## Overview

The user management engine (UME) provides a centralized user management for all Java applications and can be configured to work with user management data from multiple data sources. It is seamlessly integrated in the SAP NetWeaver Application Server (AS) Java as its default user store and can be administrated using the administration tools of the AS Java.

The UME adds business value by enabling you to leverage your existing system infrastructure by accessing user-related data on an existing LDAP directory, an AS ABAP system, a database of the AS Java, or any combination of these. In addition it reduces administrative overhead by allowing you to perform centralized user administration.

SAPUME Connector can be local or remote. It’s a war file that can be deployed in OpenIAM Application Server or in a different Application Server.

With SAPRP Connector you can do the following:

1. **Provisioning/Deprovisioning** - The Connector can provision Users and s with all attributes that the target system supports. The list of the attributes you can find in *“Configuring the Policy Map”*. In a common case, you can add any attribute in the Policy Map linked to the Groovy script; and the attribute value will be provisioned to the target system..
2. **Group Management** - The Connector supports this feature.
3. **Role Management**
4. **Reconciliation for**
   1. **Users**
   2. **Groups**
   3. **Roles**
5. **Password Synchronization** - The Connector can set up User passwords and change them.

*SAPUME Connector uses SAPUME SPML interface to communication with SAPUME system.*

## Requirements for the SAPUME

**On the connector side:**

* Operating system **any supported by AppServer**
* Application Server **: Jboss**
* JDK 1.7

**On the SAPUME server side:**

* Add two roles in SAP UME
  + SPML\_FULL\_ACCESS\_ROLE (SPML full access Role)
  + With the actions :
    - Spml\_Read\_Action
    - Spml\_Write\_Action
  + SPML\_READ\_ACCESS\_ROLE (SPML read only access Role)
  + With the actions :
    - Spml\_Read\_Action
* Add two users in SAP UME
  + spmluser (SPML full access User )
  + Rol e: SPML\_FULL\_ACCESS\_ROLE
  + spmluser\_readonly (SPML read only access User)
  + Role: SPML\_READ\_ACCESS\_ROLE

## Installing SAP UME Connector

Download and deploy SAPUMEConnector.war.

## Configure SAP UME Connector

* Log 🡪 SAPUMEConnector.war\WEB-INF\classes\log4j.properties
* Log file Abosulte Path 🡪

SAPUMEConnectorService.properties --> SAPUMEConnector.war\WEB-INF\web.xml

* Configuration Properties 🡪SAPUMEConnectorService.properties
  + If you want to cipher a property value, you can specify the “CHIPER\_” prefix. When Connector is deployed update this value with a value encrypted
  + **sap.general.policyMapAtts** property allows set attribute lookups definition. (Attribute lookups groovy doesn’t runs ok if server is not deployed in OpenIAMM AppServer ).

This attribute list is used in PolicyMap definition and Managed System Viewer.

You can define a list of attribute separated by “,” with the following pattern:

FIELD\_NAME:METADATA\_TYPE

* + - FIELD\_NAME: Name of the SAP UME Field
    - METADATA\_TYPE: You can specify if a field is HIDDEN o READ\_ONLY

|  |  |  |
| --- | --- | --- |
| Property | Description | Sample Value |
| sapume.general.  changePasswordAtNextLogon | Define is OpenIAM change Password, force to a user to change pwd at next Logon. If False, Connector change password to a dummy password and next change password with password passed to the connector | True\False |
| CIPHER\_sapume.general.  dummyPassword | Password used to dummy change. |  |
| sapume.general.dateMask | Date Format | yyyy-MM-dd |
| sapume.general.enableAccountsDate | When SAP account is enable. Value to set VALIDTO attribute | 9999-12-31 |
| sapume.general.traceSpmlRequest | Activate SPML trace. | True\False |
| sap.general.policyMapAtts | Attributte Lookups for PolicyMap attributes |  |
| sap.general.managedSysAtts | Attributte Lookups for Managed System attributes |  |
| sap.recon.search.users.atts | Attributes gets in recon search |  |

Sample SAPUMEConnectorService.properties

###################################################################

# If ciphered property is required, then add 'CIPHER\_' preffix to property #

#####################################################################

sapume.general.changePasswordAtNextLogon=False

CIPHER\_sapume.general.dummyPassword=sia12345

sapume.general.dateMask=yyyy-MM-dd

sapume.general.enableAccountsDate=9999-12-31

sapume.general.traceSpmlRequest=True

sapume.general.policyMapAtts=logonname,id:READ\_ONLY,password:HIDDEN,firstname,lastname,city,displayname,title,streetaddress,state,country,zip,email,fax,telephone,mobile,department,jobtitle,salutation,timezone,locale,validfrom,validto,islocked,securitypolicy,assignedgroups,assignedroles,uniquename,description

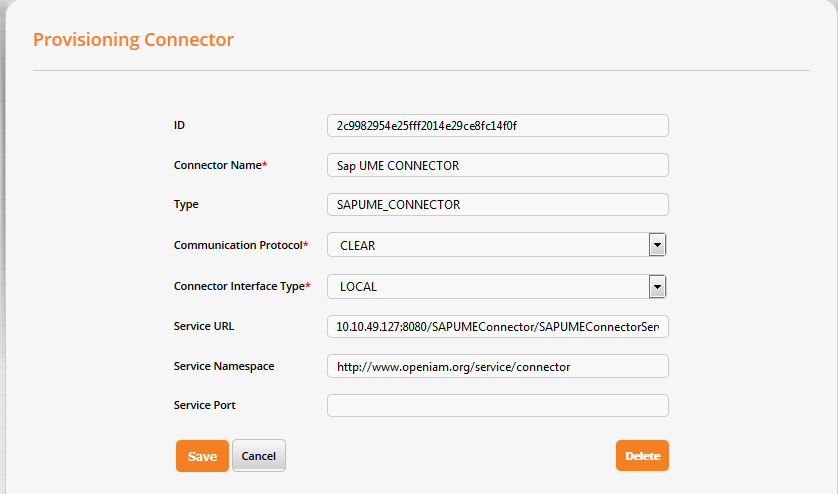
sapume.general.managedSysAtts=GROUP\_MEMBERSHIP\_ENABLED,ON\_DELETE,INCLUDE\_IN\_PASSWORD\_SYNC

sapume.recon.search.users.atts=logonname

## Configuring Connection to SAP UME

**Step 1: Configuring the Connector**

To access and configure the Connector, select Provisioning → Connectors → select a Connector.

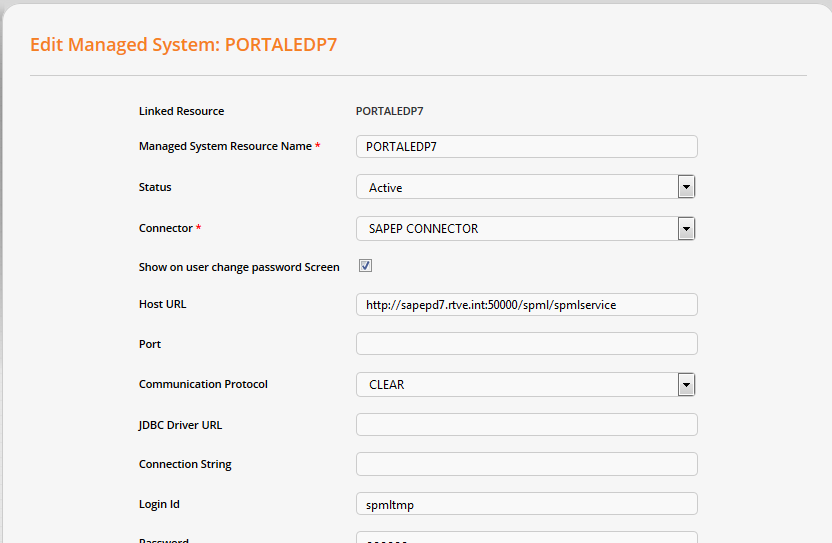


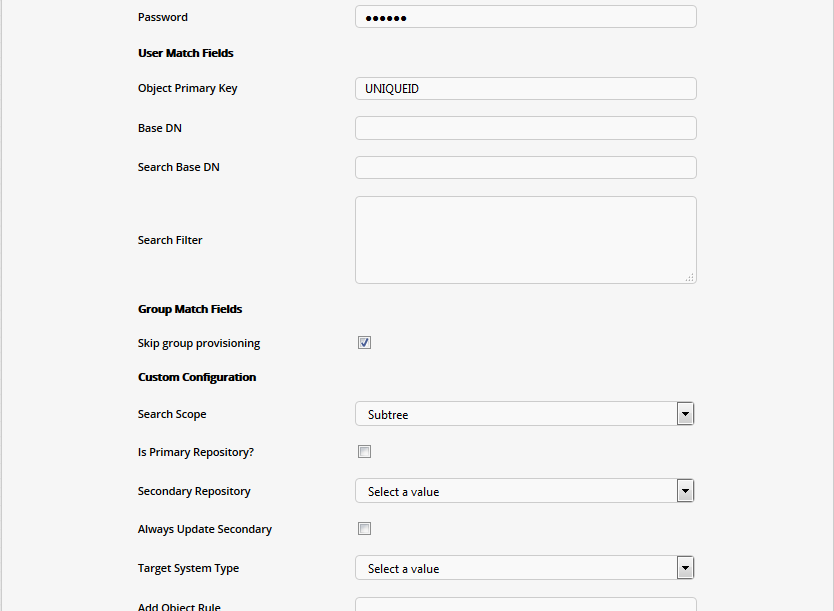
### Step 2: Configuring the Managed System

Once the Connector has been defined in the Identity Manager, you can configure the connectivity to SAP UME. See the following screenshot.

To configure the properties of the Managed System, select Provisioning → Managed System → "Edit" icon. Refer to [Viewing and Editing Managed Systems](http://wiki.openiam.com/display/IAMSUITEV3/Adding+a+Managed+System).

To add a new Managed System, select Provisioning → Managed System → Create Managed System. Refer to [Adding a Managed System](http://wiki.openiam.com/display/IAMSUITEV3/Adding+a+Managed+System).





**Host URL --> SPML Web Service EndPoint** from SAP UME

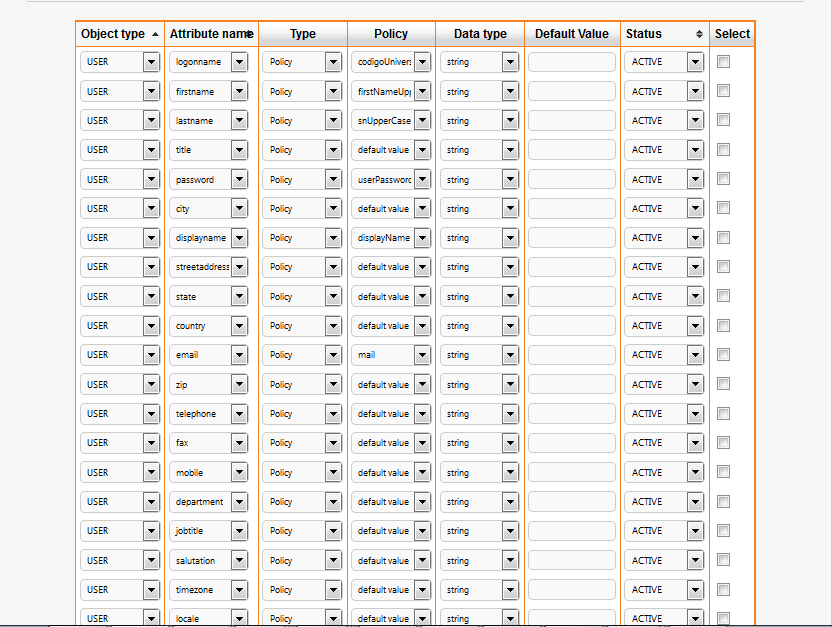
Sample --> http://10.1.11.58:50000/spml/spmlservice

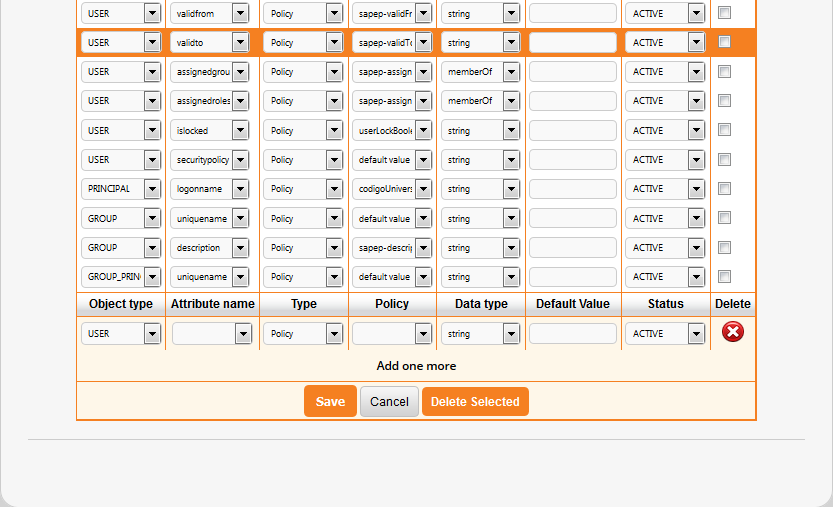
### Step 3: Configuring the Policy Map

The next task is to determine what attributes you need to pass to the SAP connector.

Combined with our use of the Groovy scripting language, we can dynamically derive any attribute that is needed in SAP from the data maintained within OpenIAM. The screenshot below provides a sample mapping between rules in the Identity Manager, called attribute policies, and the SA^P attributes. More details on the attribute policies and mappings can be found on the following page: [Defining Policy Maps](http://wiki.openiam.com/display/IAMSUITEV3/Defining+Policy+Maps).

To configure the Policy Map, go to Provisioning → Managed System → "Edit" icon → Policy Map.





#### Attributes alllowed

#######################

TAB General Information

#######################

Logon ID logonname

Last Name lastname

First Name firstname

E-Mail Address email

Form of Address salutation

Language locale

Security Policy securitypolicy ('Default' ó 'technical')

Unique ID id

Display Name displayname

Title title

Password password

#######################

TAB Account Information

#######################

Start Date of Account Validity validfrom\*

End Date of Account Validity validto \*

*\*Format Output in OpenIAM is "yyyy-MM-dd". Connector transform this date format to date format allowed by SAP UME ("yyyyMMddHHmmss'Z'")*

User Account Locked islocked

#######################

TAB Contact Information

#######################

Telephone telephone

Fax fax

Mobile mobile

Street streetaddress

City city

State/Province state

Zip/Postal Code zip

Country country

Time Zone timezone

#######################

TAB Additional Information

#######################

Position jobtitle

Department department

##################

TAB Assigned Roles

##################

Assigned Roles assignedroles

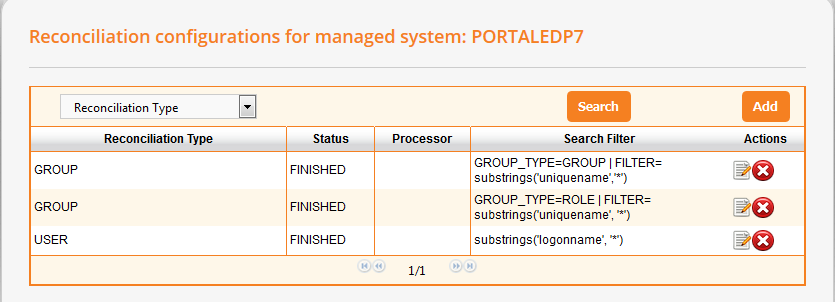
###################

TAB Assigned Groups

###################

Assigned Groups assignedgroups

## Reconciliation



### Users

* Filter Operations : equalityMatch, substrings
* Filter Logical operands: “AND” y “OR”
* Complex filters: Filter can be formed ith a one or more clauses , but with only one logical operand (AND or OR)

* Correct filters samples :
  + substrings('logonname', 'SAPUME')
  + equalityMatch('logonname', 'SAPUMEUSU01')
  + equalityMatch('logonname', 'SAPUMEUSU01') or equalityMatch('logonname', 'SAPUMEUSU02')
  + substrings('logonname', 'a') or substrings('logonname', 'z') or equalityMatch('logonname', 'sapusu01')
* Incorrect filter samples :
  + equalityMatch('logonname', 'SAPUMEUSU01') **or** equalityMatch('logonname', 'SAPUMEUSU02') **and** substrings('logonname', 'z'):

Reconciliation can be full or Incremental.

You can use *UpdateSince* *MangedSysParam* to indicate incremental mode and the date to filter last updates.

SAPSearchQuery.groovy checks if UpdateSince is informed and add

*"#TIMESTAMP#=yyyyMMddHHmmss"* to the search filter.

Connector checks this string to make a full\incremental search.Sample Incremental Filter received by the connector if you indicate UpdateSince:

substrings('logonname', 'SAPUME')#TIMESTAMP#=20150709131143Z

Connector detects #TIMESTAMP# and generates SPML filters like this:

<dsml:filter>

<dsml:and>

<dsml:substrings name='logonname'>

<dsml:any>SAPUME</dsml:any>

</dsml:substrings>

<dsml:greaterOrEqual name='lastmodifydate'>

<dsml:value>20131008015052Z</dsml:value>

</dsml:greaterOrEqual>

</dsml:and>

</dsml:filter>

### Groups/Roles

Pattern for Groups and Roles Filter:

GROUP\_TYPE=<groupType> | FILTER=<filter>

- GROUP\_TYPE --> "GROUP" or "ROLE"

- FILTER --> Filter expression like user filter expression.

Sample filters:

* GROUP\_TYPE=GROUP | FILTER= equalityMatch('uniquename', 'Authenticated Users')
* GROUP\_TYPE=ROLE | FILTER= substrings('uniquename', '\*')

Incremental reconciliation is not implemented.