SailPoint

IdentityIQ – L01

Date

–

Contents

Exercise 1

Populating Identity Cubes – Loading Authoritative Data

Use Case ID:	L01 – E01		
Use Case Name:	Populating Identity Cubes		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	Admin, IIQ System		
Description:	Creating identity cubes from authoritative applications		
Preconditions:	IIQ System is Up and Running, Authoritative application		
Post conditions:	Successful onboarding of application and formation of identity cube		
Normal Flow:	1. Integrate the authoritative application 2. Create respective aggregation tasks 3. Identity mappings 4. Identity Cube formation		
Exceptions:	NA		
Dependent Usecase:	NA		
Assumptions:	NA		
Notes and Issues:	NA		

Overview:

In this usecase, we are going to setup the following:

- Onboarding authoritative applications to IdentityIQ.
- Understanding delimiter connector
- Creating account aggregation tasks for the respective applications
- Populating identity cubes in IdentityIQ

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 will exist 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:

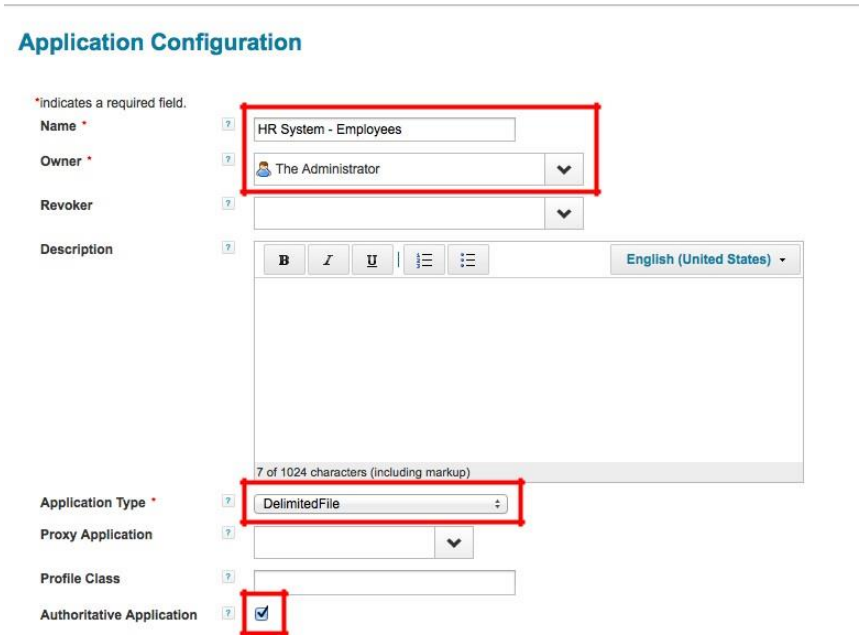
- They want an additional Identity attribute called: **status**. This attribute will be set to "**Employee**" or "**Contractor**" depending on which authoritative source is used to create the identity. This attribute needs to be searchable and a group factory (so that we can easily create groupings of 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:

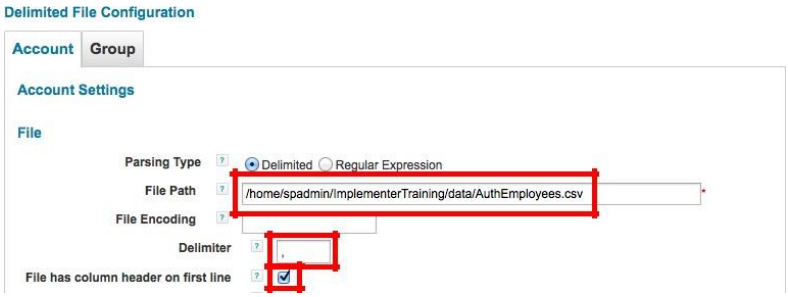
- Create an additional Identity Attribute called "Status" that will be sourced using rules that will determine if an Identity is an Employee or Contractor.
- Use a Creation Rule to set a default IdentityIQ password for each user as we create the Identity Cubes.

Define Employee and Contractor Applications

1. Create a new Application definition for the Employee Data
   a. Login to IdentityIQ as **spadmin/admin**
   b. Navigate to **Applications -> Application Defination** 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**



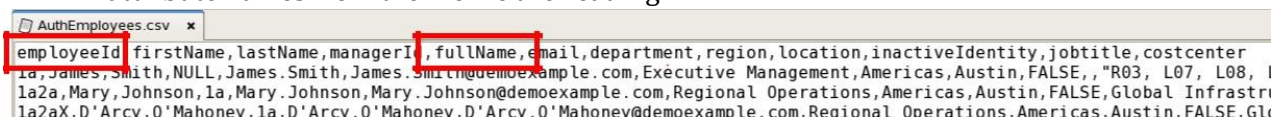
d. Under the **Configuration** tab, select **File** and configure as follows:
   i. File Path: **C:\Training\data\AuthEmployees.csv**
   ii. Delimiter: **,**
   iii. File has column header on first line: **Checked**



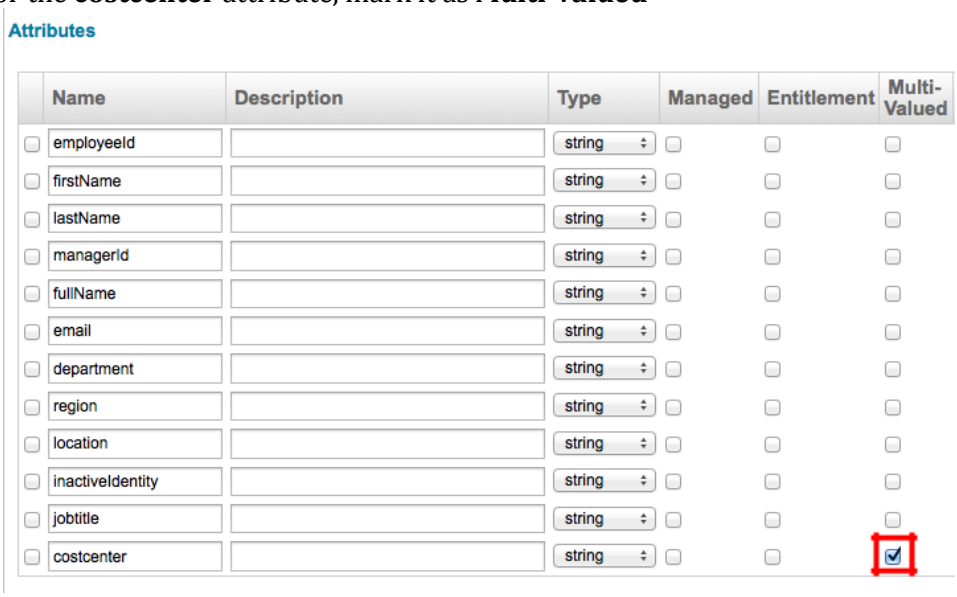
e. Click **Save** to save the application.

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. Navigate to **Define ❼ Applications** and select the Application definition you just created: **HR System – Employees**
   a. Scroll down and select the **Schema** tab
   b. Choose **Add Account Schema** and define 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 in 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.



   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, mark it as **Multi-valued**

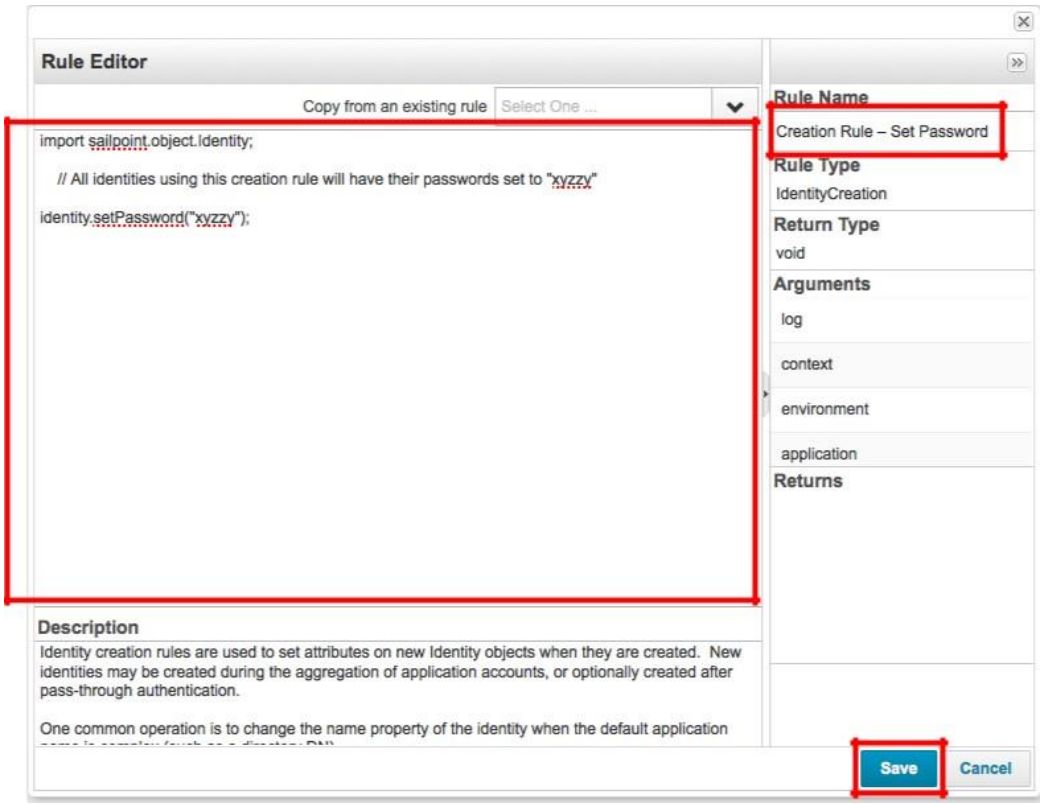


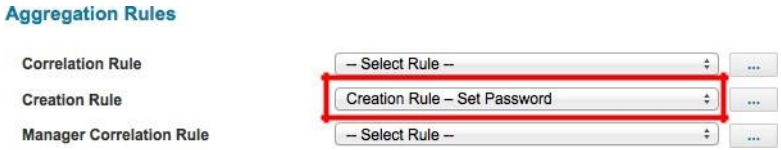
f. Click the **Preview Accounts** button. Preview Accounts iterates over the first 10 accounts and displays the results in a popup instead of loading it into IdentityIQ.

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. Select the **Rules** tab
b. To the right of the **Creation Rule**, select the button with the ellipsis (…)
c. You will now see the **Rule Editor**
d. Set the **Rule Name** to **Creation Rule – Set Password** as shown and then select **Save**





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

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 Defination 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. Under the **Attributes** tab, select **Account** and configure as follows:

        i. File Path:

        /home/spadmin/ImplementerTraining/data/AuthContractors.csv

        ii. Delimiter: **,**

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

    d. Click **Save** to save this application definition.

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.

    e. **Navigate to Applications -> Application Defination and select** you just created:

    Contractor Feed

    f. Scroll down and select the **Schema** tab

    g. Choose **Add Account Schema** and define as follows:

        i. Native Object Type: **account**

        ii. Identity Attribute: **employeeId**

        iii. Display Attribute: **fullName**

h. 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,locati

on,inactiveIdentity,costcenter

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

6. Click the **Preview Accounts** button. The first 10 records from the file will be displayed. Remember that **Preview Accounts** 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. Select the **Rules** tab

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

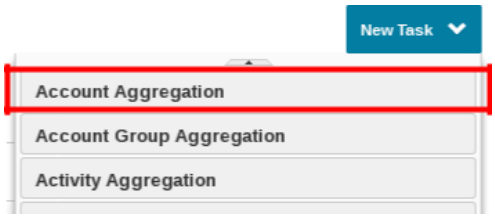
Note: We are using the same rule as we did for the previous application. It is common in IdentityIQ implementations to re-use rules. In this case, the rule applies to loading both Employees and Contractors.

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** ❼ **Tasks** and under **New task...** choose **Account Aggregation**



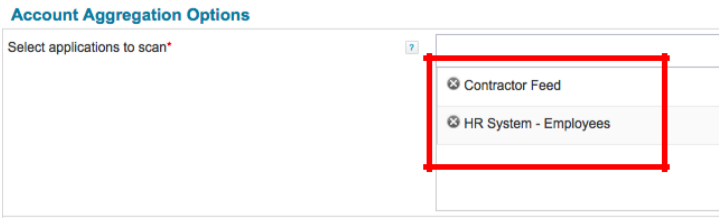
b. Define the Task as follows:

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

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

   iii. Select applications to scan:

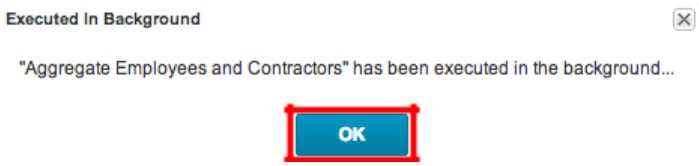
   HR System – Employees
   Contractor Feed



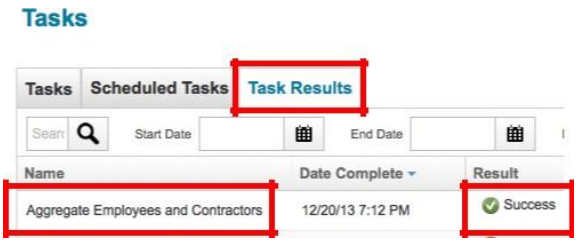
   iv. Select the **Detect Deleted Accounts** checkbox

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

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



d. Once you are back on the main **Tasks** page, select **Task Results** as shown below. Once the Task is finished, there will be a result for **Aggregate Employees and Contractors.** Click this entry to see the results of the Aggregation.



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

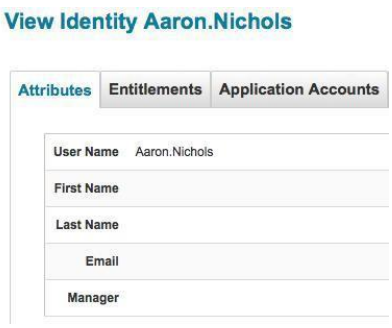
2. Confirm that the aggregation was successful.

   a. View an Identity to confirm that the aggregation was successful

      i. Navigate to **Identities** ❼

      Identities Warehouse

      ii. 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



   b. Confirm that the Creation Rule was successful

      i. **Logout** of IdentityIQ

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

      iii. **Logout** of IdentityIQ

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

      v. If you cannot login 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. **Logout** 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. Login 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? _____

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

      _____

   b. Select the **Application Account**s tab and view Aaron's account details for the **HR System - Employees** application

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

      ii. How many cost centers are associated with the account? _____

      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 **Define ➔ Applications**, 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 to number iii of 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*:

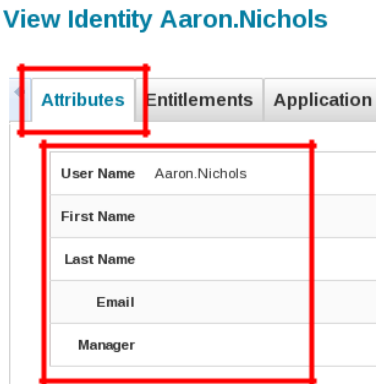
Attributes

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

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.

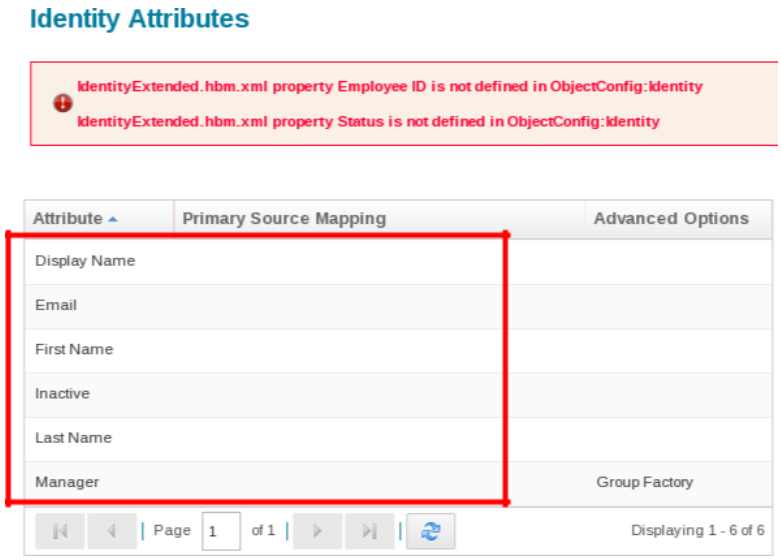
Previously we saw that so far only one Identity Attribute has been populated – User Name. User Name is populated by default from the Display Attribute in the schema header.



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 information 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.

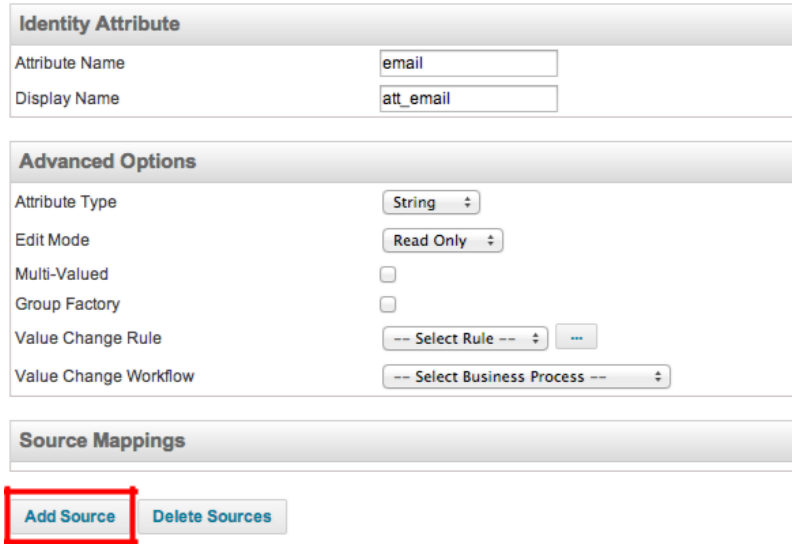
1. Navigate to **Global Setting ❼ Identity Mappings**

2. This is the main interface for configuring Identity Attributes and how they are populated. 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. Notice also that the standard attributes are created by default with no source mapping.



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

   a. Click **Add Source** to configure the source of this Identity Attribute

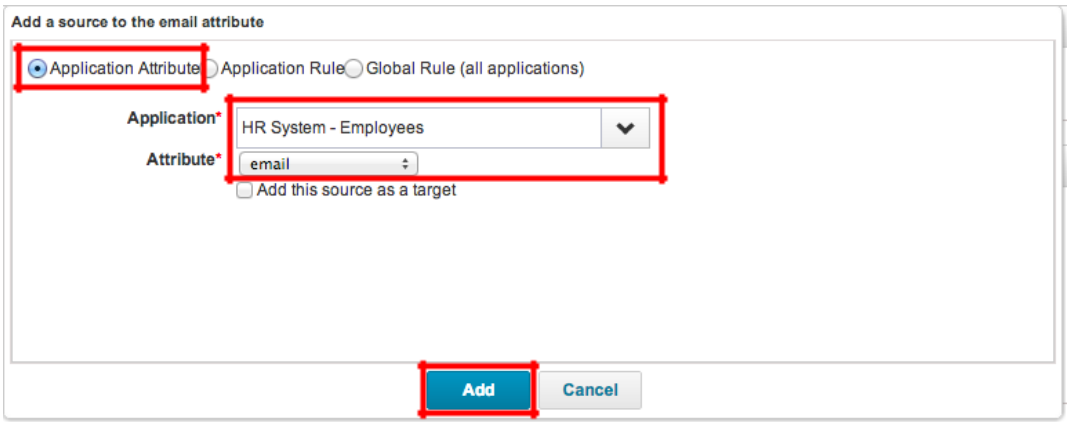


   b. Choose **Application Attribute**

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

   d. Attribute: **email**

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



   Note: Attributes can be populated by Application Attributes, or by a Rule (Application or Global.) Also, they can 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.

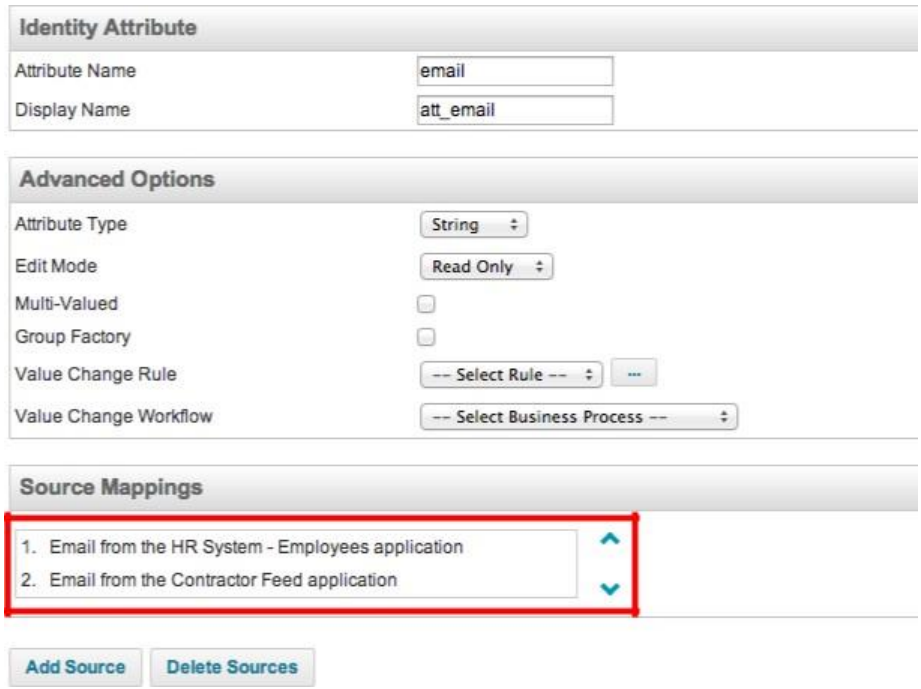
   g. Choose **Application Attribute**

h. Application: **Contractor Feed**

i. Attribute: **email**

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

k. Your attribute mapping should look like this when you are done. Note that we have 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.

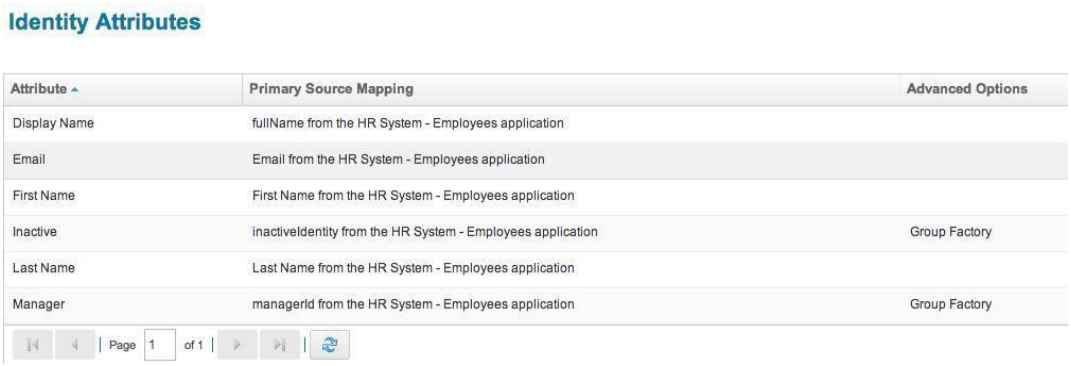


l. Click **Save** to complete the changes to the **email** attribute

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

Attribute	**Primary Source Mapping**	**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 default Identity attributes, it should look like this:



Identity Attributes

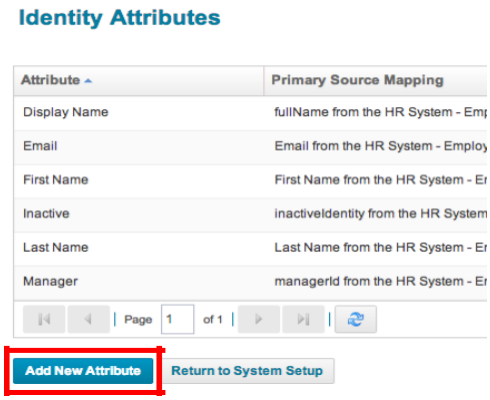
Attribute	Primary Source Mapping	Advanced Options
Display Name	fullName from the HR System - Employees application	
Email	Email from the HR System - Employees application	
First Name	First Name from the HR System - Employees application	
Inactive	inactiveIdentity from the HR System - Employees application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

Page 1 of 1

Define Extended Identity Attributes

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

1. Click the **Add New Attribute** button on the **Identity Attributes** page



Identity Attributes

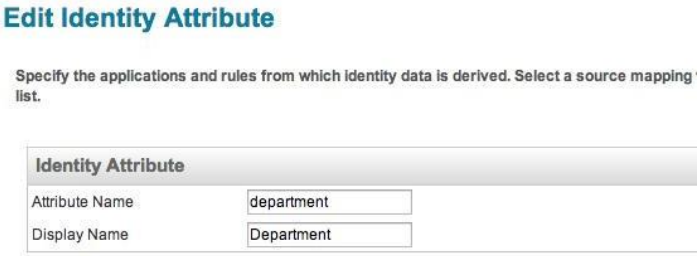
Attribute	Primary Source Mapping
Display Name	fullName from the HR System - Emp
Email	Email from the HR System - Employ
First Name	First Name from the HR System - Er
Inactive	inactiveIdentity from the HR System
Last Name	Last Name from the HR System - Er
Manager	managerId from the HR System - Er

Page 1 of 1

Add New Attribute | Return to System Setup

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 Mapping**, select **Add Source**

i. Choose **Application Attribute**

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

iii. Attribute: **department**

iv. Click **Add**

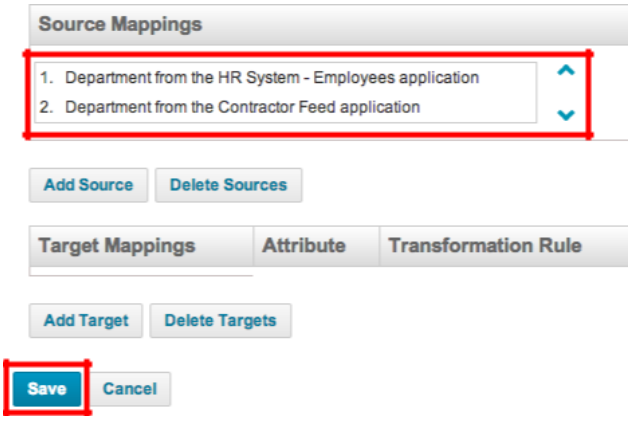
d. Click **Add Source**

i. Choose **Application Attribute**

ii. Application: **Contractor Feed**

iii. Attribute: **department**

iv. Click **Add**



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	Primary Source Mapping	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: Multi-valued attributes and all standard attributes are automatically searchable in IdentityIQ. They are not shown as searchable in the summary list because they do not count against your configured set of searchable attributes.

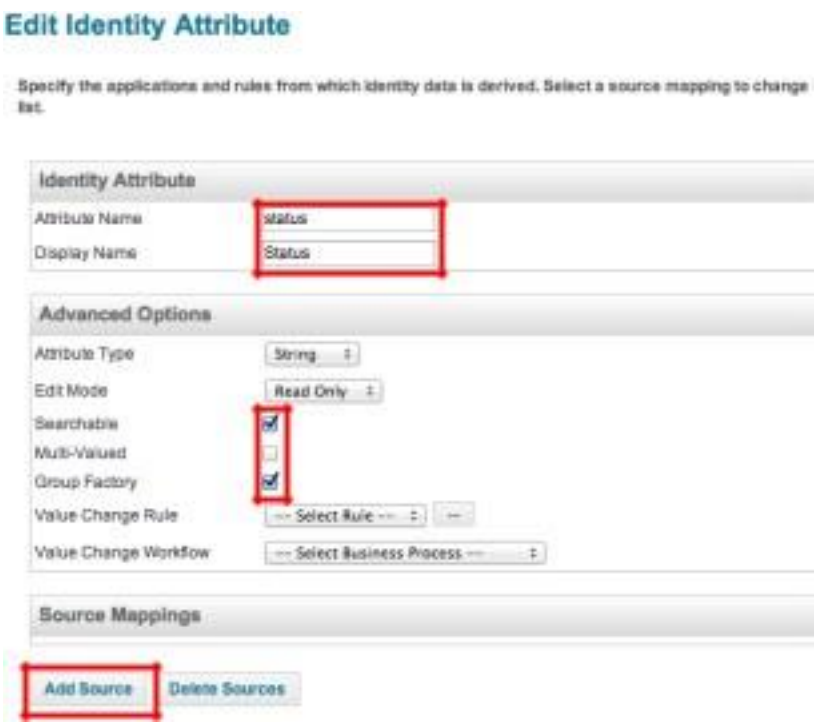
3. Next, we will add an identity attribute that will be derived with a rule. a. Select **Add New Attribute** and configure the attribute as defined:

   i. Attribute Name: **status**

   ii. Display Name: **Status**

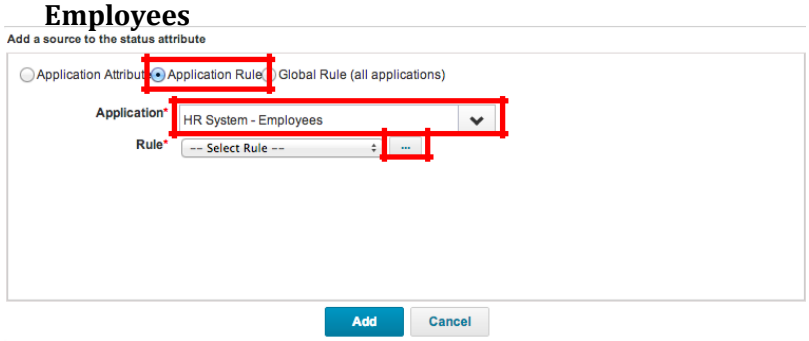
   iii. Searchable: **checked**

   iv. Group Factory: **checked**



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

      1. Configure the Source Mapping as shown
         a. Application Rule: **Checked**
         b. Application: **HR System – Employees**



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 Rule Script: **return "Employee";**

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

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

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

1. Configure the Source Mapping as shown

a. Application Rule: **Checked**

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

1. After adding these 7 additional Identity Attributes, your Identity Attributes screen should look similar to this:

Identity Attributes

Attribute	Primary Source Mapping	Advanced Options
Cost Centre	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.

Update Manager Status

The data we are reading from the delimited files includes manager data. In order to get this manager data to properly be 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 **Application ❼ Applications Defination ❼ HR System - Employees** and go to the **Correlation** tab

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

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

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

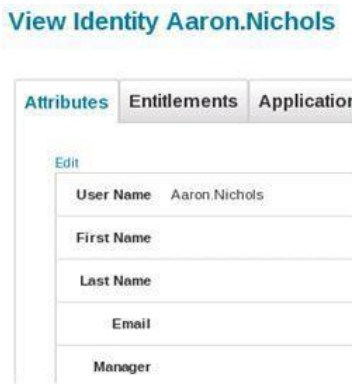
   a. Navigate to **Application ❼ Applications Defination ❼ Contractor Feed** and go to the **Correlation** tab

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.

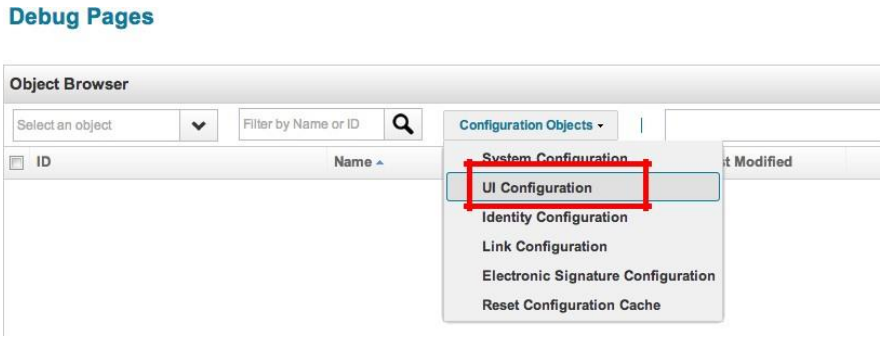


We want to extend this to show our new identity attributes, like Status, Job Code, Cost Centers, etc.

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 shortcut in you Firefox browser

   b. The IdentityIQ Debug Pages are used for advanced configuration and for debugging. Click **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 make sure that you 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 **Define ❼ Identities**, 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 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.

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

2. Refresh Identities and observe changes to the Identities

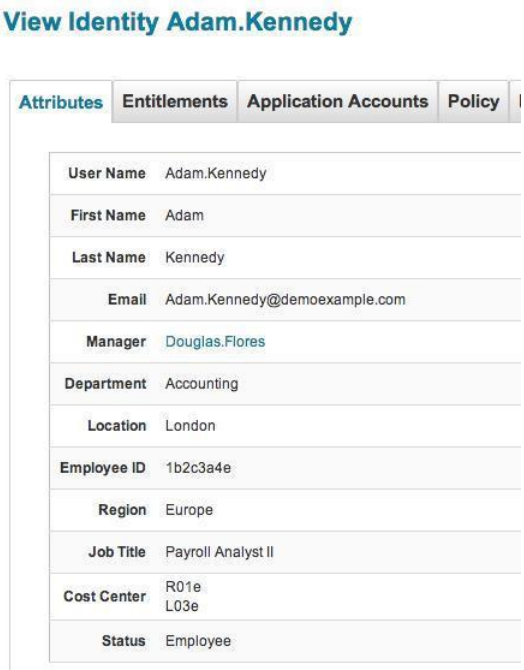
   a. Navigate to **Setup ➔ Tasks** and scroll down looking for the **Refresh Identity Cube** task

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

   c. Wait until the task is complete (Visit **Task Results** and refresh to see when it's complete.)

   d. When the refresh is complete, Navigate to **Identities ➔ Identitiy 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.



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.

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 Consle by using either method:

(1) **./iiq console -j** (from within the /WEB-INF/bin directory)

(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 identified as 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 cleaned everything out, you can always re-execute the aggregation and identity refresh tasks to reload the identity cubes.

Exercise 2

Loading and Correlating the Financials Application

Use Case ID:	L01 – E02		
Use Case Name:	**Loading and Correlating the Financials Application (Non-Auth)**		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	Admin, IIQ System		
Description:	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 as well as account data		
Preconditions:	IIQ System is Up and Running		
Post conditions:	Successful onboarding of financials application		
Normal Flow:	1. Integrate the financials application 2. Correlation Configuration 3. Create respective aggregation tasks (account and group)		
Exceptions:	NA		
Dependent Usecase:	NA		
Assumptions:	NA		
Notes and Issues:	NA		

Overview:

In this usecase, we are going to setup the following:

- Onboarding non-authoritative application to IdentityIQ.
- Creating account and group aggregation tasks for the respective application
- Correlation configuration

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 configuring 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,userName
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 Name: ❼ Applications ❼ Add New Application	
Owner:	**The Administrator**
Description:	**Finance Application**
Application Type (Connector:)	**Delimited File**
Connector Attributes: Attributes Tab	
File Path	**C:\Training\data\Finance- users.csv**
Delimiter	**,**
File has column header on first line	**Checked**
Data needs to be Merged	**Checked**
Index Column	**dbId**
Data sorted by the indexColumn(s)?	**Checked**
Which Columns should be merged?	**groupmbr**

    b. Scroll down and click **Save**

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

    a. Navigate to **Application ❼ Applications Defination** and choose the **Financials** application

    b. Scroll down and choose the **Schema** tab

    c. Choose **Add Account Schema** and then **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, 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.

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

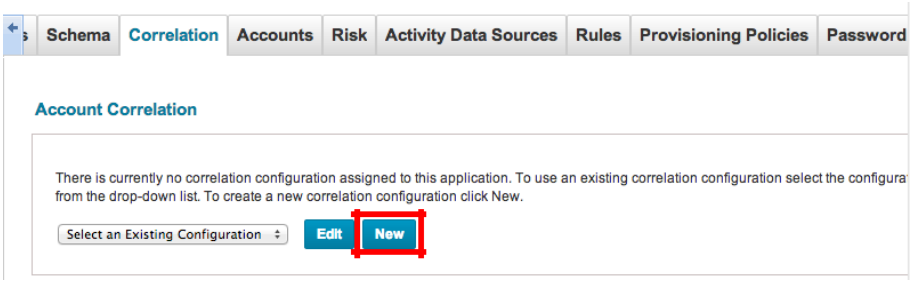
e. Scroll down and click **Save**

1. Define an Account Correlation configuration tomatch accounts from this application to existing Identity Cubes

   a. Navigate to **Application ❼ Applications Defination** and choose the **Financials** application

   b. Scroll down and choose the **Correlation** tab

      i. Click the **New** button



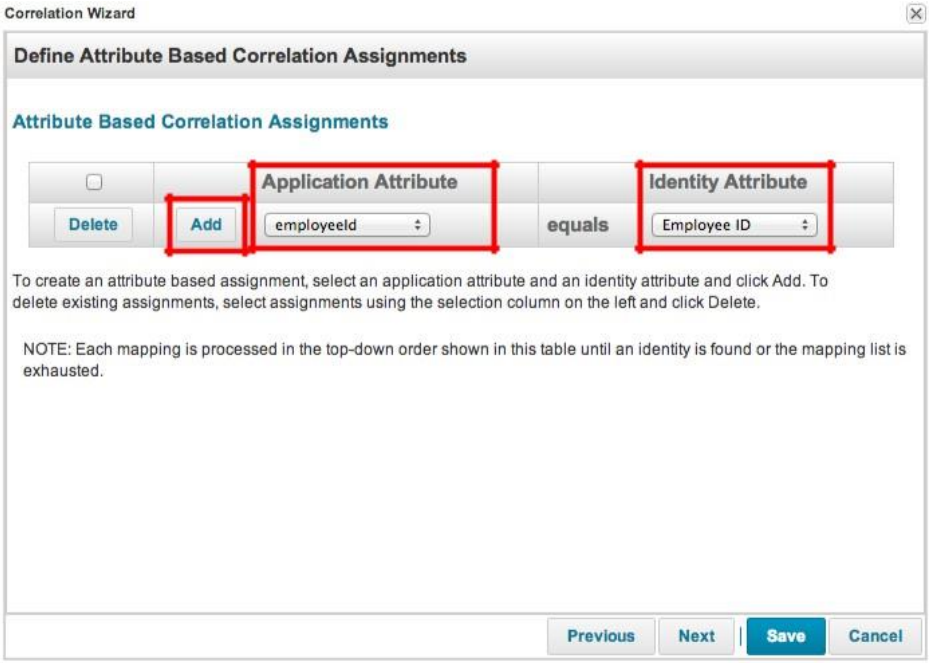
      ii. This will bring up the **Correlation Wizard**

      iii. Click **Next**

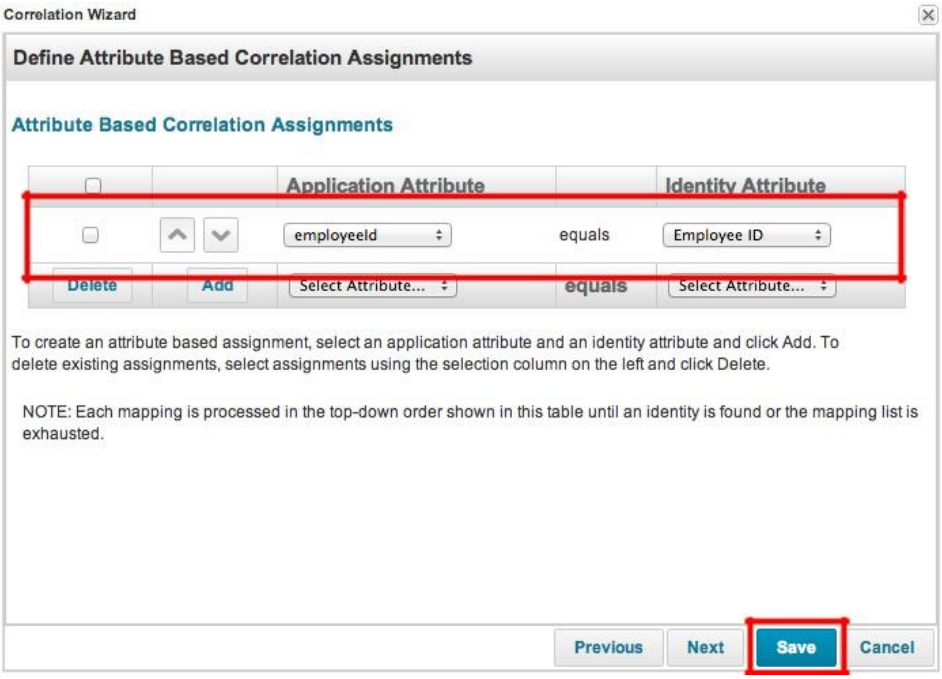
      iv. When prompted, enter a name for this configuration: **Financial Correlation**

      v. Click **Next** twice

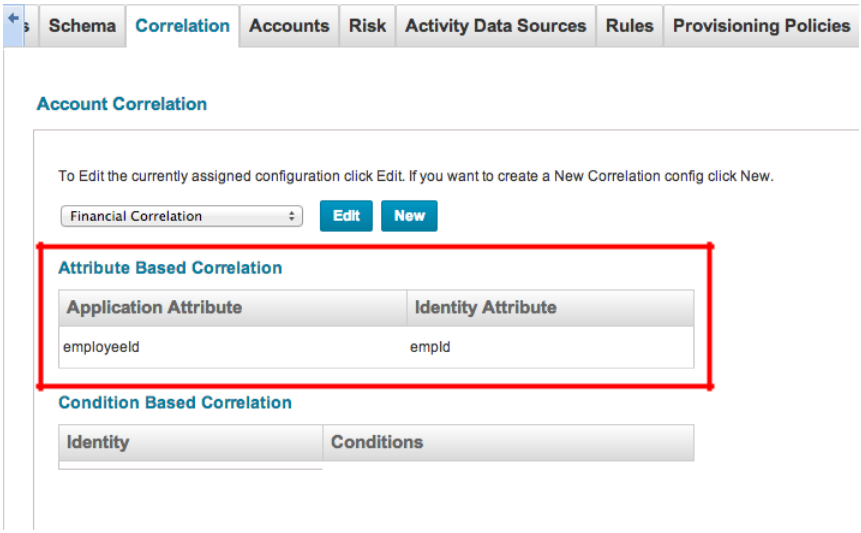
vi. When prompted, configure the **Attribute Based Correlation Assignment** as shown, then click **Add**



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



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



ix. 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** ❼ Tasks ❼ 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**

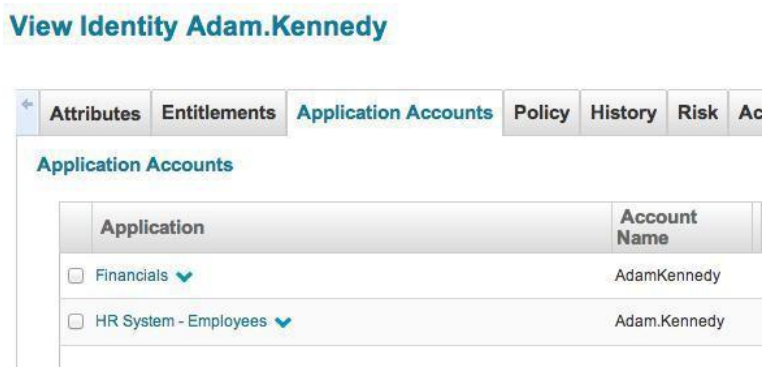
3. Go to **Task Results** and confirm that the results are shown:

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 ➐ Identity Warehouse** and find **Adam.Kennedy**

2. Click **Application Accounts** and check to make sure that the Financials account shows up:



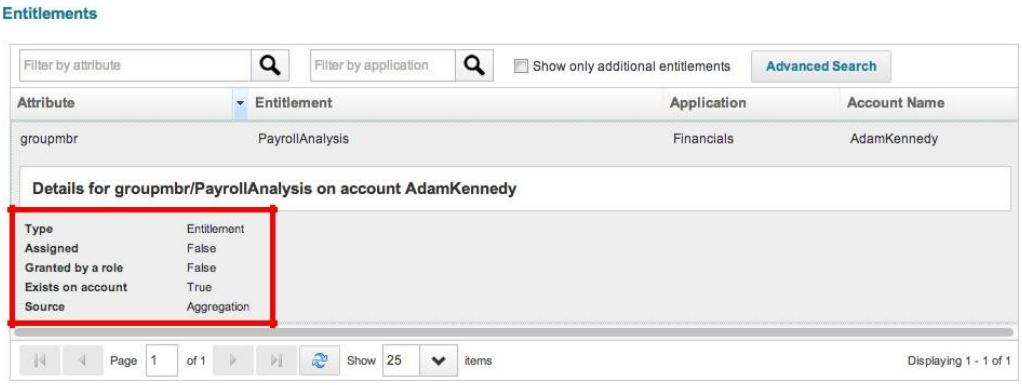
3. Click the **Financials** account and check the attributes for the **Financials** application:

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

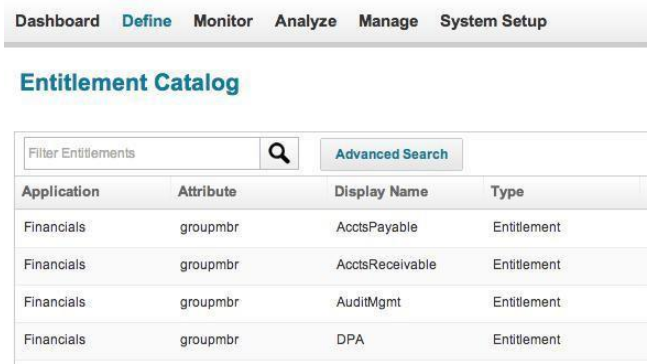


   The **Source** is aggregation and **Assigned** is false. This means this is a detected entitlement that we discovered from the IT environment via **Aggregation.**

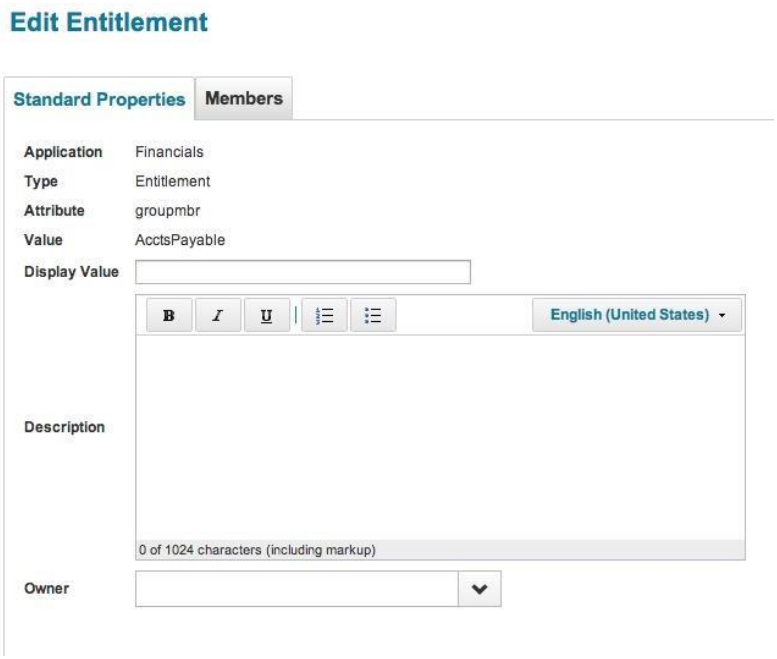
   c. Complete the following table. When you defined the Financials application, where did you configure IdentityIQ to track this entitlements information?

Which Application: **Which Tab:**	*Financials*
Which Attribute:	
Option Selected:	

5. Navigate to **Application ➔ 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.



6. In the entitlement catalog, we can assign owners to the entitlements (for approval purposes) and set multi-lingual descriptions. This data is useful during Certifications and for use with Lifecycle Manager as well. When LifeCycle Manager is installed, we can mark items as requestable (used by Lifecycle Manager to determine what can be requested).
   a. Select the **AcctsPayable** entitlement and view the options on the **Standard Properties** tab. You will not see the **Requestable** field because you have not yet installed LifeCycle Manager. Note that **Requestable** is the default.
   b. Select the **Members** tab to view the identities with the Financials AcctsPayable entitlement.



Exercise 3

Loading and Correlating the PAM Application

Use Case ID:	L01 – E03		
Use Case Name:	**Onboarding PAM Application (Non-Auth)**		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	Admin, IIQ System		
Description:	The objective of this exercise is to load an application that includes a more complicated definition including Accounts, Account Groups and Permissions		
Preconditions:	IIQ System is Up and Running		
Post conditions:	Successful onboarding of PAM application		
Normal Flow:	1. Integrate the PAM application 2. Correlation Configuration 3. Create respective aggregation tasks (account and group)		
Exceptions:	NA		
Dependent Usecase:	NA		
Assumptions:	NA		
Notes and Issues:	NA		

Overview:

In this usecase, we are going to setup the following:

- Onboarding non-authoritative PAM to IdentityIQ.
- Creating account and group aggregation tasks for the respective application
- Correlation configuration

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 resource. 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 and rights that 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","DR System:YY-Function Control:NN-BackupControl:YY"
"TEST01","20080211","FINANCE","DR System:YY-Function Control:NN-BackupControl:YY"
```

These lines define the specific permissions for both the **ACCOUNTING** and **FINANCE** permission groups. The permission data is stored as a single concatenated value, so we will use code (specifically a Build Map rule) to parse this data into individual rights and targets to store as permissions.

Create the Base PAM Application

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

Application ❼ Applications ❼	
Owner:	**Patrick.Jenkins**
Revoker:	**Albert.Woods**
Description:	**Financially significant application**
Application Type (Connector:)	**Delimited File**
Connector Attributes: Attributes Tab	
File Path	**C:\Training\data\PAM- users.csv**
Delimiter	**,**
File has column header on first line	**Checked**
Data needs to be Merged	**Checked**
Index Column	**User ID**
Data sorted by the indexColumn(s)?	**Checked**
Which Columns should be merged?	**Permission Group**

   a. Click **Save** to save your work for the PAM Application

Configure Account Schema for PAM Application

1. Navigate to **Application ❼ Applications Defination** and choose the newly created **PAM** application
2. Click the **Schema** tab to define the account schema for the **PAM** application
3. Click **Add Account Schema**

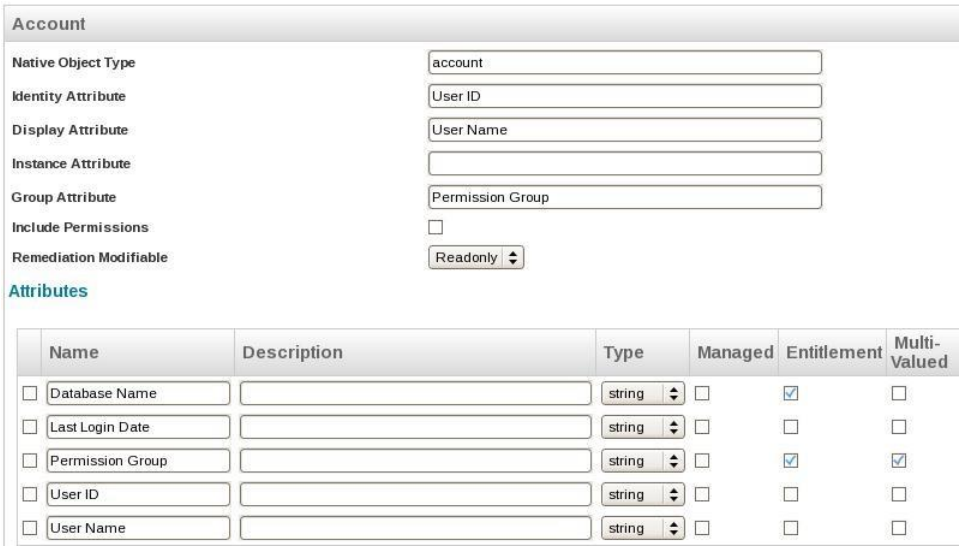
4. Fill in the configuration details:

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.

5. 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.

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

6. Once finished, your account schema should look like this:



a. Preview the accounts.

b. Scroll to the bottom of the page and select **Save**.

7. On the **Correlation** tab, under **Account Correlation,** configure a new **Account Correlation** based on the following attributes:

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

8. Click **Save** once you've configured your account correlation**.** When done, your correlation should look like this.

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

10. Open up the IIQ console by one of the following two methods: a.
    Go to the **IIQ Console** shortcut on the desktop
11. Within the IIQ Console, run the following command:

```
>connectorDebug PAM iterate
```

12. The **connectorDebug** command iterates over all of the accounts and displays the results of the iteration to the console screen instead of loading it into IdentityIQ. Like the Preview Accounts 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. The connectorDebug command displays an XML format of the resource object.

a. In the screen shot, circle the multi-value 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 Accounts (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 Accounts 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 what attributes we will read in from the PAM-group-permissions.csv file.

1. Navigate to **Application ➐ Applications Defination** and choose the **PAM** application

2. Under the **Configuration** tab, select the **Group** tab and configure the connector as

   shown:

Name	Value	Description
	C:\Training\data	
	PAM-group-permissions.csv	containing the groups
Delimiter	,	
File has column header on first line	Checked	

3. Scroll down and select **Save**

Configure Group Schema for PAM Application

1. Navigate to **Application ➔ Applications Defination** and choose the **PAM** application

2. Under the **Schema** tab, scroll down and select **Add Group Schema**

3. 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 account. You could think of this as the primary key for the application accounts. In this case, we are using the “Permission Group” which is unique for each user.
Display Attribute	Permission Group	In this case, the Permission Group name is acceptable for the Display Attribute
Include Permissions	checked	We will be loading permissions as part of the data loading exercise. This tells the connector to expect them and to include them in the Entitlement Catalog.

4. 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.

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

5. Select the **Preview Groups** 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.

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.

```
spadmin@training:~/tomcat/webapps/identityiq/WEB-INF/bin
File  Edit  View  Terminal  Tabs  Help
    </Map>
  </Attributes>
</ResourceObject>

<ResourceObject displayName="ADMINISTRATORS" identity="ADMINISTRATORS" ob
jectType="group">
  <Attributes>
    <Map>
      <entry key="Description" value="Administrators Group for PAM"/>
      <entry key="Permission Group" value="ADMINISTRATORS"/>
      <entry key="Permission Rights">
        <value>
          <List>
            <String>DR System:YY-Function Control:YY-BackupControl:YY</St
ring>
          </List>
        </value>
      </entry>
    </Map>
  </Attributes>
</ResourceObject>

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

Use a Build Map Rule to transform Permission Rights

If we look at the output from the connectorDebug command, we will see that the Permission Rights are being read as a single value similar to this:

```
"DR System:YY-Function Control:NN-BackupControl:YY"
```

This string represents a series of Targets and Rights.

The Targets are "DR System","Function Control" and "BackupControl". The Rights are either "NN" or "YY" where "YY" stands for Create and Update and "NN" stands for Execute

If we want to model this as individual permissions within IdentityIQ, we will need to break these Permission Rights up into something more like this:

- Right="Update" target="DR System"
- Right="Create" target="DR System"
- Right="Execute" target="Function Control"
- Right="Update" target="BackupControl"
- Right="Create" target="BackupControl"

Breaking up the rights and targets into individual entitlements will allow a certifier to make decisions for each Right and Target combination

Go to Global Setting -> Import and import the rule from

C:\Training\admin\backup\BuildMapRule- PAM. Now Nevigate Application ->

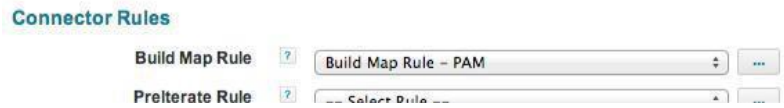
Application Defination -> PAM on the **Rule** tab, scroll down to the **Connector Rules**, and

chose the Build Map Rule and Save

Scroll down to the bottom of the rule. Look at the documentation to answer the following question.

a. What does the ArrayList object returned by the rule represent?

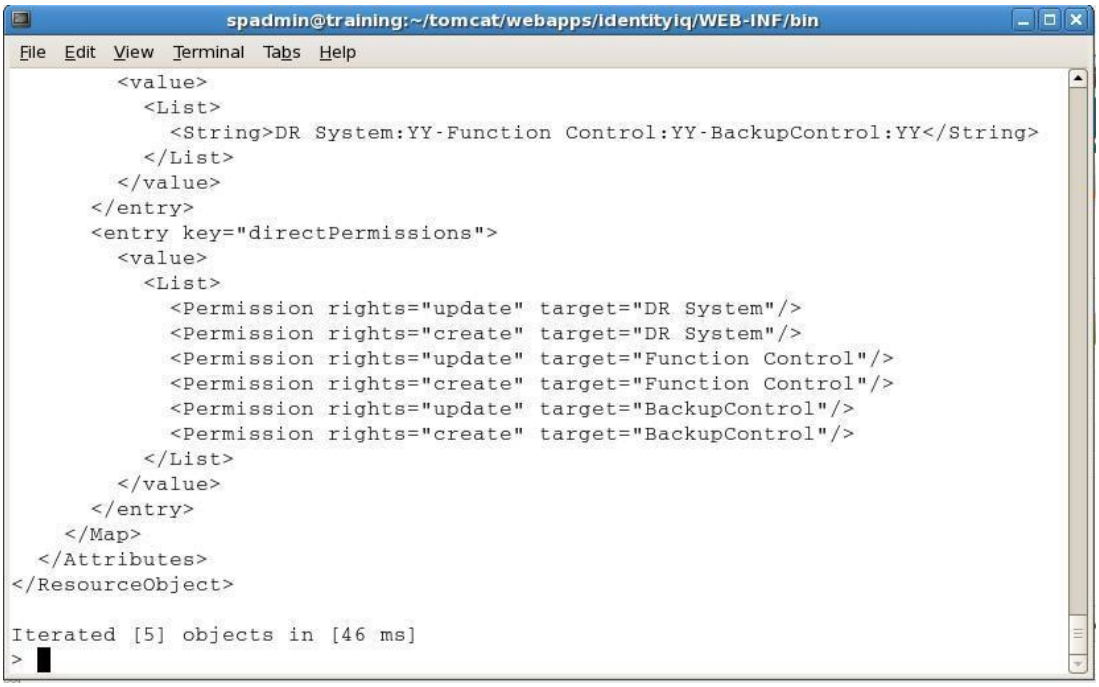
_____



Scroll down and **Save** the application

8. Go to the IIQ Console and run:

```
>clearCache
>connectorDebug PAM iterate group
```



10. You should see that your Build Map rule is now parsing out the Permission Rights from the file into individually certifiable Permissions. When we run a certification later on this data, we will see how this is presented to the user.

11. 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. We will now create a task to aggregate accounts from the PAM application:

Task

Tasks New task...

Task	
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	
❼ Tasks ❼ New task…	
Task	
Name:	**Aggregate PAM Account Groups**
Description:	**Task to aggregate account groups from the PAM application.**
Select applications to scan:	**PAM**
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 ❼ Identitiy Warehouse** and look up the user: **Carl.Foster**

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

   c. Navigate to **Application ❼ 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

Filter Entitlements | Advanced Search

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	

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

Exercise 4
Onboarding JDBC Applications

Use Case ID:	L01 – E04		
Use Case Name:	Onboarding JDBC Applications		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	Admin, IIQ System		
Description:	The objective of this exercise is to onboard account data out of multiple JDBC resources		
Preconditions:	IIQ System is Up and Running, JDBC source		
Post conditions:	Successful onboarding of JDBC application		
Normal Flow:	1. Integrate the JDBC application 2. Correlation Configuration 3. Create respective aggregation tasks (account and group)		
Exceptions:	NA		
Dependent Usecase:	NA		
Assumptions:	NA		
Notes and Issues:	NA		

Overview:

In this usecase, we are going to setup the following:

- Onboarding JDBC application to IdentityIQ.
- Creating account and group aggregation tasks for the respective application
- Correlation configuration

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. Note that you will need to save the application prior to configuring the merging:

Application **Application** ❼ Applications **Defination** ❼ Add New Application	
Name:	**TRAKK**
Owner:	**The Administrator**
Description:	**The TRAKK Time Tracking Application**
Application Type (Connector:)	**JDBC**
Connector Attributes: Attributes Tab	
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)';**

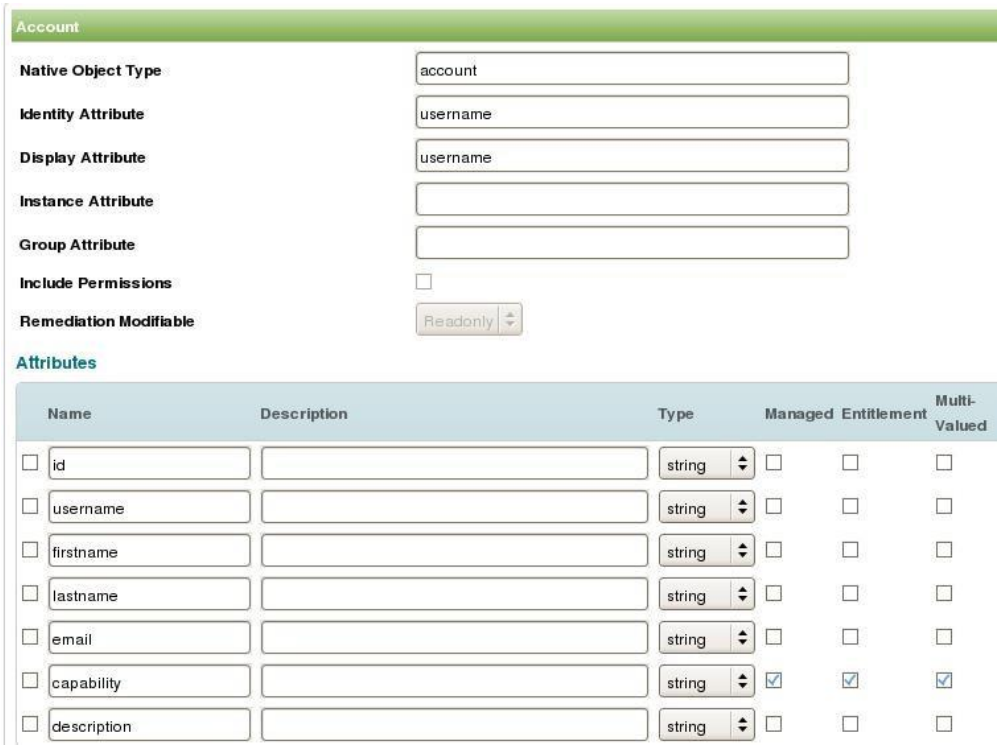
2. Click **Save** to save the application

Configure Account Schema for the TRAKK Application

1. Navigate to **Define ❼ Applications** and select **TRAKK**

2. Under the **Schema** tab, configure the account schema by clicking **Add Account Schema** and then 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**			



3. Select **Preview Accounts**.

4. Go back to the **Attributes** tab and complete the merge options.
   Connector Attributes:

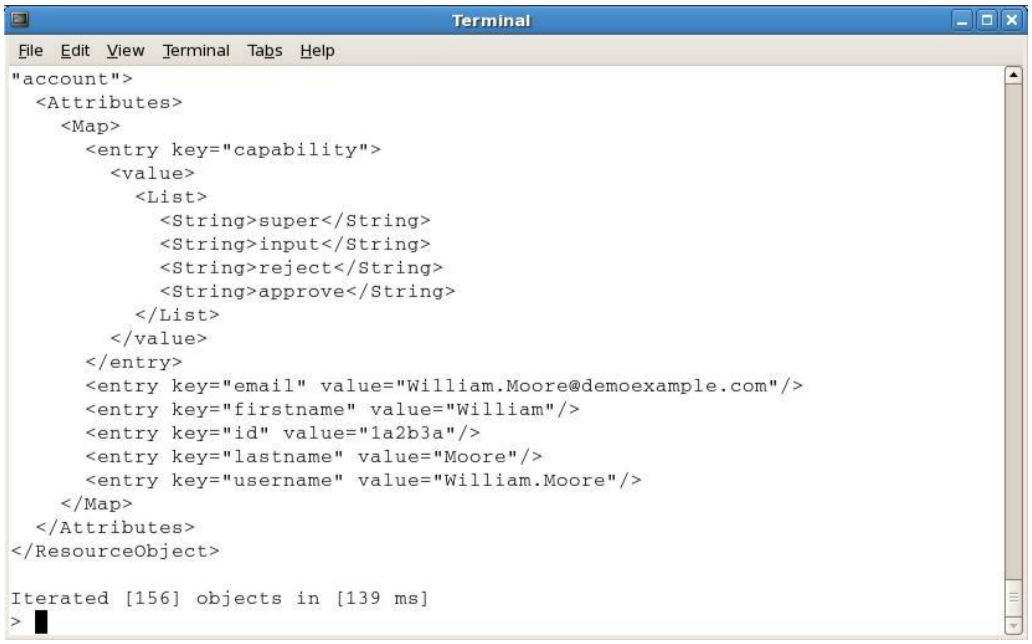
Attributes Tab	
Advance Settings, Enable Advance Option	**checked**
Data needs to be merged	**checked**
Index Column	**username**
Which Columns should be merged?	**capability**

5. Scroll down and click **Test Connection**. This option is useful to verify the connection between IdentityIQ and the application without performing an account preview.

6. If the connection is successful, click **Save** to save your work for the TRAKK Application.

7. 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



>connectorDebug TRAKK get account Adam.Kennedy

```
IIQ Console
File  Edit  View  Terminal  Tabs  Help
> 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>

>
```

8. The first connectorDebug command grabbed all the accounts out of the JDBC resource using the **SQL Statement** query that you configured. The second connectorDebug command grabbed a single account from the JDBC resource using the **getObjectSQL** query. Some connectors can support the random access of a single record. For the JDBC connector, the **getObjectSQL** query supports this random access operation.

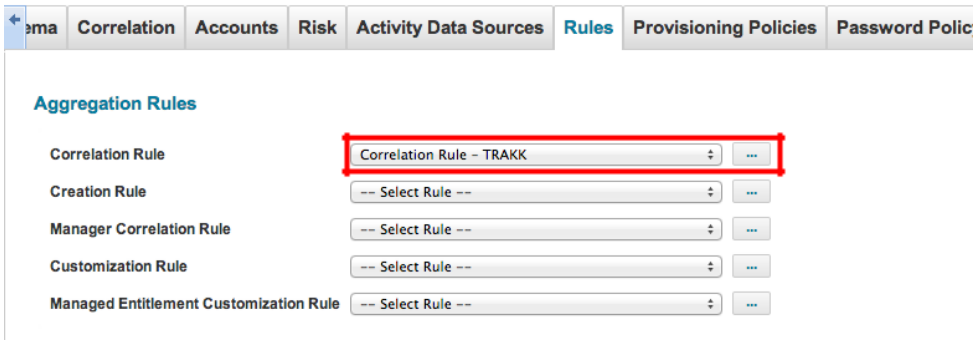
9. If you run both of the connectorDebug commands successfully, continue, otherwise re-check your configuration of the TRAKK application.

Configure Correlation Rule for the TRAKK Application

1. Configure Correlation for this new application

   a. Within the TRAKK application, go to the **Rules** tab and create a new Correlation Rule by clicking the **...** :
      - **ii. import the rule Correlation-Rule-Trakk from C:\Training\admin\backup and goto Rule tab and attach a new Rule in Corelation Rule**
      - **iii.** Click **Save**

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



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

Aggregate Accounts from TRAKK

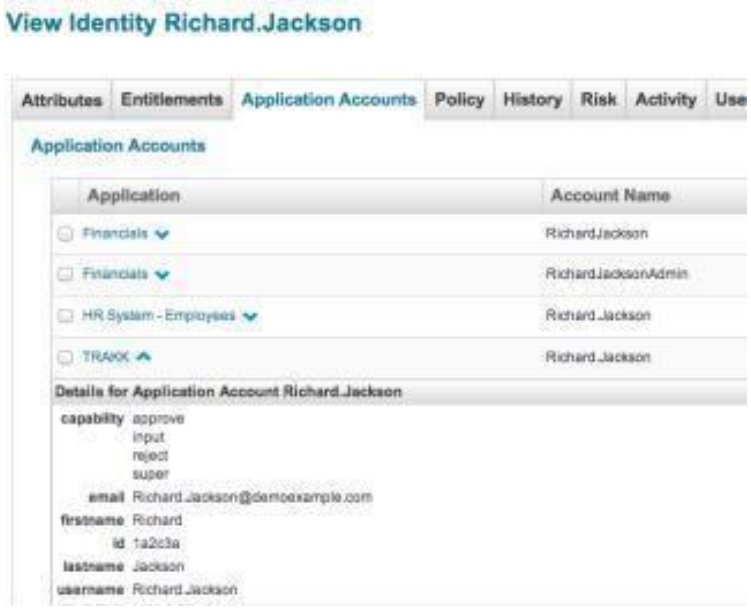
1. Configure a new task using the information from the table below

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**
Promote managed attributes:	**Checked**
Detect deleted accounts:	**Checked**
Disable optimization of unchanged accounts:	**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:



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

Entitlements

Filter by attribute | Filter by application | Show only additional entitlements | Advanced Search

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. Click **Application ❼ Entitlement Catalog** to confirm that the managed entitlements for the **TRAKK** application were loaded:

Entitlement Catalog

Filter Entitlements | Advanced Search

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 Setting ❼ Import from File** and where it says **Import Objects**, click **Browse…** and import the following file:

   C:\Training\config\PRISM\PRISM.xml

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

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 **Define ❼ Applications** 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 ➐ 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.**

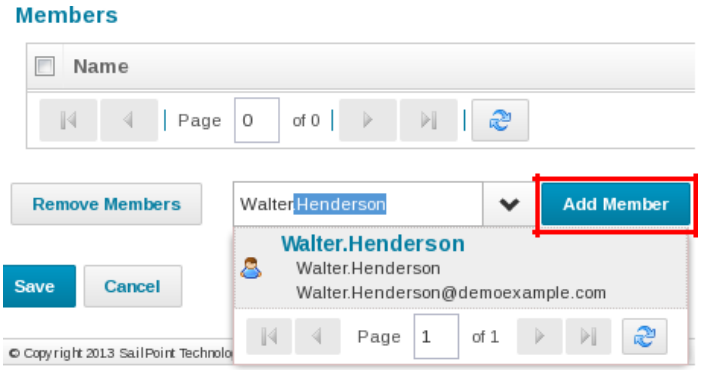
      iv. Group Email**: prism_owners@example.com**

      v. Rights: **Application Administrator**

   c. Add members to the workgroup:

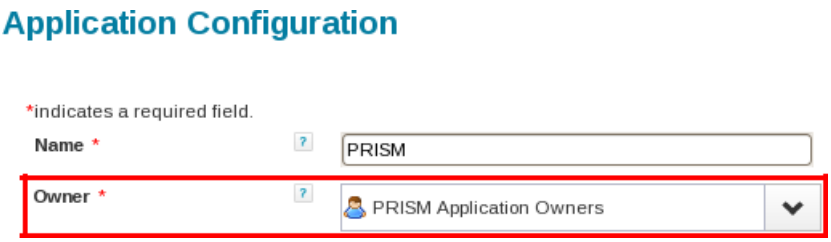
      i. Search for **Walter Henderson** and click **Add Member**

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



   d. **Save** the workgroup.

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



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

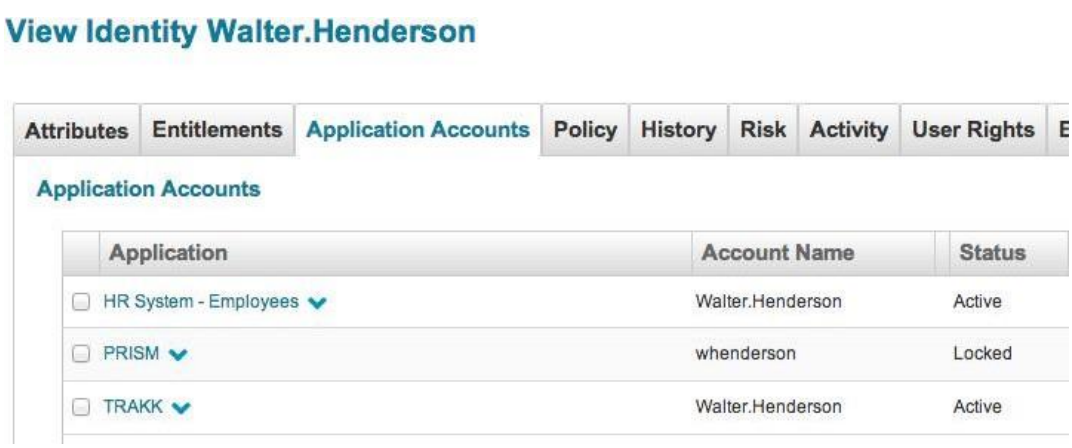
Aggregate PRISM Attributes	
Attribute	**Value**
Applications scanned	PRISM
Accounts scanned	2
Identities created	1
Identities updated	1
Extra entitlement changes	2
Managed entitlements promoted	3
Identity Entitlements Created	6

b. Aggregate PRISM Groups

Aggregate PRISM Groups Attributes	
Attribute	**Value**
Applications scanned	PRISM
Groups scanned	3
Groups updated	3

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.



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

Entitlements

Filter by attribute | Filter by application

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 5

Onboarding an LDAP Application

Use Case ID:	L01 – E05		
Use Case Name:	**Onboarding LDAP Applications**		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	Admin, IIQ System		
Description:	The objective of this exercise is to onboard account and group data out of an LDAP application		
Preconditions:	IIQ System is Up and Running, LDAP source		
Post conditions:	Successful onboarding of LDAP application		
Normal Flow:	4. Integrate the LDAP application 5. Correlation Configuration 6. Create respective aggregation tasks (account and group)		
Exceptions:	NA		
Dependent Usecase:	NA		
Assumptions:	NA		
Notes and Issues:	NA		

Overview:

In this usecase, we are going to setup the following:

- Onboarding LDAP application to IdentityIQ.
- Creating account and group aggregation tasks for the respective application
- Correlation configuration
- For this application, we will onboard an LDAP application using the LDAP direct connector.

Start the local LDAP Server

1. Double click the **OpenDS LDAP Control Panel** shortcut on the desktop of the VM
2. Click **OK** when the LDAP client starts up.
3. Under Server Status, click **Start**

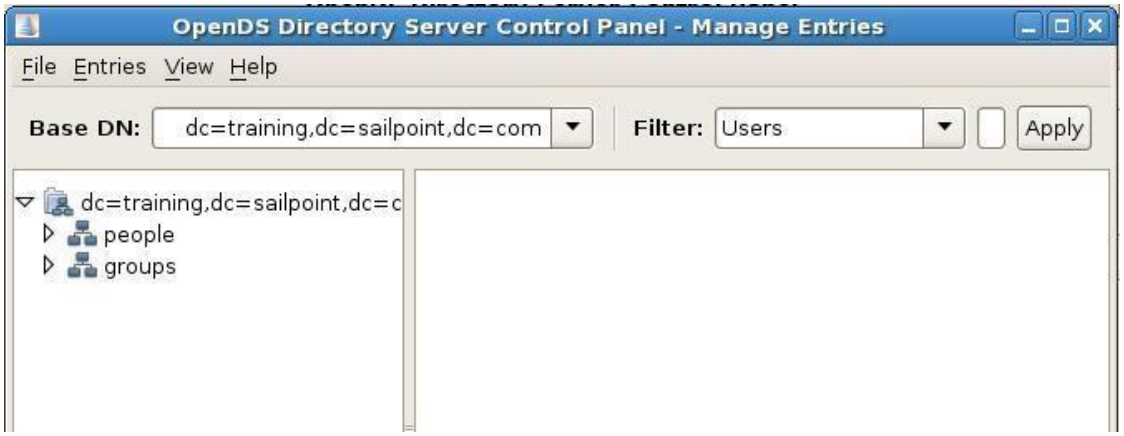
4. When prompted, enter the password: **password**

5. Confirm that the Server Status shows **Started**.

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

   a. Under **Directory Data**, select **Manage Entries**

   b. With the Filter set to Users, expand dc=training,dc=sailpoint,dc=…



   c. Expand and view the **people**.

   d. Expand and list the **groups**: _____

   e. Close the Server Control Panel.

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 Setting ❼ Import from File** and where it says **Import File**, click **Browse…** and import **all the** files **one by one**:

   C:\Training\config\LDAP

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**

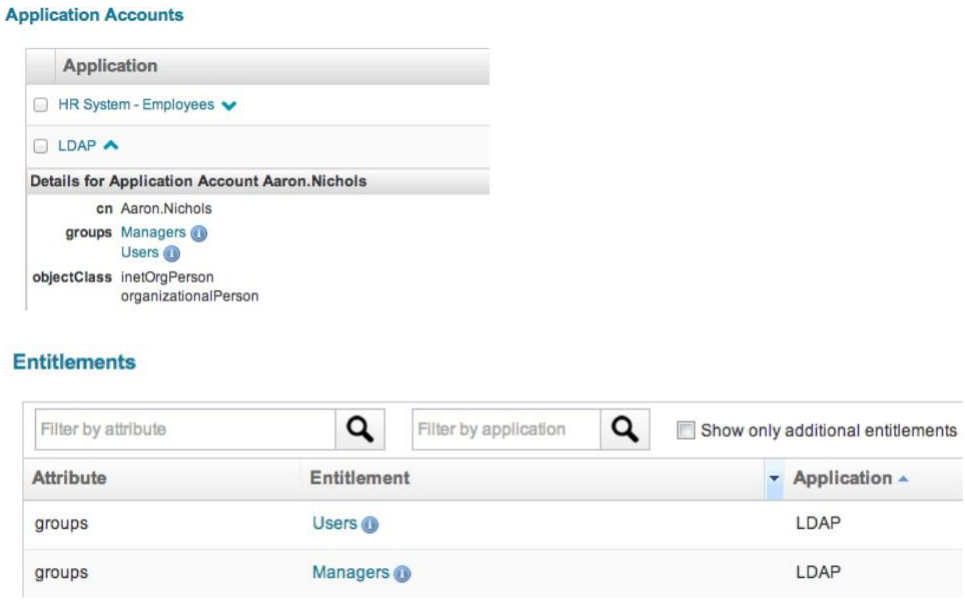
Aggregate LDAP Attributes	
Attribute	**Value**
Applications scanned	LDAP
Accounts scanned	229
Identities updated	229
Managed entitlements promoted	2
Identity Entitlements Created	2

   b. **Aggregate LDAP Groups**

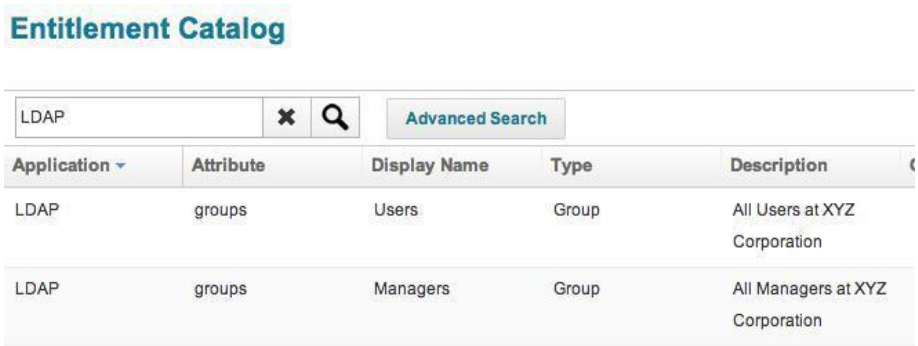
Aggregate LDAP Groups Attributes	
Attribute	**Value**
Applications scanned	LDAP
Groups scanned	2
Groups updated	2

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



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



Refresh Identities

Once aggregations are complete, an identity refresh is required to fully promote all identity attributes. 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. On the **Setup❼Tasks** tab, search for and right click the **Refresh Identity Cube** task.