SAP NetWeaver® Identity Management
Identity Center

Implementation guide
- Extension Framework

Version 7.1 Rev 2
Preface

The product

SAP NetWeaver Identity Management Identity Center is a high-end identity management solution, capable of handling a large amount of repositories containing an unlimited amount of information. The Identity Center offers a robust, flexible and scalable high-availability solution for workflow, provisioning, data synchronization and joining for a large number of data repositories.

The reader

This manual is written for people who are to extend an Identity Center configuration with 3rd party Java classes, like input validation, with the Extension Framework.

Prerequisites

To get the most benefit from this manual, you should have the following knowledge:

- Knowledge of the Identity Center.
- Basic knowledge about the SAP NetWeaver AS Java and its tools.
- Thorough knowledge about the SAP NetWeaver Developer Studio.

The following software is required:

- SAP NetWeaver Identity Management Identity Center 7.1 SP4 or newer is correctly installed and licensed.
- SAP NetWeaver AS Java as of Release 7.0 SP14 or higher, or Enhancement Package 1 for SAP NetWeaver Composition Environment 7.1, must be correctly installed and licensed.
- SAP NetWeaver Developer Studio 7.0 or 7.1.1 EHP 1.

The manual

This document describes how the Java class is implemented as a project in the SAP NetWeaver Developer Studio, how the class is deployed on the SAP NetWeaver running the User Interface. It also describes how to include the reference to the class from the task configuration in the Management Console.

This tutorial is not a substitution for training.

Related documents

You can find useful information in the following documents:

- SAP NetWeaver Identity Management Identity Center tutorials and help file.
- Relevant documentation for SAP NetWeaver AS Java and its tools.
Table of contents

**Introduction** .................................................................................................................................. 1
  Combination of versions .................................................................................................................. 1
  Use case ......................................................................................................................................... 2
  Preparing the environment ............................................................................................................ 2

**Creating the project** .................................................................................................................. 3
  Developer Studio 7.0 ..................................................................................................................... 3
  Developer Studio 7.1 ..................................................................................................................... 9

**Implementing an onSubmit class** ............................................................................................ 17
  The code ......................................................................................................................................... 17
  Developer Studio 7.0 ................................................................................................................... 18
  Developer Studio 7.1 ................................................................................................................... 22

**Referencing the class from the task configuration** ..................................................................... 29
  Testing the solution ....................................................................................................................... 30
Introduction

The Identity Center Extension Framework provides a way to extend an Identity Center configuration with the features from a separately developed Java class, and it is utilized when executing a task in the Identity Management User Interface. The extension class may be configured for any User Interface task in the Identity Center. It allows customers to write their own code for validating attributes entered by the user, using the SAP NetWeaver Identity Management User Interface. For example, when the user selects "Submit" on the task, the built-in validation is performed first. Then the user code is called and can perform additional validation. This code may return "OK", or return an error message, which is presented to the user. Only when the user code returns "OK" the data is stored in the identity store.

In the current implementation there is one possible extension point, onSubmit, corresponding to the extension class method onSubmit. The onSubmit method is executed just before the task itself is executed. It provides a way to implement input attribute validation for a given task.

The implementation process looks like this:
1. The Java class is implemented as a project in the SAP NetWeaver Developer Studio.
2. Deploy the class on the SAP NetWeaver running the User Interface.
3. Include the reference to the class from the task configuration in the Management Console.

Combination of versions

The Identity Management User Interface can be deployed on the following versions of SAP NetWeaver:

- SAP NetWeaver AS Java as of Release 7.0
- EHP 1 for SAP NetWeaver CE 7.1

The following versions of SAP NetWeaver Developer Studio are supported:

- SAP NetWeaver Developer Studio 7.0
- SAP NetWeaver Developer Studio 7.1

The following combinations are legal:

<table>
<thead>
<tr>
<th>Developer Studio 7.0</th>
<th>Developer Studio 7.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP NW 7.0</td>
<td>X</td>
</tr>
<tr>
<td>SAP NW 7.1</td>
<td>X</td>
</tr>
</tbody>
</table>

In case the procedure is different for the different combinations, there are separate sections for each of them.

In the document, the procedures are described for combinations SAP NW 7.0/Developer Studio 7.0 and SAP NW 7.1/Developer Studio 7.1. In other words, the document describes procedures where Developer Studio 7.0 is used only towards SAP NW 7.0, and likewise Developer Studio 7.1 only towards SAP NW 7.1.

The combination SAP NW 7.0/Developer Studio 7.1 can also be used, but this procedure is out of scope for this document.
Use case

In this document a use case is described, for implementing the extension class and method onSubmit:

- In this use case, the onSubmit method makes sure that the telephone number (mobile) of a user that is about to be created (by a "Create" task) is prefixed with the international country code and is valid. Here, the international country code prefix is, for simplicity's sake, the hardcoded value "+47". An alternative to using a hardcoded value is to generate the international country code based on a location parameter, by reading the country information from e.g. MX_ADDRESS_COUNTRY attribute.

Preparing the environment

To be able to implement an extension class, the following is needed:

- The Identity Management UI must be deployed on a server running one of the mentioned versions of SAP NetWeaver.
- One of the mentioned versions of SAP NetWeaver Developer Studio must be installed.
- The Identity Management Extension Framework Library (sap.com~tc~idm~jmx~extfwk~default.jar) must be present. The file is located in DesignTimeComponents\Misc on the install set.
Creating the project

This section describes how you create the project in Developer Studio. There are separate sections for Developer Studio 7.0 and 7.1. In this document, the procedures are described for combinations SAP NW 7.0/Developer Studio 7.0 and SAP NW 7.1/Developer Studio 7.1 in separate sections. The combination SAP NW 7.0/Developer Studio 7.1 can also be used, but this procedure is not described in this document.

Developer Studio 7.0

Creating the project with Developer Studio 7.0 involves the following steps:

- Creating the project: Enterprise Application Project (EAR) and Web Module Project (WAR).
- Adding the Identity Management Extension Framework Library (file sap.com~tc~idm~jmx~extfwk~default.jar located in DesignTimeComponents\Misc on the install set) to the Web Module Project classpath.
- Adding the Web Module Project to the EAR project.

Creating the project

1. Start Developer studio.
2. Create a project of type J2EE/Enterprise Application Project.
3. Choose "Next >".

![Enterprise Application Project dialog box](image)

Enter a name for the project, for instance `IdMFwk_EAR`.

4. Choose "Finish".

5. Create new project of type J2EE/Web Module Project.

![Select New Project dialog box](image)
6. Choose "Next >".

   Enter a name for the project, for instance *IdMFwk_WAR*.

7. Choose "Finish".
Adding the Identity Management Extension Framework Library to the Web Module Project classpath

After creating the EAR and WAR projects, the IdM Extension Framework Library (interfaces) needs to be added to the Web Module Project classpath. Do the following:

1. Select the Web Module Project "IdMFwk_WAR" and choose "Properties" from the context menu.

Select "Java Build Path", and then select the "Libraries" tab.
2. Choose "Add External JARs…".

Select the file sap.com~tc~idm~jmx~extfwk~default.jar.

3. Choose "OK".

4. Set a reference to the Identity Management application from the Enterprise Application Project. Open the file application-j2ee-engine.xml.
Create new reference by selecting "References" and "Add".
Fill in the following information:

**Reference target**
Enter "tc/idm/jmx/app".

**Reference type**
Select "hard".

**Reference target type**
Select "application".

**Provider name**
Enter "sap.com".

5. Save the file.

6. Alternatively, this can be done by adding the following section to the file `application-j2ee-engine.xml`:

```xml
<reference reference-type="hard">
  <reference-target
    provider-name="sap.com"
    target-type="application">tc/idm/jmx/app</reference-target>
</reference>
```

7. Save the file.

**Adding the Web Module Project to the EAR project**

To add the Web (WAR) Module to the EAR project, do the following:

1. Select the Enterprise Application Project (EAR).

2. Add the Web project to the EAR project by choosing "Add modules" from the context menu.

Select the WAR project (IdMFwk_WAR).

3. Choose "OK".
Creating the project

Developer Studio 7.1

Creating the project with Developer Studio 7.1 involves the following steps:

- Creating the project: Enterprise Application Project (EAR) and Dynamic Web Project (WAR). The Dynamic Web Project is added to the EAR project when created.
- Adding the Identity Management Extension Framework Library (file sap.com~tc~idm~jmx~extfwk~default.jar located in DesigntimeComponents\Misc on the install set) to the Dynamic Web Project classpath.

Creating the project

1. Start Developer studio.
2. Create a project of type J2EE/Enterprise Application Project.
3. Choose "Next >".

Enter a name for the project, for instance IdMFwk_EAR.
Make sure that "Use default" is selected.
Make sure that "SAP Libraries" is selected in the "Target Runtime" field.
Select "SAP EAR Java EE 5 Project" in the "Configurations" field.

4. Choose "Next >".
Make sure that the following project facets are selected:

- Create SAP JMS resources descriptor (Version 1.0)
- EAR (Version 5.0)
- SAP Specific Ear Module (Version 5.0)

5. Choose "Next >" and then "Finish".

7. Choose "Next >".

Enter a name for the project, for instance *IdMFwk_WAR*.

Make sure that "Use default" is selected.

Select "SAP Web Java EE 5 Project" in the "Configurations" field.

Select "Add project to an EAR" and enter the EAR project name (*IdMFwk_EAR*) in the "EAR Project Name" field.
8. Choose "Next >".

Make sure that the following project facets are selected:
- Dynamic Web Module (Version 2.5)
- Java (Version 5.0)
- SAP Specific Web Module (Version 2.5)

9. Choose "Next >"

Make sure that context root is \textit{IdMFwk\_WAR}.

10. Choose "Finish".
Adding the Identity Management Extension Framework Library to the Dynamic Web Project classpath

After creating the EAR and WAR projects, the IdM Extension Framework Library (interfaces) needs to be added to the Dynamic Web Project classpath. Do the following:

1. Select the Dynamic Web Project "IdMFwk_WAR" and choose "Properties" from the context menu.

Select "Java Build Path", and then select the "Libraries" tab.
2. Choose "Add External JARs…".

Select the file sap.com~tc~idm~jmx~extfwk~default.jar.

3. Choose "OK".

4. Set a reference to the Identity Management application from the Enterprise Application Project. Open the file application-j2ee-engine.xml placed in EarContentMETA-INF of the IdMFwk_EAR project.

Add the following section to the file:
<reference reference-type="hard">
  <reference-target
    provider-name="sap.com"
    target-type="application">tc/idm/jmx/app</reference-target>
</reference>
Implementing an onSubmit class

This section describes the necessary steps to implement the extension class. The javadoc extfwk_javadoc.zip, located in DesigntimeComponents\Misc on the install set, contains information about the classes.

Implementing the class in both Developer Studio 7.0 and Developer Studio 7.1 involves the following steps:

- Creating the class.
- Implementing the class method onSubmit.
- Adding the onSubmit class to the runtime part of the Identity Management User Interface.
- Building and deploying the application.

The code

The following code is used to implement the class method onSubmit according to given use case (see section Use case on page 2):

```java
package testcustomer.idm.impl;
import java.util.Locale;
import com.sap.idm.extension.IdMExtensionException;
import com.sap.idm.extension.TaskProcessingAdapter;
import com.sap.idm.extension.api.IdMSubmitData;
import com.sap.idm.extension.api.IdMValueChange;

public class MyOnSubmit extends TaskProcessingAdapter{
    /**
     * This onSubmit method makes sure that the mobile phone
     * number is prefixed with the international country code
     * if it is missing (here "+47").
     */
    public IdMValueChange[] onSubmit(Locale locale, int subjectMSKEY, int objectMSKEY, Task task, IdMSubmitData validate)
            throws IdMExtensionException {
        IdMValueChange[] changes = validate.getChangeList();
        for (int i = 0; i < changes.length; i++) {
            IdMValueChange aIdmValueToFix = changes[i];
            String aAttr = aIdmValueToFix.getAttributeName();
            if (aAttr.equalsIgnoreCase("MX_MOBILE_PRIMARY")) {
                String aValue = aIdmValueToFix.getAttributeValue().trim();
                // Add international prefix if missing
                if (!aValue.startsWith("+")) {
                    aValue = "+47 " + aValue;
                    aIdmValueToFix.setAttributeValue(aValue);
                }
            }
            aIdmValueToFix.setAttributeValue(aValue.trim());
        }
        return changes;
    }
}
```

The existing sample code needs to be extended with the bold text above. For the onSubmit method the input IdMSubmitData validate is actually the IdMValueChange[], an array consisting of all changes made to the original data. The expected results are:

- A list of changes (the optionally manipulated input data).
- Null, meaning that no changes are applied to the original data.
- Exception string displayed in the User Interface when validation was not successful.
Note:
Adding an attribute using the onSubmit method does not require the attribute to be a part of the task schema.

Developer Studio 7.0

To implement the class in Developer Studio 7.0, complete the steps previously listed (on page 17).

Creating the class

Create a new Java Class implementing the TaskProcessingAdapter interface:
1. Select the "source" node of the project and select "New/Java class…":

![New Java Class dialog](image)

Fill in the following information:
Make sure that the source folder is IdMFwkWARsource.
Enter a name for the package, in this example testcustomer.idm.impl.
Enter a name for the class, here MyOnSubmit.
Select the name of the class you want to extend, com.sap.idm.extension.TaskProcessingAdapter.

2. Choose "Finish".
Implementing the onSubmit method

To implement the class method onSubmit, do the following:

1. Extend the sample code with the bold text presented in the section The code on page 17. Make sure that all imports are present before saving the class.
2. Save the class.

Adding the onSubmit class to the runtime part of Identity Management User Interface

Create a servlet that is used to initiate the class.

1. Create a servlet by selecting the "IdMFwk_WAR" project and choose "New/Servlet…":
   - Enter a "Servlet Name", here IdMRegisterServlet.
   - Select a "Servlet Package", testcustomer.idm.impl.
   - Select "init()".
2. Choose "Finish".
3. Register your onSubmit by adding the following line to the init method of your servlet:
   ```java
   IdMFactory.getInstance().registerTaskProcessingInterface(new MyOnSubmit());
   ```
4. If necessary, import the com.sap.idm.extension.IdMFactory class.
5. Make sure the servlet is invoked on startup. Open the `web.xml` file in the WAR project and navigate to the "Web Objects" tab:

![Image of web.xml configuration](image)

Make sure that "load on startup" has the value 1 in "IdMRegisterServlet".

6. Save the file.
Building and deploying the application

The last step is to build and deploy the application containing the extension class:

1. If necessary, you must define a J2EE server by opening Window/Preferences... and configuring SAP J2EE engine:

   Specify the host and port number of the J2EE engine.

2. Choose "OK".

3. Build the application by selecting the "IdMFwk_EAR" project and choosing "Build Application Archive" from the context menu.

4. Deploy your application by selecting the built application archive (typically IdMFwk_EAR.ear) and choosing "Deploy to J2EE engine" from the context menu.
Developer Studio 7.1

To implement the class in Developer Studio 7.0, complete the steps previously listed (on page 17).

Creating the class

Create a new Java Class implementing the TaskProcessingAdapter interface:

1. Select the "source" node of the project and select "New/Java class":

   Fill in the following information:

   Make sure that the source folder is IdMFwk_WAR\src.
   Enter a name for the package, in this example testcustomer.idm.impl.
   Enter a name for the class, here MyOnSubmit.
   Select the name of the class you want to extend, com.sap.idm.extension.TaskProcessingAdapter.

2. Choose "Finish".
Implementing the onSubmit method

To implement the class method onSubmit, do the following:

1. Extend the sample code with the bold text presented in the section The code on page 17. Make sure that all imports are present before saving the class.
2. Save the class.

Adding the onSubmit class to the runtime part of Identity Management User Interface

Create a servlet that is used to initiate the class.

1. Create a servlet by selecting the "IdMFwk_WAR" project and choose "New/Servlet":

Select a "Java package", testcustomer.idm.impl.
Enter a "Class name", here IdMRegisterServlet.
2. Choose "Next >".

3. Choose "Next >".

Select "init()".

4. Choose "Finish".

5. Register your onSubmit by adding the following line to the init method of your servlet:
6. If necessary, import the `com.sap.idm.extension.IdMFactory` class.
7. Make sure the servlet is invoked on startup. Open the `web.xml` file in the WAR project:

Add the following line to the file as shown above:

```
<load-on-startup>1</load-on-startup>
```

8. Save the file.
Building and deploying the application

The last step is to build and deploy the application containing the extension class:

1. If necessary, you must define a J2EE server by opening Window/Preferences..., and configuring SAP AS Java engine:

2. Choose "Add new SAP System".

Specify the host and the instance number of the engine.
3. Choose "OK".

Make sure that "Default System" is selected.

4. Choose "OK".
5. Create an export by selecting the "IdMFwk_EAR" project and choosing "Export" from the context menu.

Make sure that "EAR application" (here IdMFwk_EAR), "Destination" and "Overwrite existing file" are selected.

6. Choose "Finish".

7. Deploy your application by selecting the created export file and choosing "Deploy" from the context menu.

Logon dialog box appears. Enter administrator credentials for the engine.

8. Choose "OK".

Information that the deployment finished successfully is given.

9. Choose "OK" to close the dialog box.
Referencing the class from the task configuration

The class is referenced from the task configuration in the Management Console.

1. View the properties of the task and select the "Result handling" tab:

   ![Management Console screenshot]

   Enter the name of the extension class in the "On task submit" field.

2. Choose "Apply".

   The next time the task is executed, the submitted values are validated using the mechanisms in the class.
Testing the solution

To test the implemented code, do the following:

1. Open the Identity Management User Interface and the task for creating a user (here task Create User) and fill in the necessary information about the user:

Here, a user Diana Anderson with a unique ID "i10" and a mobile number "90119190" is specified (department "DEVELOPMENT" and a primary e-mail address "alternative@demo.com").

Note:
The attribute MX_MAIL_PRIMARY is here defined as a multi-valued attribute with "SingleLine" presentation and the number of rows in the table set to "5".

2. Choose "Save" to execute the task and create the specified user, then close the task.
3. Find the user ("i10") and display its details.

Observe that the primary mobile number for this user is prefixed with the international country code "+47", as specified in the `onSubmit` method.