh assigned, detected roles and promote additional entitlements
      - Refresh manager status

   b. From the above list, circle the option that promotes entitlements from the Account data to the Identity Cube.

   c. Scroll up and look at the task options. List the two methods for constraining the identities that will be refreshed. (complete the phrases)

      Optional ____________________________ string to constrain the identities…

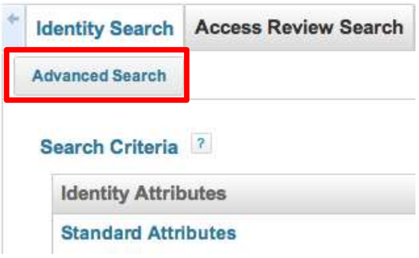
      Optional list of ____________________________ or ________________________________ …

Constrain the Refresh Identity Task

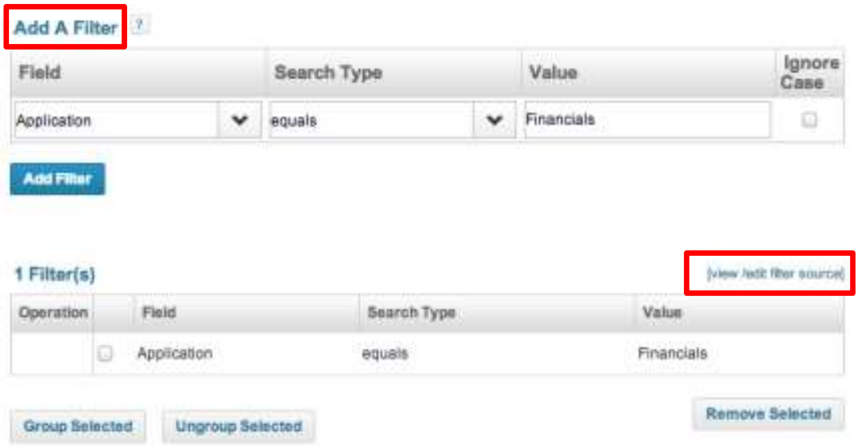
We will perform a Refresh on only identities who have an account on the Financials Application. We could configure a filter, a population, or a group to achieve our goal. For this exercise, we will use a filter string. (You will learn more about groups and populations in the next section.)

1. We will use Advanced Analytics to provide the filter syntax.

   a. Navigate to **Intelligence → Advanced Analytics**. On the Identity Search tab, click **Advanced Search**



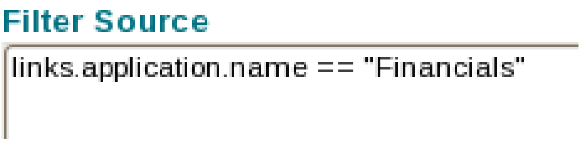
   b. Add a filter where **Application** is **equal** to **Financials**, and then click the link **view/edit filter source**.



Field	Search Type	Value	Ignore Case
Application	equals	Financials	□

Operation		Field	Search Type	Value
	□	Application	equals	Financials

   c. This is the filter syntax we will copy and paste into the filter constraint for the Refresh Identity Cube Task. **Copy the filter string**.

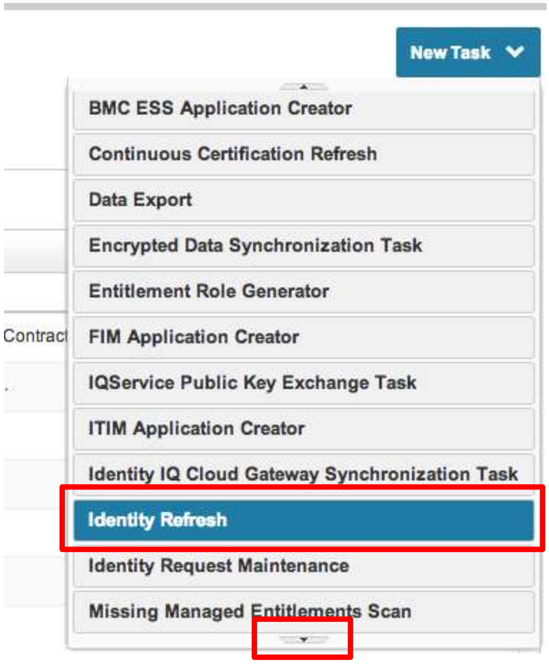


```
links.application.name == "Financials"
```

d. Notice the term, “links”. In IdentityIQ, *link* is synonymous with *account*. The term *link* is typically used internally to the product, and the term *account* is typically used in the user interface. In the back of this book, there is a list of common terms and their synonyms.

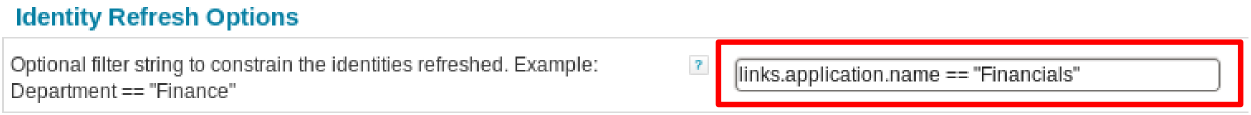
e. Run the search. How many identities were returned? ______________

2. Navigate to **Setup → Tasks** and under **New Task** use the arrow at the bottom to scroll. Choose **Identity Refresh** to create a new, blank Identity Refresh task.


New Task
BMC ESS Application Creator
Continuous Certification Refresh
Data Export
Encrypted Data Synchronization Task
Entitlement Role Generator
Contract
FIM Application Creator
IQService Public Key Exchange Task
ITIM Application Creator
Identity IQ Cloud Gateway Synchronization Task
Identity Refresh
Identity Request Maintenance
Missing Managed Entitlements Scan

3. Name your task **Refresh Financials Identities**

4. Paste the filter string into the input box titled **Optional filter string to constrain the identities refreshed**.



Identity Refresh Options

Optional filter string to constrain the identities refreshed. Example: Department == "Finance"

links.application.name == "Financials"

5. Select the default Identity Cube Refresh task options that you identified previously:

    a. Refresh identity attributes
    b. Refresh manager status
    c. Refresh assigned, detected roles and promote additional entitlements
    d. Refresh the identity risk scorecards
    e. Check active policies

6. Scroll to the bottom and click **Save and Execute**.

7. Navigate to **Setup → Tasks → Task Results** tab and confirm that the number of identities examined matches the number of identities from the search.

This concludes Section 1.

Extension Exercises (optional): Onboarding Logical and Multiplexed Applications

Objective

In this section we will load two additional types of applications that are used in special cases: Logical Applications and, Multiplexed Applications.

Overview

Sometimes, we need to identify an "application" and corresponding "accounts" as something that spans one or more applications. Take an example whereby a web application uses a specific entitlement in Active Directory to define membership in an application. Any user who has this specific Active Directory entitlement has an "account" on this application. This is an example of a Logical Application.

In our exercise, we are going to create a logical application to represent a web app that allows users to enter their time into the TRAKK system. This web application checks to make sure that a user has the right entitlement (capability = input) on the TRAKK application. Therefore, any user with that entitlement has an "account" on the TRAKK Web Application.

After the Logical application, we will load an example of a Multiplexed application. Sometimes, many applications are stored in the same target system or entitlement repository. In this case, the file is a single delimited file containing many applications that need to be loaded. For this type of application loading, the appropriate type is Multiplexed. The Multiplexed application is a special implementation of the Delimited File connector. It contains a BuildMap (or Customization) rule that controls how accounts are distributed across multiple applications as they are read in.

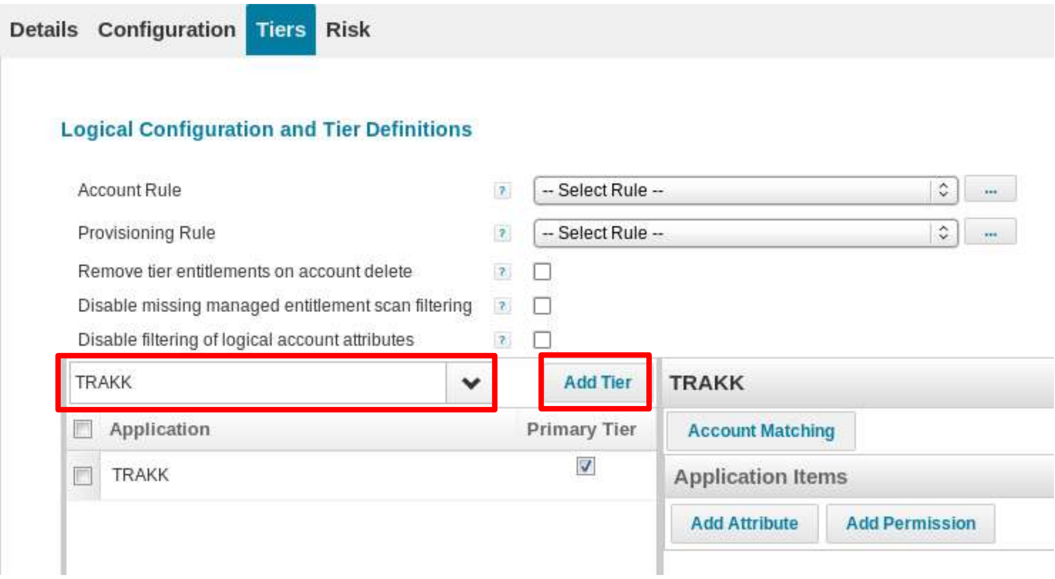
Define a Logical Application

Onboarding the Logical application is an optional exercise. If you have Logical application use cases, it is highly recommended that you perform this exercise. No further exercises in this training depend upon completion of onboarding this application.

1. Create a new application using the information from the table:

Application Applications → Application Definition → Add New Application	
Name:	**Logical Application – TRAKK**
Owner:	**The Administrator**
Application Type (Connector):	**Logical**
Description:	**A logical application that defines web-based access to the TRAKK Time Sheet application.**

2. Navigate to **Configuration**, and in the **Enter an application** dropdown, choose **TRAKK** and add it by selecting **Add Tier**. Select/confirm that **Primary Tier** is selected.



Account Rule ? -- Select Rule --

Provisioning Rule ? -- Select Rule --

Remove tier entitlements on account delete ? □

Disable missing managed entitlement scan filtering ? □

Disable filtering of logical account attributes ? □

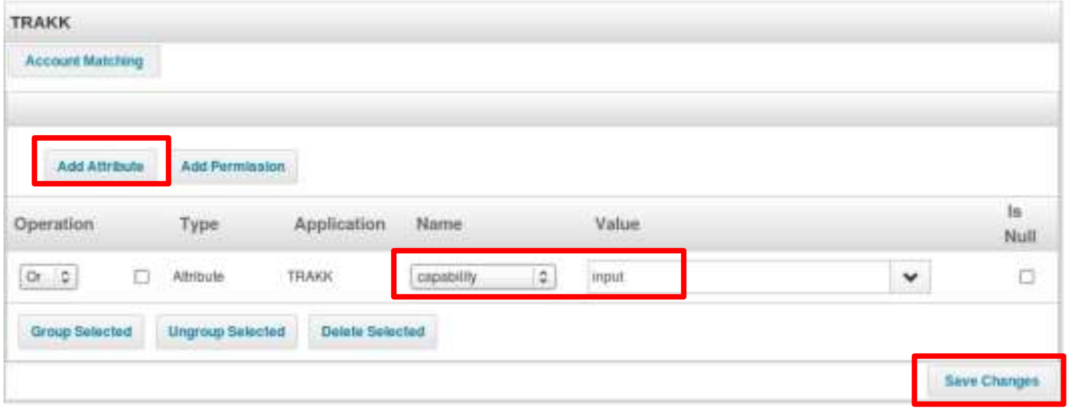
TRAKK

□ Application	Primary Tier
□ TRAKK	☑

3. On the right, click **Add Attribute**. Here is where we will define what attribute (or attributes) will need to be present for us to create an account for the logical application.

   a. Name: **capability**

   b. Value: **input**



Operation		Type	Application	Name	Value	Is Null
Or	□	Attribute	TRAKK	capability	input	□

   c. Click **Save Changes**

Configure the Schema for the Logical Application

1. Navigate to **Configuration → Schema** and configure the

    a. Native Object Type: **account**

    b. Identity Attribute: **id**

    c. Display Attribute: **username**

2. Click **Discover Schema Attributes** to fill in the schema attributes from the TRAKK application

3. Scroll down and **Save** the application

Aggregate Logical Accounts

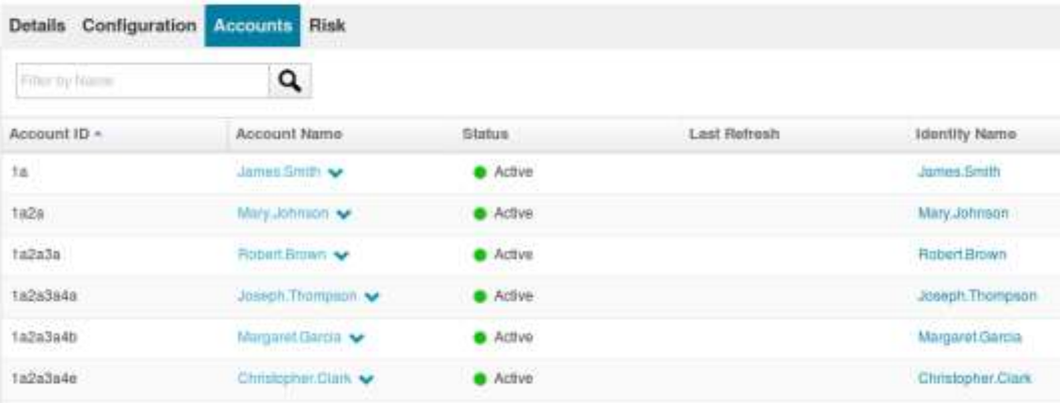
1. Using the following parameters, create a task to aggregate accounts from the logical application:

Task Setup → Tasks → New Task	
Type:	**Account Aggregation**
Name:	**Aggregate Logical Application**
Select applications to scan:	**Logical Application – TRAKK**
Detect deleted accounts:	**Checked**

2. Scroll down and select **Save and Execute** and click **OK.** Check the **Task Results** to see that the task ran successfully.

    a. Navigate to **Applications → Application Definition**, select **Logical Application – TRAKK** and choose the **Accounts** tab

    b. Confirm that there are 156 accounts for the TRAKK Logical Application



Account ID ▴	Account Name	Status	Last Refresh	Identity Name
1a	James.Smith	● Active		James.Smith
1a2a	Mary.Johnson	● Active		Mary.Johnson
1a2a3a	Robert.Brown	● Active		Robert.Brown
1a2a3a4a	Joseph.Thompson	● Active		Joseph.Thompson
1a2a3a4b	Margaret.Garcia	● Active		Margaret.Garcia
1a2a3a4e	Christopher.Clark	● Active		Christopher.Clark

Configure a Multiplexed Application

Onboarding the Multiplexed application is an optional exercise. If you have a Multiplex use case, it is highly recommended that you perform this exercise. No further exercises in this training depend upon completion of onboarding this application.

1. Create a new application using the information from the following table

Application Applications → Application Definition → Add New Application	
Name:	**Multiplex-App Base**
Owner:	**The Administrator**
Application Type (Connector:)	**Delimited File**
Connector Attributes: Configuration → Settings	
File Path	**/home/spadmin/ImplementerTraining/data/MultiPlexApp.csv**
Delimiter	**,**
File has column header on first line	**Checked**

Configure the Schema and Build Map rule for the Multiplex Application

1. Complete schema header:

Schema Configuration→Schema→account		
Name	**Value**	**Description**
Native Object Type	account	
Identity Attribute	employeeId	The Identity Attribute defines which attribute will be used to determine the uniqueness of the account.
Display Attribute	fullName	A more “friendly” identifier for the account used in the GUI.

2. Click **Discover Schema Attributes** to load up the 7 attributes found in the delimited file.

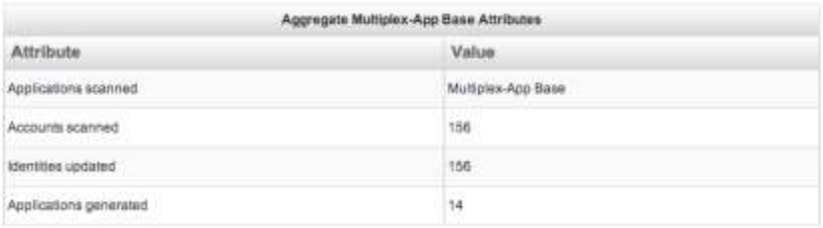
3. Navigate to **Rules** and add a Build Map rule as shown:
    a. Name: **Build Map Rule - Multiplexed App**
    b. Copy and paste the rule from **/home/spadmin/ImplementerTraining/beanshell/BuildMapRule-MultiPlexRule.txt**
    c. Click **Save** to save the rule
4. Make sure to select the **Build Map Rule - Multiplexed App** from the drop down selection after creating the rule.
5. Navigate to **Correlation**.
6. Create an Account Correlation
    a. Name: **Multiplex Correlation**
    b. Map **employeeId** to **Employee ID**
    c. **Add** the attribute to the correlation configuration and **Save**
7. Click **Save** to save the application

Run an Aggregation Task to load the Multiplexed Application

1. We will now create a task to aggregate accounts from the Multiplexed application:

Task Setup → Tasks → New Task	
Type:	**Account Aggregation**
Name:	**Aggregate Multiplex-App Base**
Description:	**Aggregation Task for Multiplexed application**
Select applications to scan:	**Multiplex-App Base**
Detect deleted accounts:	**Checked**
Disable optimization of unchanged accounts:	**Checked**

2. Scroll down and select **Save and Execute** and click **OK.** Check the **Task Results** tab and confirm that your identities are updated as shown:

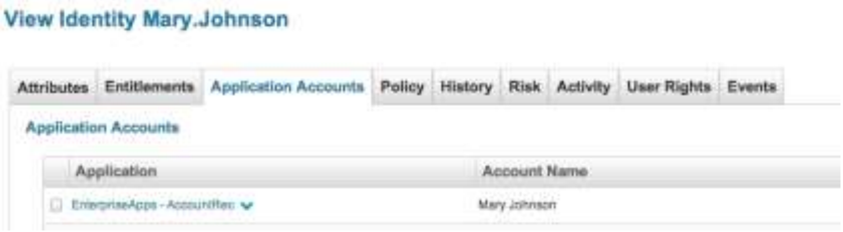


Aggregate Multiplex-App Base Attributes

Attribute	Value
Applications scanned	Multiplex-App Base
Accounts scanned	156
Identities updated	156
Applications generated	14

3. Verify that the multiplex application loaded okay.

   d. Check **Mary.Johnson** and see if she has an account on an EnterpriseApps application:

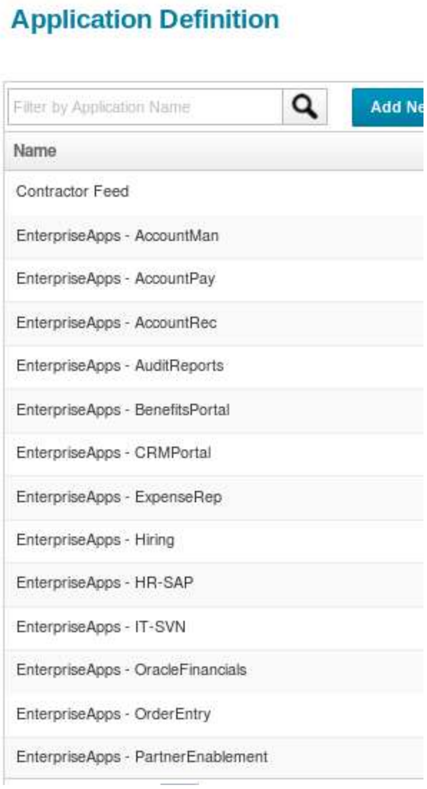


View Identity Mary.Johnson

Application Accounts

	Application	Account Name
□	EnterpriseApps - AccountRec	Mary Johnson

   e. Check to see if all the Applications were created properly. There should be 14 total applications that start with **EnterpriseApps.**

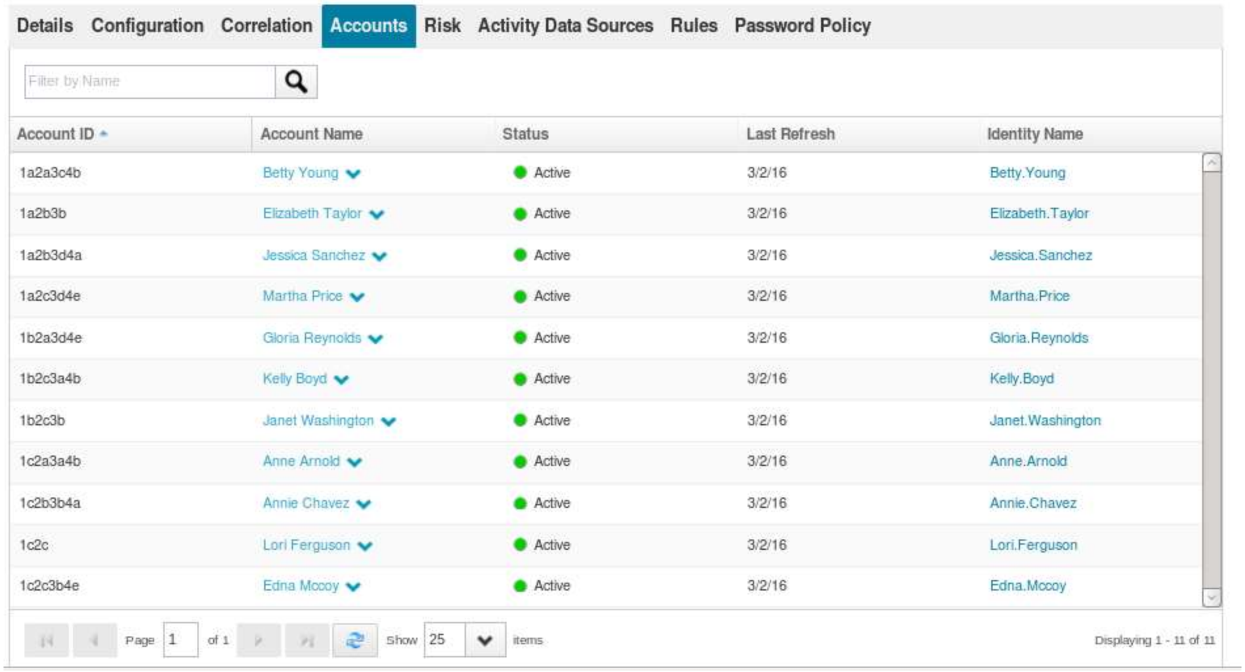


Application Definition

Name
Contractor Feed
EnterpriseApps - AccountMan
EnterpriseApps - AccountPay
EnterpriseApps - AccountRec
EnterpriseApps - AuditReports
EnterpriseApps - BenefitsPortal
EnterpriseApps - CRMPortal
EnterpriseApps - ExpenseRep
EnterpriseApps - Hiring
EnterpriseApps - HR-SAP
EnterpriseApps - IT-SVN
EnterpriseApps - OracleFinancials
EnterpriseApps - OrderEntry
EnterpriseApps - PartnerEnablement

4. Select one of the EnterpriseApps and confirm that accounts have been loaded.

EnterpriseApps - AccountMan:



Account ID	Account Name	Status	Last Refresh	Identity Name
1a2a3c4b	Betty Young	● Active	3/2/16	Betty.Young
1a2b3b	Elizabeth Taylor	● Active	3/2/16	Elizabeth.Taylor
1a2b3d4a	Jessica Sanchez	● Active	3/2/16	Jessica.Sanchez
1a2c3d4e	Martha Price	● Active	3/2/16	Martha.Price
1b2a3d4e	Gloria Reynolds	● Active	3/2/16	Gloria.Reynolds
1b2c3a4b	Kelly Boyd	● Active	3/2/16	Kelly.Boyd
1b2c3b	Janet Washington	● Active	3/2/16	Janet.Washington
1c2a3a4b	Anne Arnold	● Active	3/2/16	Anne.Arnold
1c2b3b4a	Annie Chavez	● Active	3/2/16	Annie.Chavez
1c2c	Lori Ferguson	● Active	3/2/16	Lori.Ferguson
1c2c3b4e	Edna Mccoy	● Active	3/2/16	Edna.Mccoy

5. If the applications loaded okay, then your aggregation of the Multiplexed application was successful.