# **CA Identity Portal 14.2 – Advanced Training**

# Lab Guide

Instructor: **David Franco** | Sr. Software Engineer | Identity Portal R&D | [David.Franco@ca.com](mailto:David.Franco@ca.com)

Contents

[Labs Prerequisites 3](#_Toc520929218)

[Lab 01: Form Handlers, API and Prop Contexts 4](#_Toc520929219)

[Chapter 1: Enrich the 'Modify User Contact Info' form 4](#_Toc520929220)

[Chapter 2: Creating a 'View User Contact Info' form 7](#_Toc520929222)

[Chapter 3: Identity Suite Deployment 13](#_Toc520929223)

[Lab 02: Plugins 19](#_Toc520929224)

[Chapter 1: RhinoJS (Javascript) Plugin 19](#_Toc520929225)

[Chapter 2: Java Plugin 22](#_Toc520929228)

[Chapter 3: Add imadmin to the ‘[LAB] Human Resources’ Admin Role 31](#_Toc520929231)

[Chapter 4: Create a Policy Xpress Policy 33](#_Toc520929232)

[Chapter 5: Create an Employee in CA Identity Manager 46](#_Toc520929233)

[Lab 03: Service Actions 48](#_Toc520929234)

[Chapter 1: Create a Module, Module Action, Form and Task in CA Identity Portal 48](#_Toc520929235)

[Chapter 2: Create an Employee from CA Identity Portal 63](#_Toc520929236)

[Chapter 3: Create a Business Use Case 66](#_Toc520929237)

[Lab 4: Bulk Configuration 79](#_Toc520929238)

# Labs Prerequisites

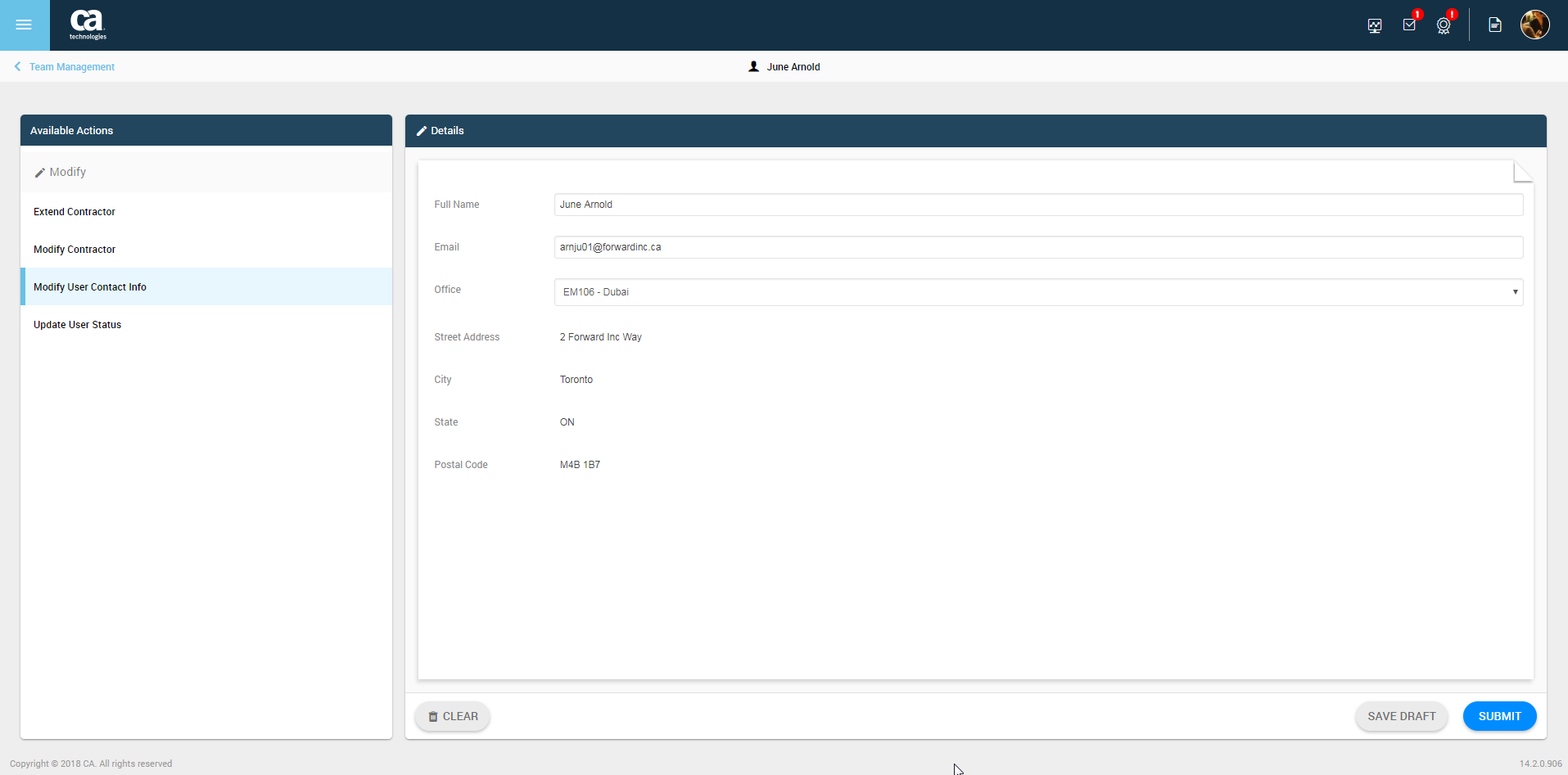
In the following labs you will need to have an Identity Suite 14.2 vApp environment ready.

Moreover, it is assumed that you have the demo environment from the Identity Suite 14.2 DOD environment. If you do not use the Identity Suite 14.2 DOD you need to deploy the Training Lab Artifact on you 14.2.

Code snippets and files that will be used throughout the labs are available in the public Git repository "duviduduvid/AdvancedTraining". Links to the files will be supplied wherever relevant. Simply browse to the link on your browser to fetch the needed code snippets or files.

# Lab 01: Form Handlers, API and Prop Contexts

## Chapter 1: Enrich the 'Modify User Contact Info' form Introduction

The DOD environment includes a Module named 'Team Management' which has a 'Modify User Contact Info' module action. The module action uses the 'Modify User Contact Info' form.  
  


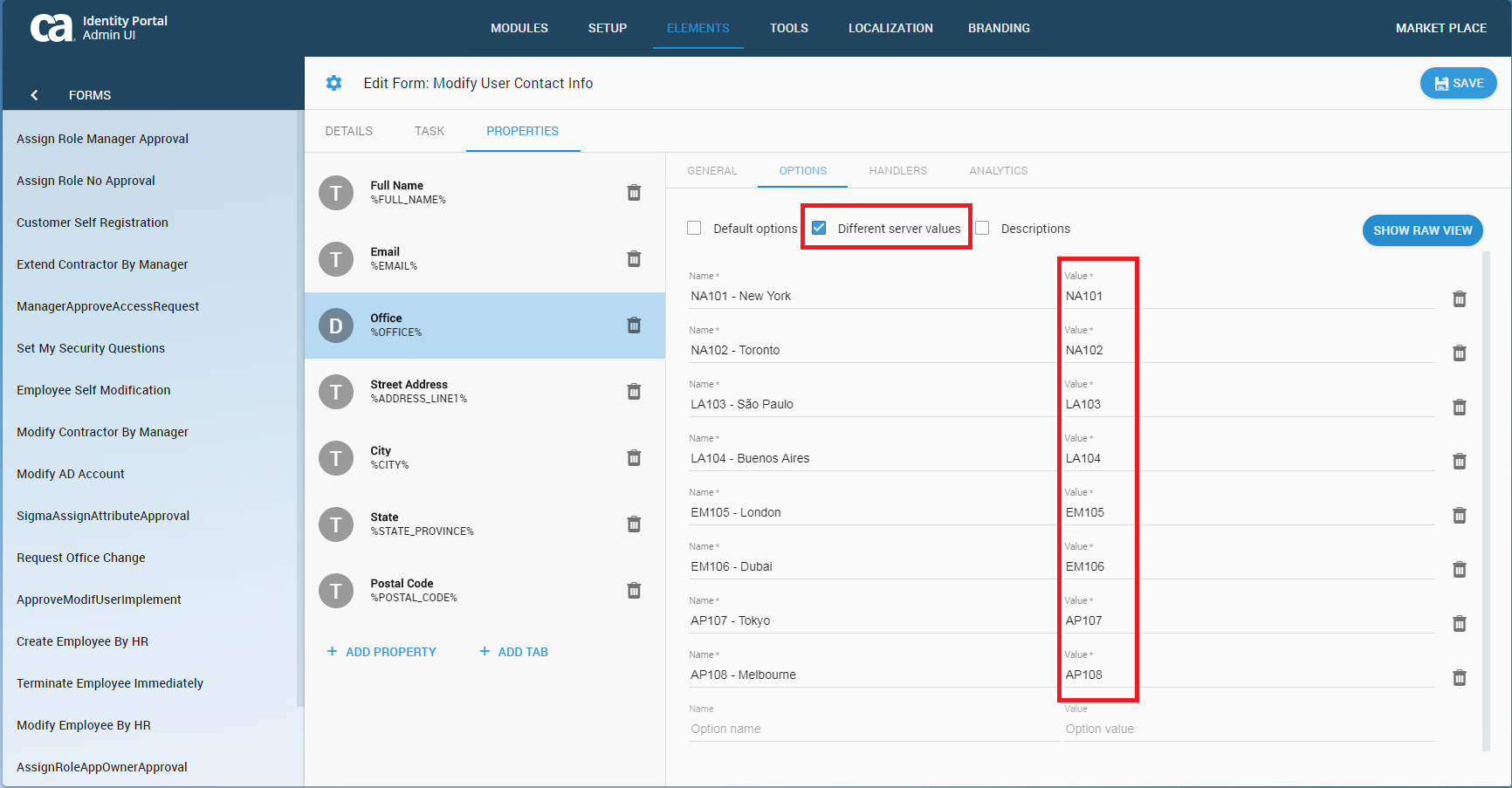
### Steps

1. Before we start, export the form configuration for backup:  
   Browse to the Identity Portal Management console (Admin UI).   
   Click on the TOOLS tab. In the Export screen, choose the Forms category and select the form 'Modify User Contact Info'. Click on the Export button and save the file.
2. Click on the 'ELEMENTS' tab and choose ’Forms' from the left panel menu.

Search for 'Modify User Contact Info' form and open it for editing. Change the form name to '[Advanced Training] Modify User Contact Info' and save the change.

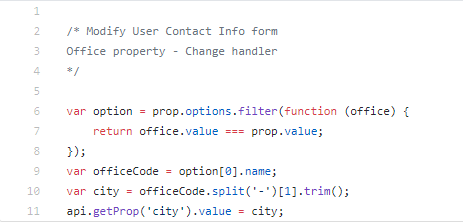
1. Click on the 'Office' form property and then open its 'OPTIONS' tab.   
   Notice that the options do not have a different server value other than the display value.   
   The problem is that these values do not match the values in Identity Manager. For instance, instead of "NA101 - New York" which appears in the list of options, the value that exists in the backend is "NA101". Let's fix it.  
     
   Mark the 'Different server values' checkbox. The 'value' column will appear to the right. Edit the server values so that they contain only the office code without the city name (see image below).

Save the changes.

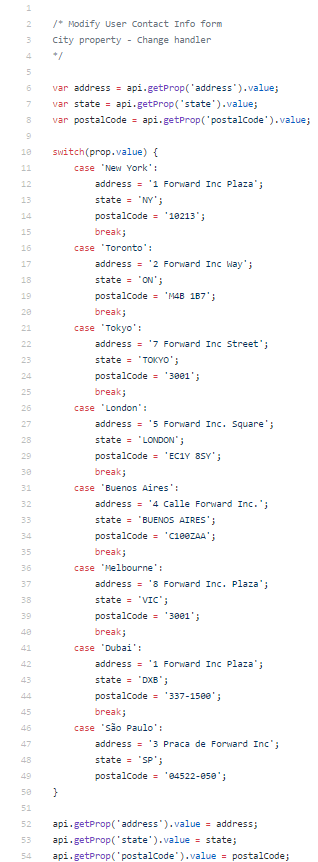


1. The properties 'Street Address', 'City', 'State' and 'Postal Code' are Read Only properties. These attributes are related to the value of the Office.  
   When the end user changes the Office value it will make the four properties above incorrect, yet since these are Read Only attributes the user cannot manually update their values.   
   We would like these values to be modified automatically according to the Office whenever the value of the Office property changes.  
     
   Add reference names to each one the properties:
   * Add the reference name 'address' to the 'Street Address' property.
   * Add the reference name 'city' to the 'City' property.
   * Add the reference name 'state' to the 'State' property.
   * Add the reference name 'postalCode' to the 'Postal Code' property.
2. Copy the code from the "*ModifyUserContactInfo-Office-ChangeHandler.js*" file in the following link to the Change handler of the 'Office' property and save the changes.

<https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab1/Chapter1/ModifyUserContactInfo-Office-ChangeHandler.js>



1. Copy the code from the "*ModifyUserContactInfo-City-ChangeHandler.js*" file in the following link to the Change handler of the 'City' property and save the changes:  
   <https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab1/Chapter1/ModifyUserContactInfo-City-ChangeHandler.js>



1. Login to Identity Portal User Console with jonri01 (pass: CAdemo123) and verify the behavior of the form logic is as expected.

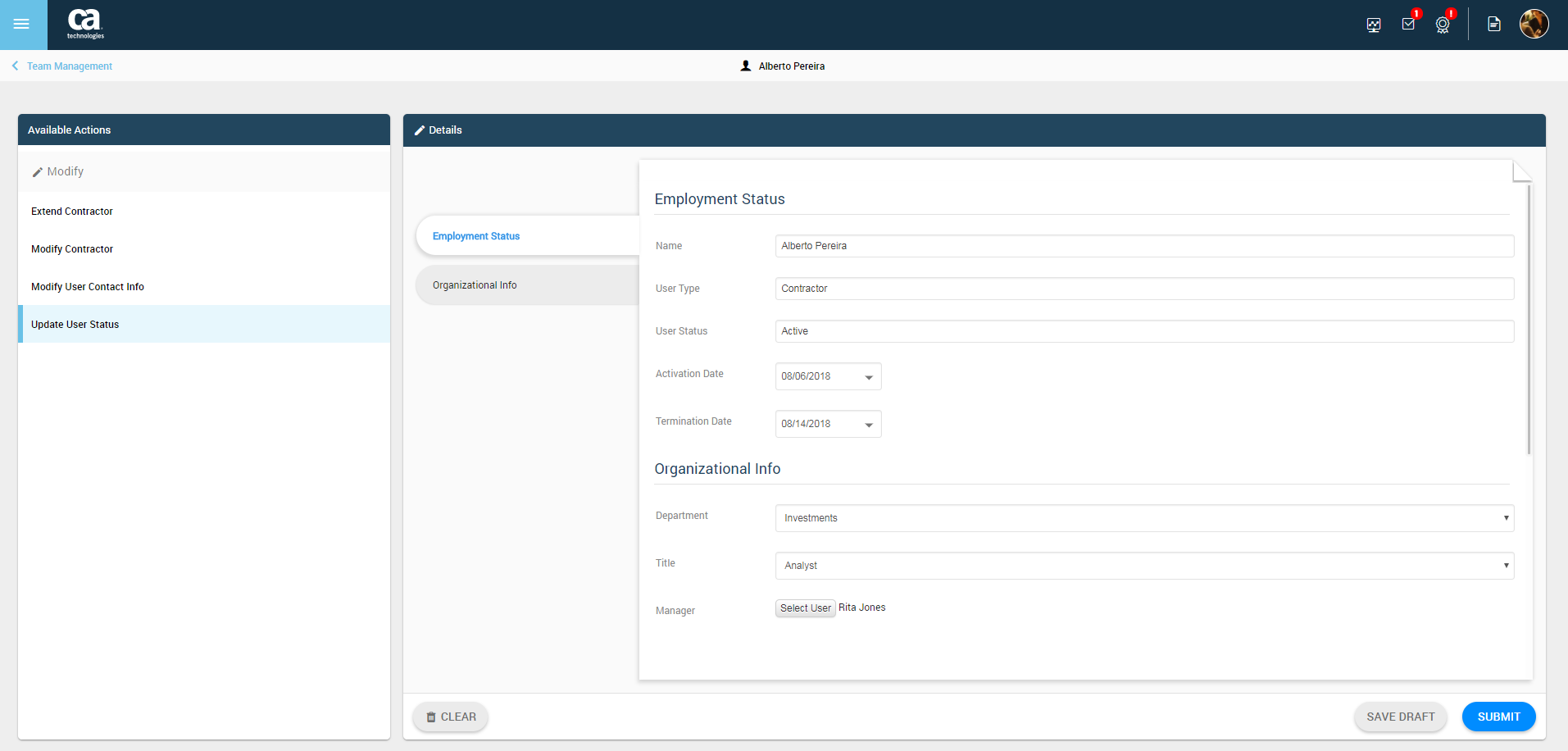
**You have successfully completed Lab01 Chapter 1: Enrich the 'Modify User Contact Info' form.**

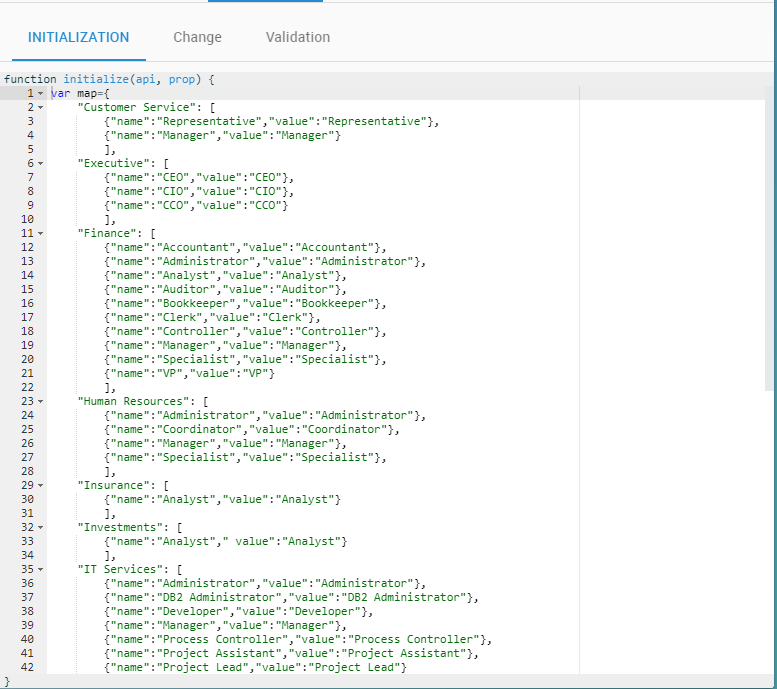
STOP! DO NOT PROCEED TO THE NEXT LAB UNTIL INSTRUCTED TO.



# Lab 02: Plugins

## Chapter 1: RhinoJS (Javascript) Plugin – Populate Title Options According to Department

Introduction  
The DOD environment includes a Module named 'Team Management' which has an 'Update User Status' module action. This module action uses the 'Update User Status' form.   
  


In this form, the options of the 'Title' property are populated dynamically according to the value of the 'Department' property. This is done using the Initialization and Change handlers of the 'Department' property: the code defines a mapping of Department names to a set of optional Titles. Then it populates the options attribute of the 'Title' property accordingly.   
Open the 'Update User Status' form configuration in the Admin UI and examine the code.   
  


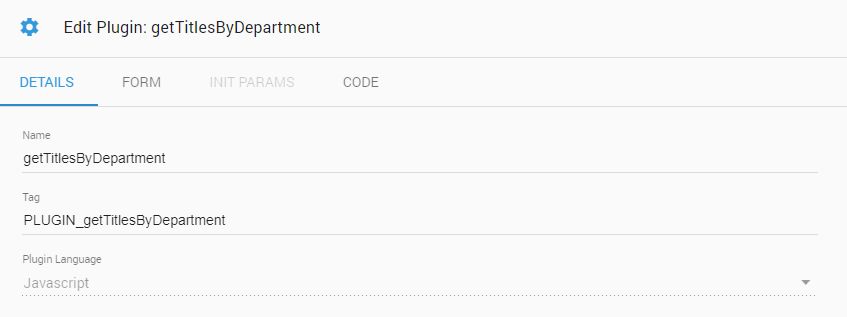
Yet, this logic exists in various forms throughout the system and the same code is duplicated repeatedly. This means that whenever you need to add a new Title or Department or modify an existing one, all the forms that use this code will need to be updated.

In this chapter we will create a Javascript plugin to replace the duplicated mapping of values, so it is extracted from the form itself and then can be used whenever needed.

### Steps

1. Browse to the Identity Portal Management console (Admin UI). Click on the 'ELEMENTS' tab and choose 'Plugins' in the left menu.
2. Create a new Javascript plugin:
   1. Click on the Create button.
   2. Fill in the following details in the DETAILS tab (notice that the Tag field is filled automatically):

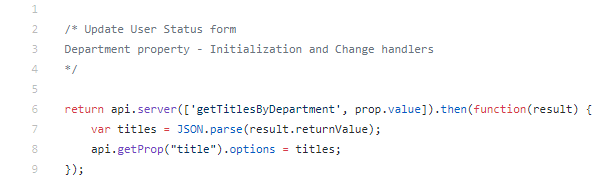
|  |  |
| --- | --- |
| Name | getTitlesByDepartment |
| Tag | PLUGIN\_getTitlesByDepartment |
| Plugin Language | Javascript |



1. In the plugin's FORM tab, choose the 'Update User Status' form.
2. Copy the code from the "*getTitlesByDepartment-plugin.js*" file in the following link to the CODE tab of the plugin.

<https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab2/Chapter1/getTitlesByDepartment-plugin.js>  
Save the plugin.

1. Now we need to update the Initialization and Change handlers code in the 'Update User Status' form to use the plugin:
2. Open the 'Update User Status' form for editing.
3. Remove the code in the Initialization and Change handlers of the 'Department' property.
4. Copy the code from the "*UpdateUserStatus-Department-InitAndChangeHandler.js*" file in the following link to the Initialization and Change handler of the Department property. <https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab2/Chapter1/UpdateUserStatus-Department-InitAndChangeHandler.js>



1. Make sure that the 'Title' property's reference name is 'title'.
2. Save the changes.
3. Login to Identity Portal User Console with jonri01 (pass: CAdemo123) and verify the behavior of the form logic is as expected.

**You have successfully completed Lab02 Chapter 1: RhinoJS (Javascript) Plugin – Populate Title Options According to Department**

## Chapter 2: RhinoJS (Javascript) Plugin – Fetch Options from a Database Table

### Introduction

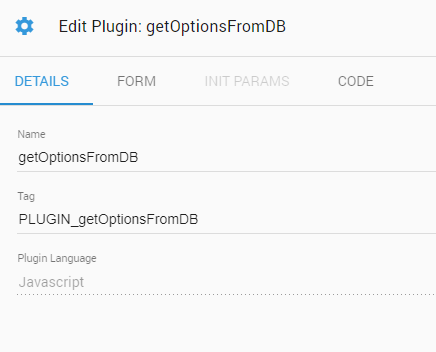
In many cases, the list of options for a certain form property should be fetched from a database table. The following Javascript plugin implements that sort of logic. It fetches the list of options from a specific column of a database table. We will use it to get the list of all permissions defined in the current Identity Portal deployment.

Notice that the plugin utilizes the Wildfly application server datasource to establish a connection to the database. A datasource is already defined for the Identity Portal database and the plugin simply uses this one, but in case you need to connect to a different database schema you can add a new datasource in the Wildfly configuration and use it instead.

### Steps

1. Browse to the Identity Portal Management console (Admin UI). Click on the 'ELEMENTS' tab and choose 'Plugins' in the left menu.
2. Create a new Javascript plugin:
   1. Click on the Create button.
   2. Fill in the following details in the DETAILS tab:

|  |  |
| --- | --- |
| Name | getOptionsFromDB |
| Tag | PLUGIN\_getOptionsFromDB |
| Plugin Language | Javascript |



* 1. In the FORM tab choose the 'Update User Status' form.
  2. Copy the code from the "*getOptionsFromDB-plugin.js*" file in the following link to the CODE tab of the plugin:

<https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab2/Chapter2/getOptionsFromDB-plugin.js>

* 1. Save the plugin.

1. asdfdsafdsaf

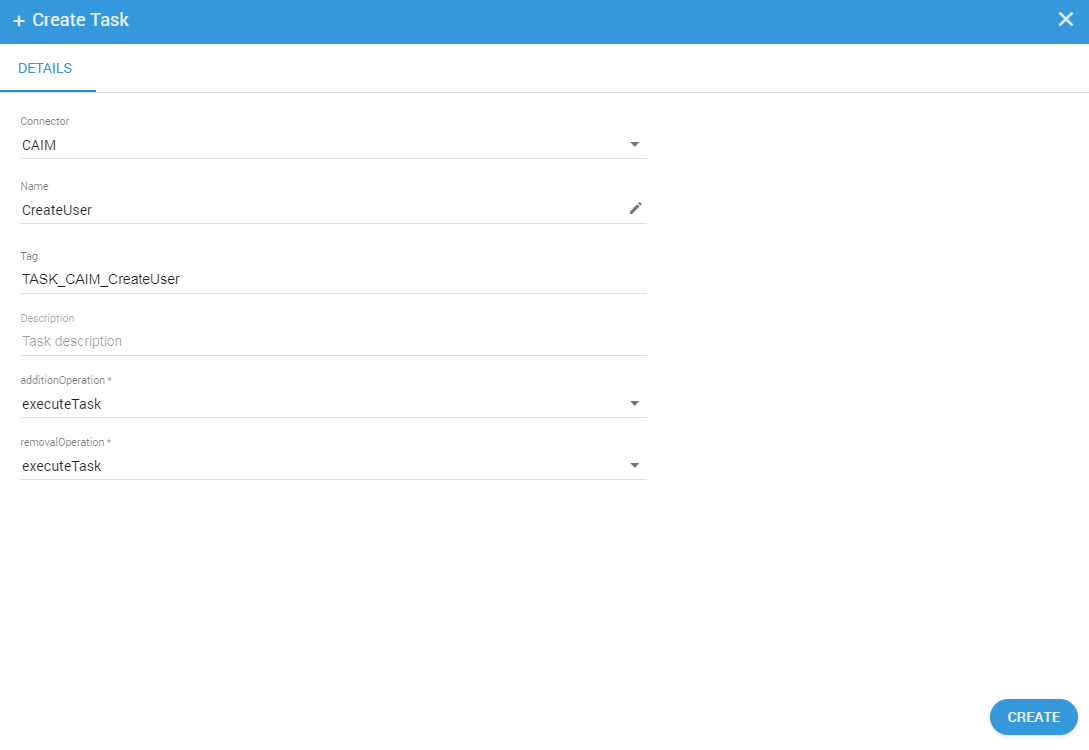
## Chapter 3: Java Plugin – Check User Id Uniqueness

Introduction  
The following Java plugin is used to verify the uniqueness of a given user id. We can use it in a Create User module action in which the end user manually types a value in the UserId property.  
The plugin connects to the Identity Manager userstore (LDAP) and returns a boolean value indicating whether a user with the given user id already exists in the LDAP.

Steps

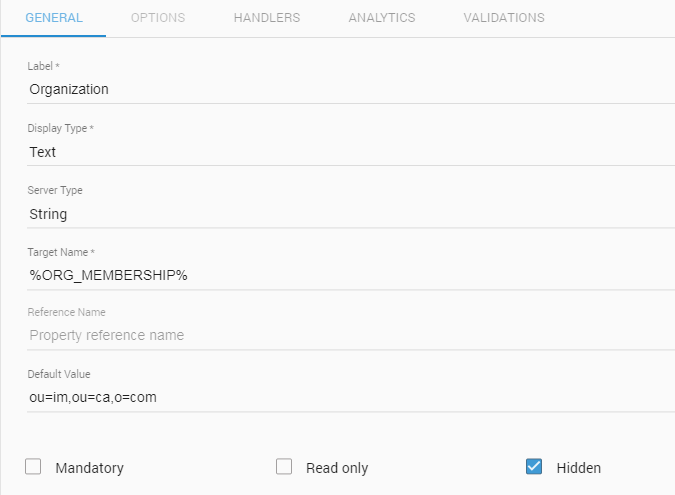
1. Browse to the Identity Portal Management Console (Admin UI).
2. Create the task 'Create User' in Identity Portal:
   1. Click on the 'ELEMENTS' tab and choose 'Tasks' from the left menu.
   2. Click on the '+CREATE' button.

|  |  |
| --- | --- |
| Connector | CAIM |
| Name | CreateUser |
| Tag | TASK\_CAIM\_CreateUser |
| additionOperation | executeTask |
| removalOperation | executeTask |

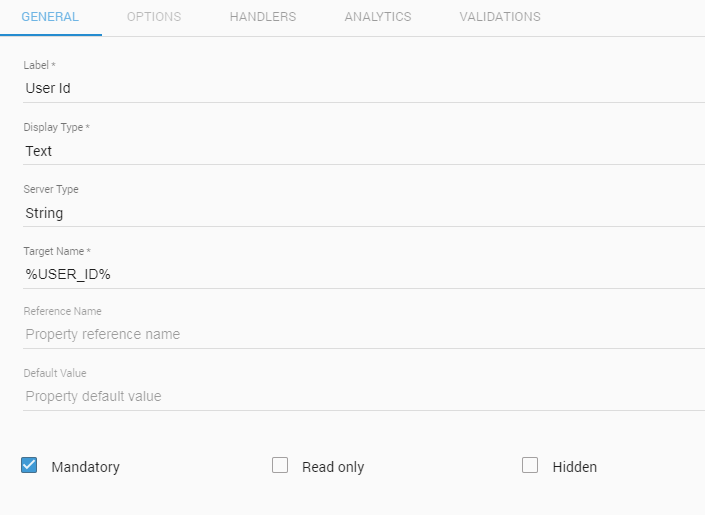
* 1. Fill in the following details in the Create Task dialog:  
       
       
       
       
       
       
     
  2. Click on the Create button and then on the FINISH button.

1. Create the form 'Create User Form':
   1. Click on 'Forms' in the left menu.
   2. Click on the '+CREATE' button.
   3. In the DETAILS tab of the Create Form dialog fill in the Name of the form – "Create User Form".
   4. In the TASK tab choose the 'Create Task' task.
   5. In the PROPERTIES tab add the following properties:

|  |  |
| --- | --- |
| Label | Organization |
| Display Type | Text |
| Server Type | String |
| Target Name | %ORG\_MEMBERSHIP% |
| Default Value | ou=im,ou=ca,o=com |
| Hidden | checked |

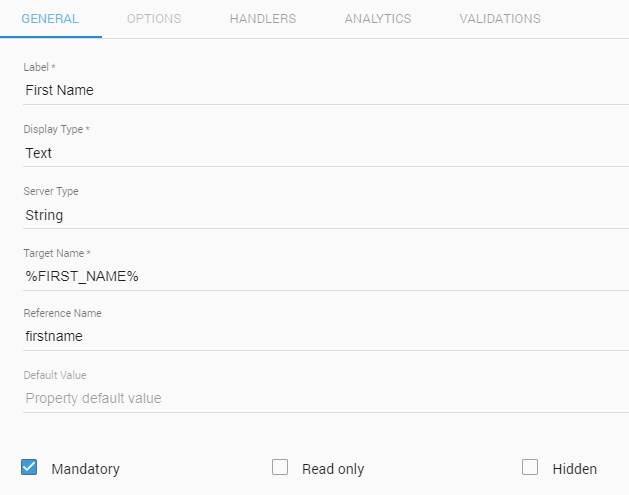
* + 1. **Organization** -
    2. **User Id** -

|  |  |
| --- | --- |
| Label | User Id |
| Display Type | Text |
| Server Type | String |
| Target Name | %USER\_ID% |
| Mandatory | checked |



* + 1. **First Name** –

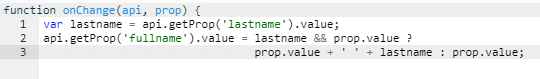
|  |  |
| --- | --- |
| Label | First Name |
| Display Type | Text |
| Server Type | String |
| Target Name | %FIRST\_NAME% |
| Reference Name | firstname |
| Mandatory | checked |

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Add the following code to the Change handler:

var lastname = api.getProp('lastname').value;

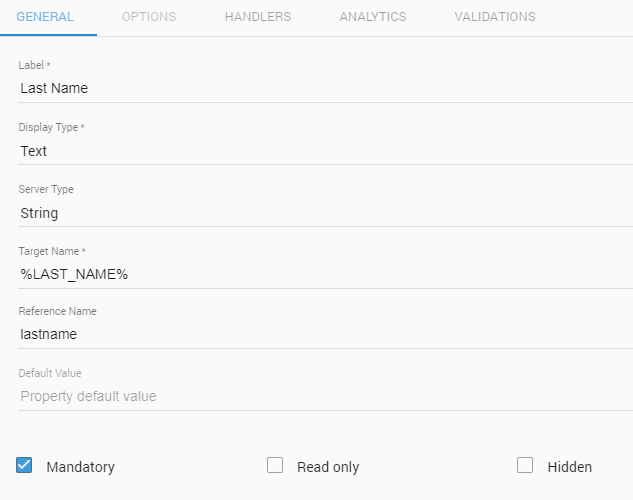
api.getProp('fullname').value = lastname && prop.value ?

prop.value + ' ' + lastname : prop.value;



* + 1. **Last Name** –

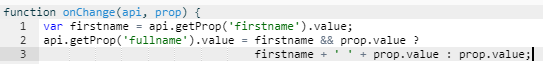
|  |  |
| --- | --- |
| Label | Last Name |
| Display Type | Text |
| Server Type | String |
| Target Name | %LAST\_NAME% |
| Reference Name | lastname |
| Mandatory | checked |



Add the following code to the Change handler:  
var firstname = api.getProp('firstname').value;

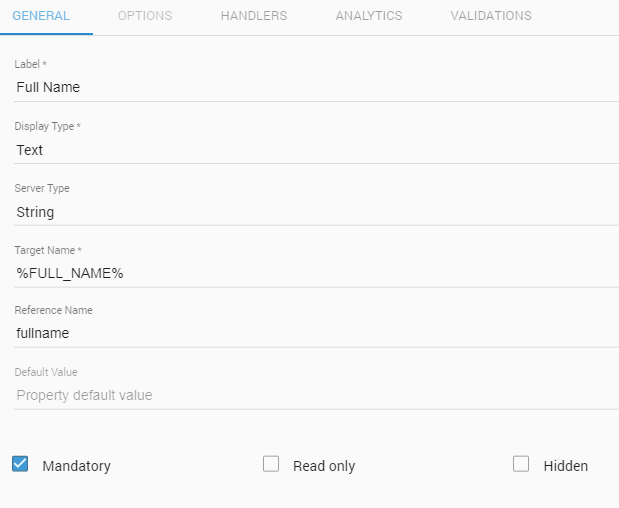
api.getProp('fullname').value = firstname && prop.value ?

firstname + ' ' + prop.value : prop.value;



* + 1. **Full Name** -

|  |  |
| --- | --- |
| Label | Full Name |
| Display Type | Text |
| Server Type | String |
| Target Name | %FULL\_NAME% |
| Reference Name | fullname |
| Mandatory | checked |

* + 1. 
    2. **Password** -

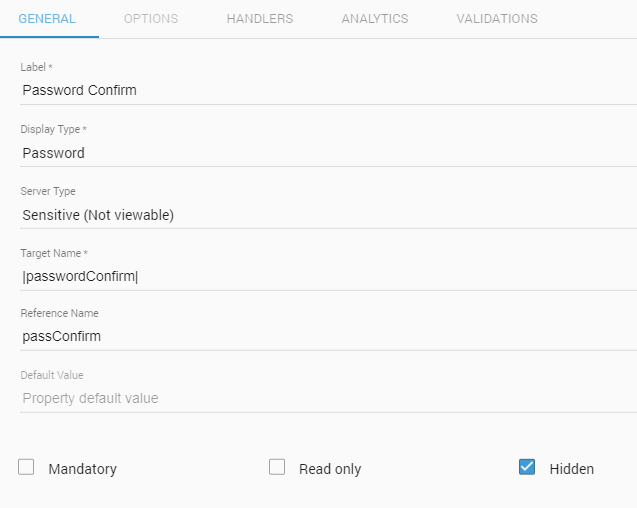
|  |  |
| --- | --- |
| Label | Password |
| Display Type | Password |
| Server Type | Sensitive (Not Viewable) |
| Target Name | %PASSWORD% |
| Mandatory | checked |

* + 1. Add the following code to the Change handler:

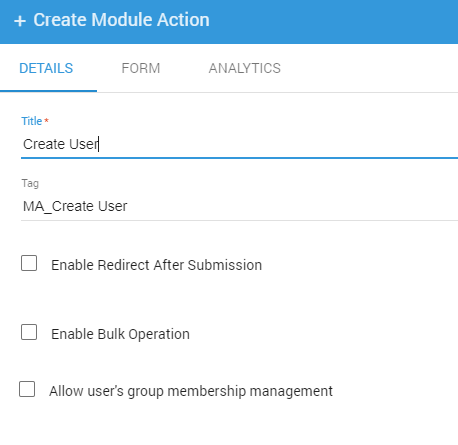
api.getProp('passConfirm').value = prop.value;  
  


* + 1. **Password Confirm** -

|  |  |
| --- | --- |
| Label | Password Confirm |
| Display Type | Password |
| Server Type | Sensitive (Not Viewable) |
| Target Name | |passwordConfirm| |
| Reference Name | passConfirm |
| Hidden | checked |



* 1. Click on the 'CREATE' button and then on 'FINISH'.

1. Create a new module action in the 'Employee Life Cycle' module:
   1. Click on the 'MODULES' tab and choose the 'Employee Life Cycle' module.
   2. Click on the 'ACTIONS' tab of the module and then click on the '+ Create module action' button.
   3. In the DETAILS tab of the Create Module Action dialog fill in the title 'Create User'.  
        
      
   4. In the FORM tab choose the 'Create User Form'.
   5. Click on 'CREATE' button and then on 'FINISH'.
2. Copy the code from the "*LdapPluginUniquenessCheck.java*" file in the following link  
   <https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab2/Chapter3/LdapPluginUniquenessCheck.java>

Save it to a file on your local machine with the name "*LdapPluginUniquenessCheck.java"*.   
Copy the file to the /home/config directory on you vApp machine.

1. Open SSH connection to your vApp machine.   
   Run the following command from the /home/config directory:  
   wget 'https://github.com/duviduduvid/AdvancedTraining/raw/master/labs/lab2/Chapter3/sigma.plugin-0.0.2-SNAPSHOT.jar'

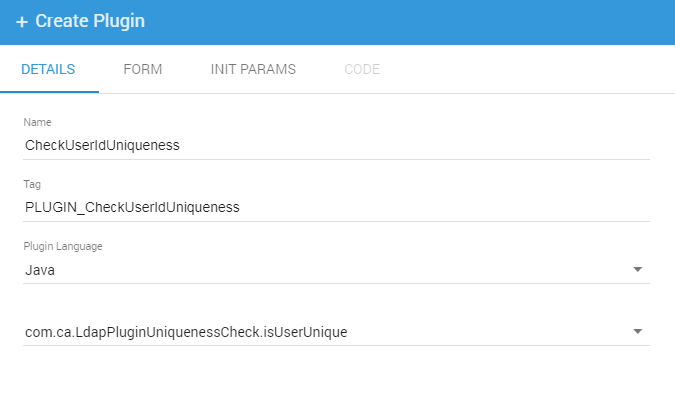
This will download the "sigma.plugin-0.0.2-SNAPSHOT.jar" to your vApp machine.   
This jar is required for the compilation of Java plugins. It's included in the sigma.war file, in the path "sigma.war\WEB-INF\lib".

1. Run the following command to compile the Java class:

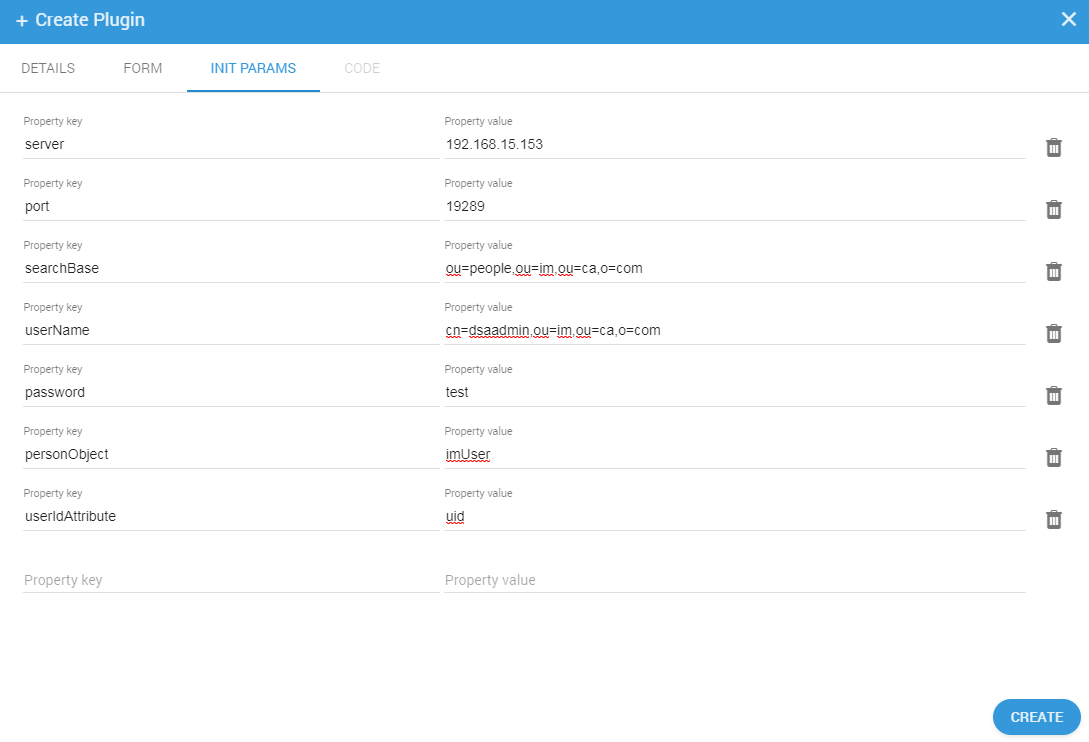
javac -cp sigma.plugin-0.0.2-SNAPSHOT.jar LdapPluginUniquenessCheck.java  
  
It will create the "LdapPluginUniquenessCheck.class" on the home directory.

1. Create the needed path in the Plugins directory on your vApp machine and copy the class file to it:
   1. By default, the Plugins directory on vApp is in "/opt/CA/IdentityPortal/plugins".
   2. We need to create the appropriate folder hierarchy to match the package declared in the *LdapPluginUniquenessCheck.java* file, which is "/com/ca".   
      Run the command:  
      mkdir -p /opt/CA/IdentityPortal/plugins/com/ca
   3. Copy the class file to the path above using the following command:  
      cp LdapPluginUniquenessCheck.class /opt/CA/IdentityPortal/plugins/com/ca
2. In the web browser, open a new tab and browse to the following url so that the server will load the new plugin:

*<identity-portal-address>/sigma/rest/admin/plugins/refresh*

1. In Identity Portal Management Console, create a new Java plugin:
   1. Click on the 'ELEMENTS' tab and then click on 'Plugins' in the left menu.
   2. Click on the '+CREATE' button.
   3. In the DETAILS tab of the 'Create Plugin' dialog, type the Name "CheckUserIdUniqueness".
   4. Choose the Plugin Language 'Java' and in the 'Select a method' dropdown choose function 'isUserUnique'.   
        
        
      
   5. In the FORM tab choose the form 'Create User Form'.
   6. In the INIT PARAMS tab, fill in the following details:

|  |  |
| --- | --- |
| server | <userstore-address> |
| port | 19289 |
| searchBase | ou=people,ou=im,ou=ca,o=com |
| userName | cn=dsaadmin,ou=im,ou=ca,o=com |
| password | test |
| personObject | imUser |
| userIdAttribute | uid |



* 1. Click on 'CREATE' and then 'FINISH'.

1. Add the validation handler code that uses the plugin to the User Id property in the 'Create User Form':
   1. Edit the 'Create User Form'.
   2. Add the code from the "*CreateUserForm-UserId-ValidationHandler.js*" file from the following link to the Validation handler of the User Id property:  
      <https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab2/Chapter3/CreateUserForm-UserId-ValidationHandler.js>
   3. Save the changes.
2. Login to Identity Portal User Console with jonri01 (pass: CAdemo123). Go to the 'Employee Life Cycle' module and choose the 'Create User' action.  
   Test the plugin works as expected.

**You have successfully completed Lab02 Chapter 3: Java Plugin – Check User Id Uniqueness**

 STOP! DO NOT PROCEED TO THE NEXT LAB UNTIL INSTRUCTED TO.

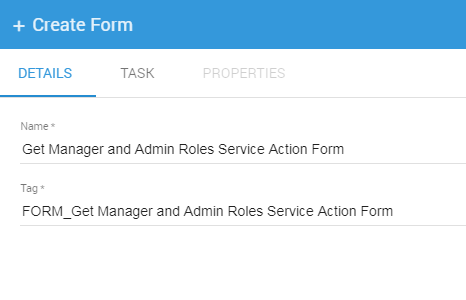
# Lab 03: Service Actions

Introduction   
In the following lab you will create a Service Action that will allow you to fetch user information from Identity Manager while performing an action on a different user.

We will use the 'Update User Status' form from the 'Team Management' module. The form has a User Selector property for the Manager attribute. We will add a change handler to the Manager property to display the manager's current admin roles and his/her own manager.

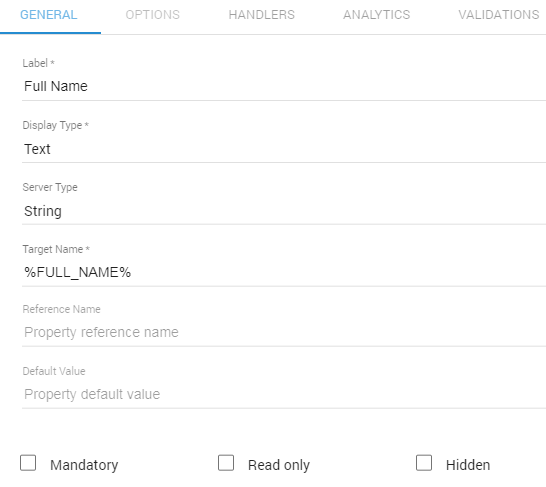
Steps

1. Browse to Identity Portal Management Console (Admin UI). Click on the 'ELEMETS' tab.
2. Create the form 'Get Manager and Admin Roles Service Action Form':
   1. Click on 'Forms' in the left menu.
   2. Click on the '+CREATE' button.
   3. In the DETAILS tab of the Create Form dialog fill in the Name of the form – "Get Manager and Admin Roles Service Action Form".



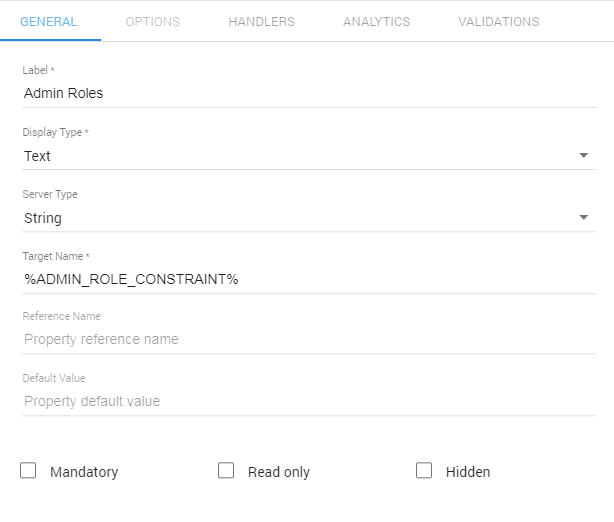
* 1. In the TASK tab choose the 'ModifyUesrByManager' task.  
     Note – the scope of the service action that will use this form will be the same as the scope of the admin task above.
  2. In the PROPERTIES tab, add the following properties:
     1. **Full Name**

|  |  |
| --- | --- |
| Label | Full Name |
| Display Type | Text |
| Server Type | String |
| Target Name | %FULL\_NAME% |



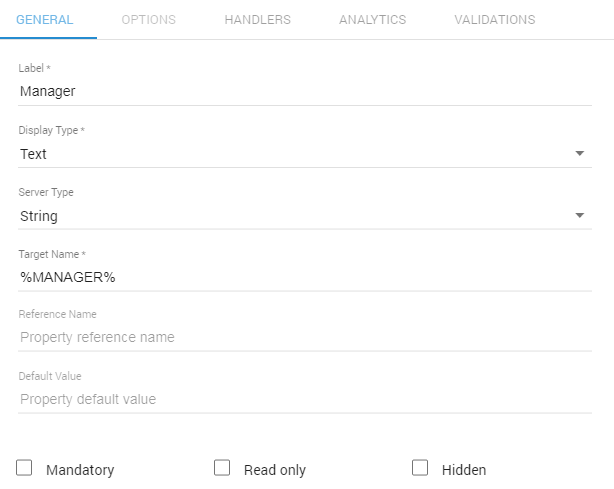
|  |  |
| --- | --- |
| Label | Admin Roles |
| Display Type | Text |
| Server Type | String |
| Target Name | %ADMIN\_ROLE\_CONSTRAINT% |

* + 1. **Admin Roles**

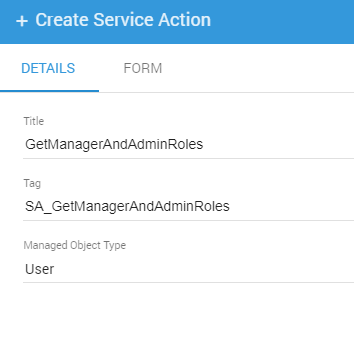


|  |  |
| --- | --- |
| Label | Manager |
| Display Type | Text |
| Server Type | String |
| Target Name | %MANAGER% |

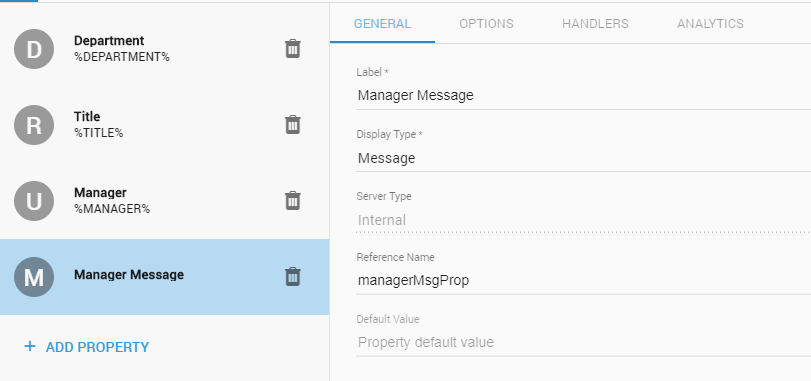
* + 1. **Manager**



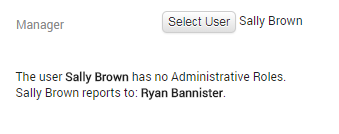
* 1. Click on the 'CREATE' button and then on 'FINISH'.

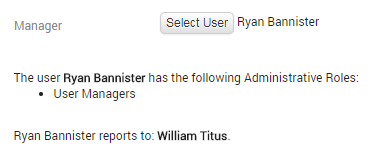
1. Create the 'GetManagerAndAdminRoles' Service Action:
   1. In the DETAILS tab, fill in the Title 'GetManagerAndAdminRoles' and choose User in the Managed Object Type dropdown.  
        
      
   2. In the FORM tab, choose the form 'Get Manager and Admin Roles Service Action Form'.
   3. Click on the 'CREATE' button and then on 'FINISH'.
2. Edit the 'Update User Status' form:
   1. Add a new property of type 'Message' in the 'Organizational Info' tab, beneath the 'Manager' property, with the following details:

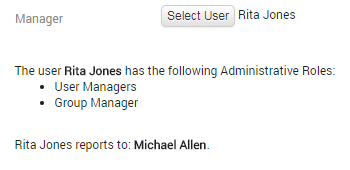
|  |  |
| --- | --- |
| Name | Manager Message |
| Display Type | Message |
| Reference Name | managerMsgProp |



* 1. Copy the code from the "*UpdateUserStatus-Manager-ChangeHandler.js*" file in the following link to the Change handler of the Manager property:  
     <https://github.com/duviduduvid/AdvancedTraining/blob/master/labs/lab3/UpdateUserStatus-Manager-ChangeHandler.js>

1. Verify the new logic of the form works as expected:
   1. Login to Identity Portal User Console with jonri01 (pass: CAdemo123).
   2. Open the Team Management module and choose the Update User Status action.
   3. Choose a user from the search result.
   4. When the form is loaded, scroll down to the Manager property and change its values.
   5. Check that the message displayed correctly.   
      For instance, if you select 'Sally Brown' the result should be:  
      

If you select 'Ryan Bannister' the result should be:  
  


If you select 'Rita Jones' (or click on the 'Select Myself' button in the User Selector) the result will be:  
  


If you want this information to be displayed immediately when the form is loaded, add the same code from the Manager's property Change handler to the property's Initialization handler. Do not forget to add "return" before the first "api.service" call so that the form initialization will wait for the result to be ready.

**You have successfully completed Lab 03: Service Actions.**

 STOP! DO NOT PROCEED TO THE NEXT LAB UNTIL INSTRUCTED TO.

# Lab 4: Bulk Configuration

## Chapter 1: Bulk Create User – Onboarding