A MITEL
PRODUCT
GUIDE

Unify OpenScape UC Application V10

SPML SDK

Programming Guide
08/2024

A31003-S50A0-R101-08-7620

Mitel

Notices

The information contained in this document is believed to be accurate in all respects but is not warranted by Mitel Europe Limited. The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes. No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Mitel Networks Corporation.

Trademarks

The trademarks, service marks, logos, and graphics (collectively “Trademarks”) appearing on Mitel’s Internet sites or in its publications are registered and unregistered trademarks of Mitel Networks Corporation (MNC) or its subsidiaries (collectively “Mitel), Unify Software and Solutions GmbH & Co. KG or its affiliates (collectively “Unify”) or others. Use of the Trademarks is prohibited without the express consent from Mitel and/or Unify. Please contact our legal department at iplegal@mitel.com for additional information. For a list of the worldwide Mitel and Unify registered trademarks, please refer to the website: http://www.mitel.com/trademarks.

Contents

1 History of Changes

Date	Changes	Reason
30-01-2020	Initial creation	
01-04-2020	Update for LDAPS configuration & activation	UCBE-22850
12-01-2023	Updated chapter: - Chapter 3.4.3 User	UCBE-31816
10-03-2023	Updated chapter: - Chapter 3.4.3 User	UCBE-32160
21-07-2023	Updated chapter: - Chapter 3.4.3 User	UCBE-32968
14-12-2023	Updated: - Chapter 3.4.3 User - Chapter 3.4.4 Contact - Chapter 3.4.17 ExternalIdentifier	UCBE-33660
15-01-2024	Updated: - Chapter 3.3.3 Import Use Case: Modifying a User	DOCLOC-8107
22-03-2024	Updated: - Chapter 3.4.3 User	UCBE-34309
20-05-2024	Updated: - Chapter 3.4.3 User	UCBE-34688

2 Introduction

The Service Provisioning Markup Language (SPML) is an XML-based framework for exchanging user, resource and service provisioning information between cooperating systems. The Service Provisioning Markup language is the open standard for the integration and interoperation of service provisioning requests. SPML is based on the concepts of Directory Service Markup Language (DSML).

This document describes how to use the Symphonia import export service. Symphonia is the SOA framework of Unify GmbH & Co. KG upon which OpenScape UC application is build. The import export service is a part of the overall Symphonia provisioning infrastructure and provides means to:

- Initially load the supported domain entities (e. g. users, phones, etc.) when setting up a system from scratch
- Update already provisioned entities (e. g. change user attributes)
- Add new entities to the already existing entity population (e.g. add a new phone)

IMPORTANT: Please note that the import export service is complementary to the Symphonia backup infrastructure but not designed to provide an alternative backup solution to e. g. fully restore a system. Not all entities of a system currently are importable (e. g. domains must be created by an administrative action) respectively exportable (e. g. passwords must not be exported).

2.1 Addressees

This documentation gives detailed coding information for application developers wanting to implement Java applications accessing the SPML/SOAP interface of the OpenScape UC Application (see Provisioning by SPML/SOAP Interface on page 9).

Provisioning LDAP Directories on page 85 gives information about editing an configuration XML file in order to provision LDAP directories to the OpenScape UC Application. If extended features are to be implemented, additional information is supplied for Java application developers.

The reader needs to have a basic understanding of the following items:

- OpenScape UC Application
- SPML
- For Provisioning by SPML/SOAP Interface on page 9 only:
  - Java
- For Provisioning LDAP Directories on page 85 only:
  - LDAP
  - XML
  - Java, if extended features should be realized

2.2 Document Conventions

The following conventions are used in the manual to clearly distinguish the various types of information.

1) Single operating instruction steps are numbered.

- Enumerations are indicated with dots.

NOTICE: A note is inserted in the text to draw your attention to a feature or to provide information that facilitates working with the program.

IMPORTANT: Signals information of high priority. The corresponding details must be heeded to avoid damages to the system or loss of data.

Indications of operating controls, such as buttons and window captions, are emphasized in boldface, for example, **Login Options**.

2.3 Acronym Directory

The following table lists in alphabetic sequence the most important acronyms used.

Acronym	Description
BCOM	**B**asic **C**ommunication
BGL	**B**usiness **G**roup **L**ine
CMP	**C**ommon **M**anagement **P**latform
CSV	**C**omma **S**eparated **V**alues
DN	**D**istinguished **N**ame
DSML	**D**irectory **S**ervice **M**arkup **L**anguage
GNF	**G**lobal **N**umber **F**ormat
HTTPS	**H**yper**t**ext **T**ransfer **P**rotocol **S**ecure
IDE	**I**ntegrated **D**evelopment **E**nvironment
JDK	**J**ava **D**evelopment **K**it
JRE	**J**ava **R**untime **E**nvironment
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
ONS	**O**ne **N**umber **S**ervice
PNPextension	**P**rivate **N**umber **P**lan Extension
RDN	**R**elative **D**istinguished **N**ame

Acronym	Description
RPM	**R**ed **H**at **P**ackage **M**anager
SDK	**S**oftware **D**evelopment **K**it
SLES	**S**use **L**inux **E**nterprise **S**erver
SPML	**S**ervice **P**rovisioning **M**arkup **L**anguage
SOAP	**S**imple **O**bject **A**ccess **P**rotocol
SSL	**S**ecure **S**ocket **L**ayer
SSO	**S**ingle **S**ign-**O**n
TS	**T**arget **s**ystem
XML	E**x**tensible **M**arkup **L**anguage
URN	**U**niform **R**esource **N**ame
XSI	**X**ML **S**chema **I**nstance

3 Provisioning by SPML/SOAP Interface

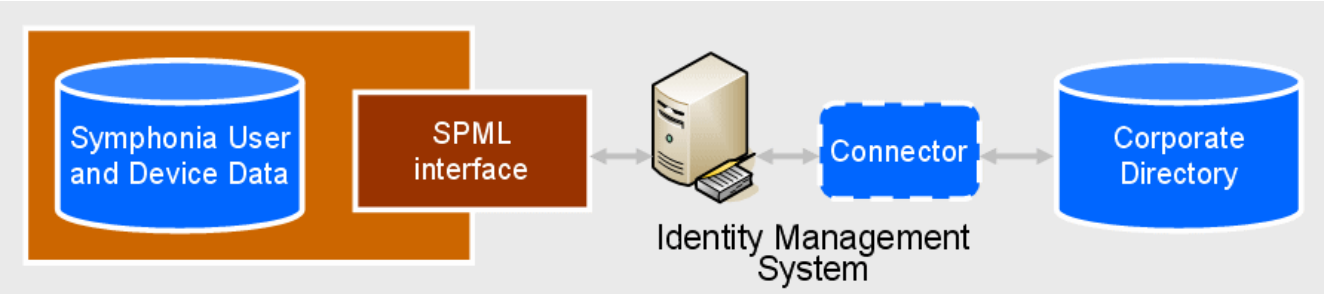
3.1 Overview

3.1.1 General

The OpenScape UC Application offers an SPML over SOAP interface for provisioning. Active Directory is supported with this scenario as well.

The OpenScape UC Application does not provide an out of the box connector to the corporate directory. The connector is also responsible for the correct attribute mapping.

NOTICE: Specific connectors (for example Lotus Domino, Active Directory etc.) can be provided by the Professional Services on request.



Initial or general system configuration (for example the CSTA link configuration) is not possible via this interface.

The OpenScape UC Application SPML interface does not extend to the provisioning of users into the communication system or Voice over IP (VOIP) system.

NOTICE: The OpenScape Voice System has its own mass provisioning toolset. Further information can be found at http://www.unify.com/developerportal/Resource%20Center/OpenScape-Voice.aspx.

NOTICE: Importing and exporting data by using CSV files is described in the administrator documentation *OpenScape UC Application Configuration and Administration*.

The interface supports SMPL V1.0. It is a request/reply based interface as defined by SPML V1.0.

3.1.2 Access to the SPML Interface

Depending on the server configuration the SPML interface is reachable via an HTTP interface at

```
http://<IP address>:8099/symphonia-spmlresponder/services/SpmlSoapService
```

If you use HTTPS, use

```
https://<IP address>:9943/symphonia-spmlresponder/services/SpmlSoapService
```

instead. `<IP address>` is the application computer of the OpenScape UC Application. The application computer is the computer where the backend services of the OpenScape UC Application have been installed.

The HTTP basic authentication mechanism has to be used (user name and password).

A well formed SPML document is sent via this interface to the OpenScape UC Application executing the specified operation (creation, modification or deletion).

3.1.3 Supported Objects and Requests

This section lists the objects that can be manipulated by the SPML interface and the SPML requests that can be executed using the SPML interface. The objects to be used to create the supported objects are described in Object Reference on page 32.

Table 1: Supported Requests

Object	Source System/Destination System													
	Any except LDAP Directory							LDAP Directory						
	Add	Modify	Delete	Search	Batch	Status	Cancel	Add	Modify	Delete	Search	Batch	Status	Cancel
Domain	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
User	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Contact	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Phone	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Phone4K	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
AllocatedPhone	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
VoiceMail	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
DPPublic	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
DPPrivate	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
NPPrivate	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
NPPublic	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗

Object	Source System/Destination System													
	Any except LDAP Directory							LDAP Directory						
	Add	Modify	Delete	Search	Batch	Status	Cancel	Add	Modify	Delete	Search	Batch	Status	Cancel
NPUnknown	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Switch8k	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Switch4k	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
BCFederation	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
ExternalIdentifier	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
QueueDevice	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Role	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗

3.2 Preconditions

The following objects have to be configured via the CMP before running provisioning:

Mandatory

- User profiles
- Predefined profiles exist for example for the system and customer administrator. Typical users do not need a configured profile. See the administrator documentation *OpenScape UC Application Configuration and Administration*, section “Creating a User Profile” for details.
- If voicemail systems must be assigned to users, one or more voicemail systems must be defined via the CMP. See the administrator documentation *OpenScape UC Application Configuration and Administration*, section “Configuring a Voice Portal” or section “Configuring the Connection to OpenScape Xpressions” for details.
- If queue devices are to be assigned to users, one or more queue devices must be defined via the CMP. You can do so by selecting the Administrator user, opening **Configuration** > **Unified Communications** > **Accounts** > **List** > **<User>** > **Resources** > **Queue** > **Number** and adding the queue devices you wish to use.

3.3 Code Development

3.3.1 Preparations

Installation of Eclipse IDE

1) Execute the following command, to check whether JRE 1.5.x or higher or JDK 1.5.x or higher is installed.

```
java -version
```

A succesful output is for example:

```
Java version "1.6.0_17"
Java(TM) SE Runtime Environment (build 1.6.0_17-b04)
Java HotSpot(TM) Client VM (build 14.3-b01, mixed mode,
 sharing)
```

2) Open `http://www.eclipse.org/downloads/`.

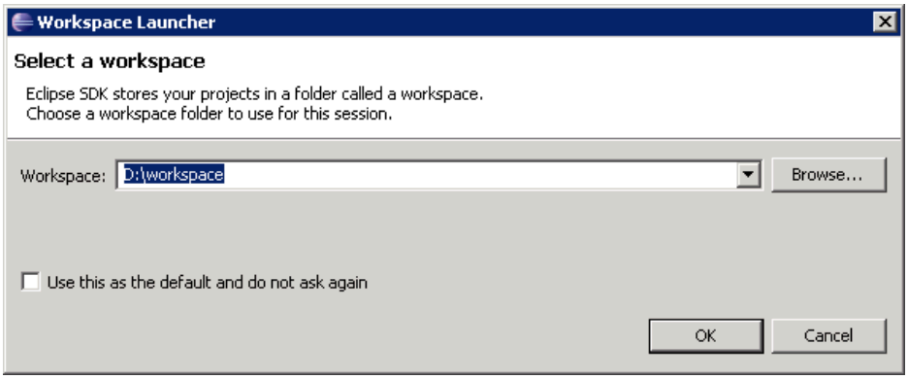
3) Download the Eclipse Classic IDE. Eclipse Classic 3.6.0 (`eclipse-SDK-3.6.win32.zip`) is sufficient and is used for this documentation. Version 3.6.0 is also called Helios.

4) Decompress the ZIP file.

5) Install Eclipse.

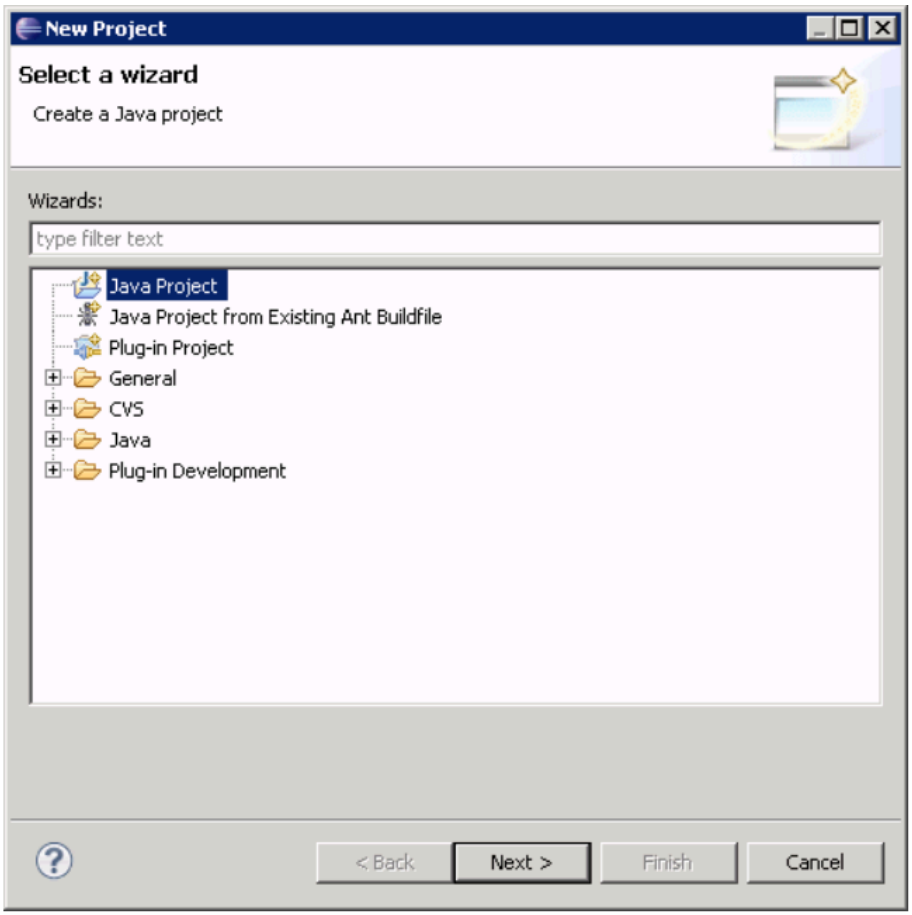
Project Creation

1) Open **Start** > **Programs** > **Eclipse 3.6.0**.

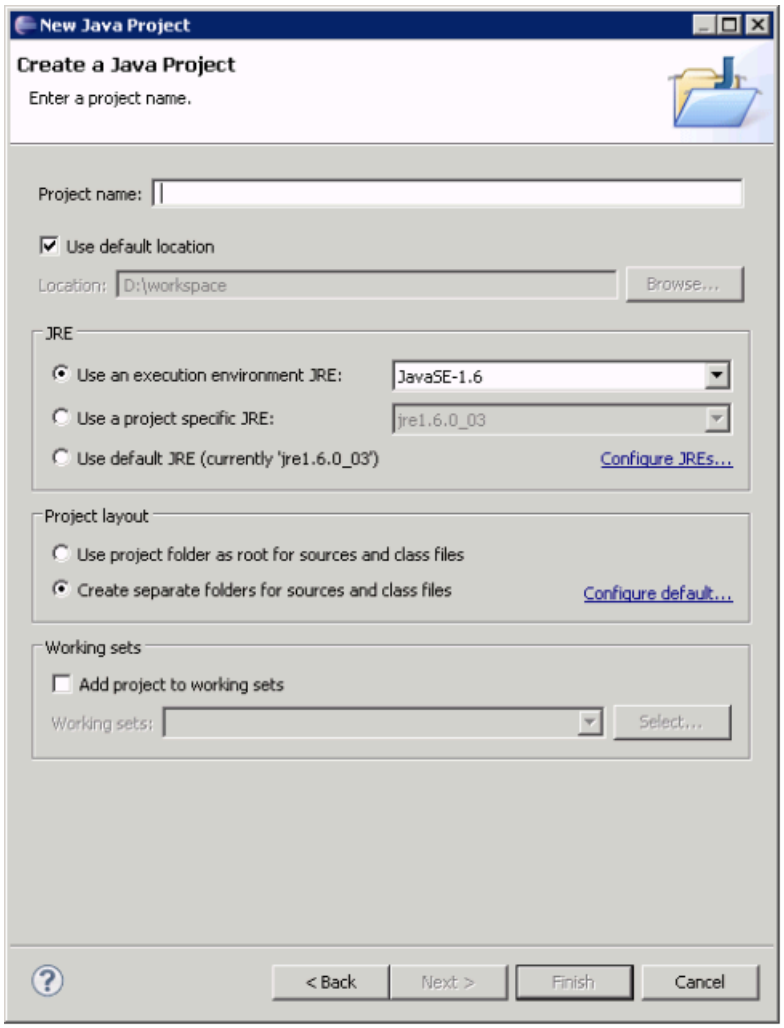


2) Select a directory as Eclipse workspace.

3) Open **File** > **New** > **Project...** > **Java Project**.



4) Click the **Next** button.

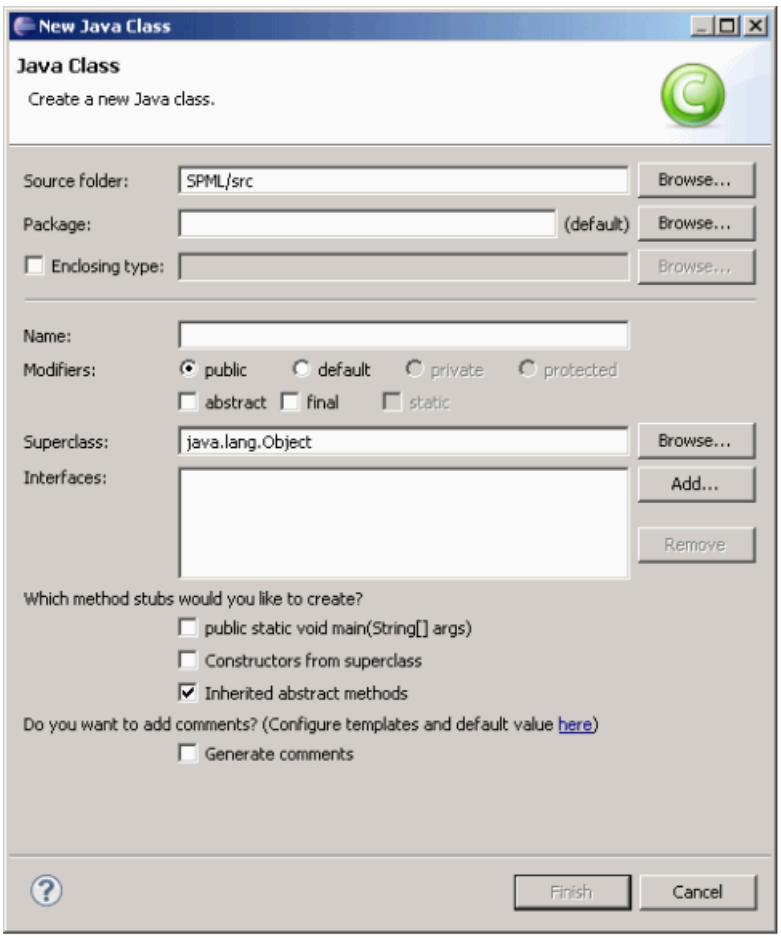


5) Enter for example `SPML` into the **Project name** field.
6) Check whether the correct Java runtime environment has been selected in the **Use an execution environment JRE** field.
7) Click the **Next** button.
8) Check the **Default output folder** field's value, for example `SPML/bin`.
9) Click the **Finish** button.

   Subdirectories in `<workspace>` are created.

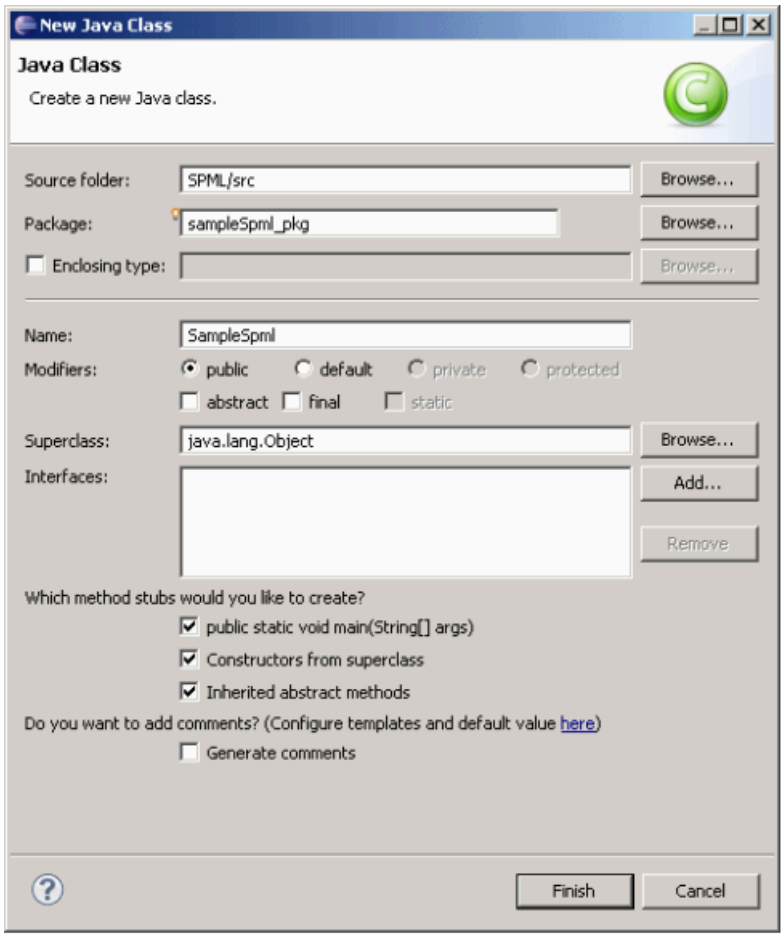
First Class Creation

1) Open **File** > **New** > **Class**.



2) Enter for example `sampleSpml_pkg` into the **Package** field.
3) Enter the class name, for example `SampleSpml`, into the **Name** field.

4) Make sure the **public static void main(String[] args)** radio button is activated.

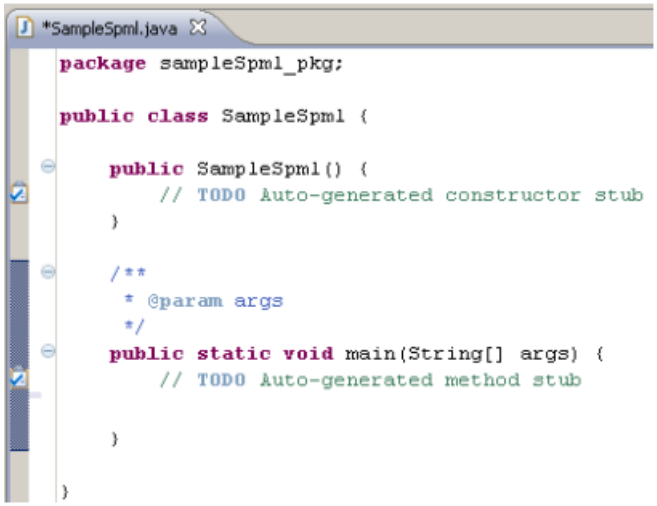


5) Click the **Finish** button.

   The `<workspace>\SPML\src\sampleSpml_pkg\SampleSpml.java` file is created.

6) Open **Window** > **Show** > **View** > **Package Explorer**.

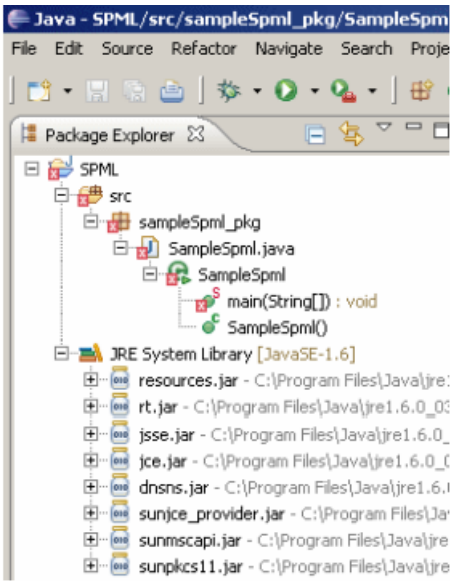
7) Double click the `SampleSpml` class. The source code of the class is shown in the right pane.



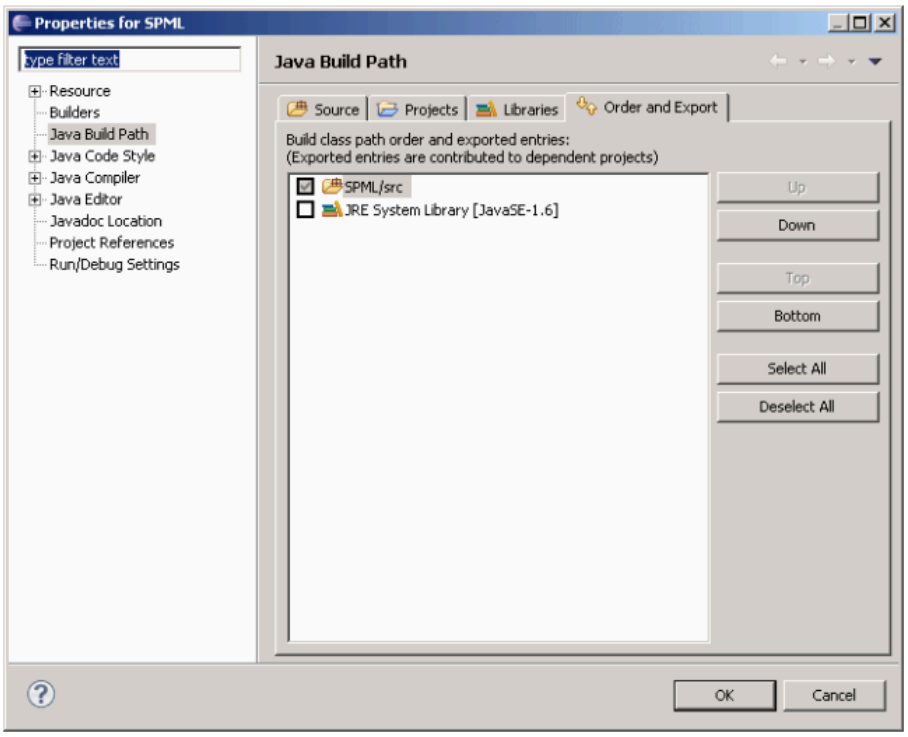
Additional JAR Files

1) Create the following directory:
   `<workspace>\SPML\lib`
2) Copy the files
   - `commons-codec-1.4.jar`
   - `commons-httpclient-3.0.1.jar`
   - `commons-logging-1.0.3.jar`

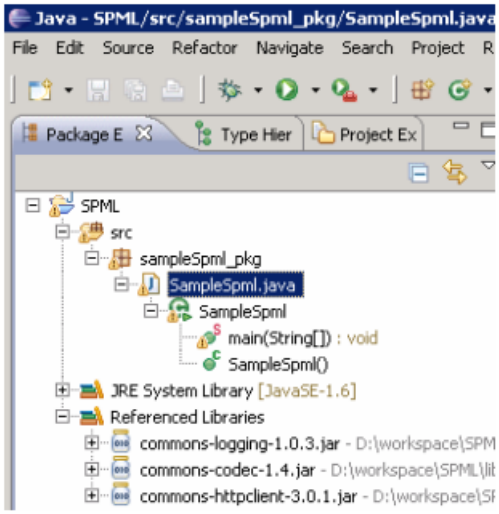
   into this directory. You find these files in the ZIP file containing this documentation.
3) Select the package name in the Package Explorer with the right mouse button.



4) Select **Build Path** > **Configure Build Path...**.



5) Click the **Libraries** tab.
6) Click the **Add External JARs...** button.
7) Select the JAR files copied in step 23.
8) Click the **Open** button.
9) Click the **OK** button.



3.3.2 Import Use Case: Adding a User

Execute the following steps in the source code editor pane to add a new user to the OpenScape UC Application via the SPML interface.

1) Start with the package and import commands for the communication between the client and the OpenScape UC Application.

```
package sampleSpml_pkg;
import org.apache.commons.httpclient.Credentials;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.StringRequestEntity;
```

2) Start coding the main method. SPML always uses port 8099. - `userNameWithDomain` is used for the user accessing the SPML interface. Be sure not to miss the “@” string followed by the domain (usually `system`) at the end of `userNameWithDomain`. `password` is his password. `host` is the computer the OpenScape UC Application resides on.

```
public class SampleSpml
{
public static void main(String[] args)
{
int port = 8099;
String url = "/symphonia-spmlresponder/services/SpmlSoapService";
String userNameWithDomain = "user01@system";
String password = "passwd01";
String host = "host01";
```

3) The variable `spmlRequest` contains the SPML command to be executed. `<addRequest>` indicates that a new object is to be added. `<id>` identifies the object to be added. `<attr>` within `<identifierAttributes>` indicates that a user is to be added since its `name` attribute has the value `objectclass` and it contains a `<value>` having the value `User`. The second `<attr>` within `<identifierAttributes>` indicates that the domain with the value `system` is used for identifying the user as well.

```
String spmlRequest= "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
"<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
"<soapenv:Body>" +
"<addRequest xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<identifier xmlns=\"urn:oasis:names:tc:SPML:1:0\"
type=\"urn:oasis:names:tc:SPML:1:0#DN\">" +
"<id xmlns=\"urn:oasis:names:tc:SPML:1:0\">Peter Petersson</id>" +
"<identifierAttributes xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"objectclass\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">User</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"domain\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">system</value>" +
"</attr>" +
"</identifierAttributes>" +
"</identifier>" +
```

4) The following `<attr>` within the `<attributes>` describe which properties of this user should be set to which values. See Section 2.4.3, “User”, on page 42 for details about the properties that can be set and the meanings they have.

```
"<attributes xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"id\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Peter Petersson</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"domain\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">system</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"defaultLocale\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">en_US</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"displayName\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Peter Petersson display name</
value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"pin\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">12345678</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"homeTimeZone\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Europe/Berlin</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"pwdNeedChange\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">0</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"password\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">X1ksjkadkfls_</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"assignedProfile\">" +
```

```
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Professsional:OpenScape UC
 App</
value>" +"
</attr>" +
"</attributes>" +
```

5) Be sure to close the SPML request properly.

```
"</addRequest>" +
"</soapenv:Body>" +
"</soapenv:Envelope>";
```

> **NOTICE:** All SPML requests that are described in detail in this documentation are available as XML files in the `spml_requests` directory in the ZIP file this documentation is part of.

6) The user name, the domain, the computer name of the OpenScape UC Application, the port to be used etc. have been defined in step 2 on page 21. Hand them over to the SPML client.

```
try
{
HttpClient client = new HttpClient();
Credentials defaultcreds = new UsernamePasswordCredentials(userNameWithDomain,
password);
client.getState().setCredentials(new AuthScope(host, port, AuthScope.ANY_REALM),
defaultcreds);
client.getParams().setAuthenticationPreemptive(true);
client.getHostConfiguration().setHost(host, port, "http");
postMethod.addRequestHeader("Content-Type", "text/xml; charset=UTF-8");
postMethod.addRequestHeader("Cache-Control", "no-cache");
postMethod.addRequestHeader("SOAPAction", "\"\"");
postMethod.addRequestHeader("ContentLength",Integer.toString(spmlRequest.length()));
postMethod.setRequestEntity(new StringRequestEntity(spmlRequest));
```

7) Send the SPML request.

```
int exeResult = client.executeMethod(postMethod);
```

8) Let's do some error handling.

```
if(HttpStatus.SC_NOT_IMPLEMENTED == exeResult)
{
System.err.println("The Post method is not implemented by this URI: " +
postMethod.getURI());
}
```

System.err.println("The Post method is not implemented by this URI: " + postMethod.getURI());
}

9) If everything went well, give some output concerning the response of the OpenScape UC Application.

```
else
{
BufferedReader br = new BufferedReader(new
InputStreamReader(postMethod.getResponseBodyAsStream()));
String readLine;
while((readLine = br.readLine()) != null)
{
```

```
System.err.println(readLine);
}
}
}
```

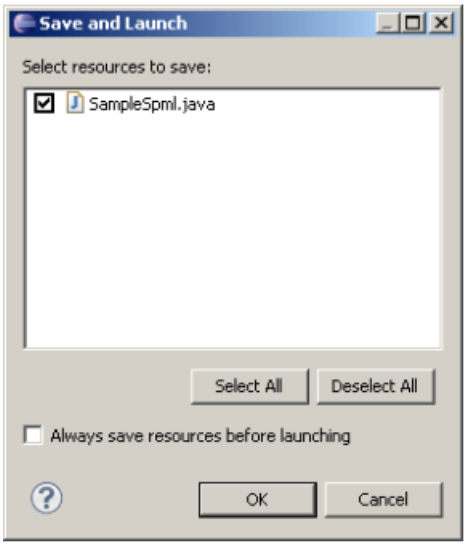
10) Do some error handling again.

```
catch(Exception e)
{
System.err.println("Exception caught: " + e);
}
```

11) Release the connection to OpenScape UC Application.

```
finally
{
postMethod.releaseConnection();
}
}
}
```

12) Click **Run** > **Run As** > **Java Application**.



13) Click the **OK** button.

14) The following output is created at the SPML client when everything went well.

The <addResponse> shows that an object `Peter Petersson` with the `objectclass User` and the `domain system` successfully has been created.

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:addResponse result="urn:oasis:names:tc:SPML:1:0#success"
xmlns="urn:oasis:names:tc:SPML:1:0" xmlns:ns1="urn:oasis:names:tc:SPML:1:0">
<ns1:identifier type="urn:oasis:names:tc:SPML:1:0#DN">
<ns1:id>Peter Petersson</ns1:id><ns1:identifierAttributes>
<ns1:attr name="objectclass">
<ns51:value type="string" xmlns:ns51="urn:oasis:names:tc:DSML:2:0:core">User</
ns51:value>
```

```
</ns1:attr>
<ns1:attr name="domain">
<ns52:value type="string" xmlns:ns52="urn:oasis:names:tc:DSML:2:0:core">system</
ns52:value>
</ns1:attr>
</ns1:identifierAttributes>
</ns1:identifier>
</ns1:addResponse>
</soapenv:Body>
</soapenv:Envelope>
```

15) If the output is similar to the following, check the connection between the SPML client and the OpenScape UC Application, for example remove the firewall between them or open the port 8099 between them.

```
14.10.2010 15:20:18 org.apache.commons.httpclient.HttpMethodDirector
 executeWithRetry
INFO: I/O exception (java.net.ConnectException) caught when processing request:
Connection timed out: connect
...
INFO: Retrying request
Exception caught: java.net.ConnectException: Connection timed out: connect
```

16) Open `https://<IP address>` in a browser. Substitute `<IP address>` with the IP address or the computer name of the computer the OpenScape UC Application resides on.
17) Enter the user name and password of a user having the privilege to enter the CMP. This is not necessarily the user that has been used in step 2 on page 21.
18) Open **User Management > Administration > Users & Resources**. You see the created user.
19) Click the created user to see detailed information about him.
20) Check whether the properties have been set according to the assignments made in step 4 on page 22.

3.3.3 Import Use Case: Modifying a User

IMPORTANT: Be careful when you modify attributes that might be used for linking objects to each other. See the objects overview diagram in Object Reference on page 32 to get to know which objects be linked. Example: A contact contains an `assignedSymUserIdentity` attribute having the value of the `identity` attribute of a user. If you modify the `identity` attribute of this user, the contact is not linked any more to the user. Set the `assignedSymUserIdentity` attribute of the contact to the modified value of the `identity` attribute of the user to re-establish the link. For more details about meaning of a contact, see the introduction of Contact.

The Java source code for modifying an user is the same as in Import Use Case: Adding a User on page 18 but with one exception. The variable `spmlRequest` now has a different contents.

1) The SPML code described in step 3 on page 21 is the same for modifying a user except that <addRequest has to be exchanged by <modifyRequest.

```
...
"<modifyRequest xmlns=\"urn:oasis:names:tc:SPML:1:0\">"
 +
...
```

2) The next differences start after </identifier>.

   Exchange <attributes> by <modifications>. There is one <-modifications> describing all properties of the user that should be changed. It contains one or several <modification>. One <-modification> is used to describe one property to be changed. Each <-modification> has an attribute name stating the property to be changed and a <value> containing the value the property should be set to. See Section 2.4.3, “User”, on page 42 for details about the properties that can be set and the meanings they have.

   The following example sets the default language to German, the time zone to Europe/Paris and the user’s display name to Peter Petersson - display name. The display name is the user’s name to be displayed in the CMP.

   Note that each <modification> has an attribute operation having the value replace.

```
"<modifications xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<modification xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"defaultLocale\"
operation=\"replace\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">de_DE</value>" +
"</modification>" +
"<modification xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"homeTimeZone\"
operation=\"replace\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Europe/Paris</value>" +
"</modification>" +
"<modification xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"displayName\"
operation=\"replace\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Peter Petersson display name
modified</value>" +
"</modification>" +
"</modifications>" +
```

NOTICE: If you do not want to change the value of an attribute but to remove the value of an attribute, replace operation=\”replace\” by operation=\”delete\” and leave the content of the corresponding <value> empty. Example: If you want to set the assignedVoicemail attribute empty since a user does not have any voicemail box any more, use the following code: "<modification xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"assignedVoicemail\" operation= \"delete\">" + "<value xmlns= \"urn:oasis:names:tc:DSML:2:0:core\"></ value>" +

When modifying the assignedPhone4kAltId or assignedPhone8kAltId attributes, you must also specify the

corresponding `assignedPhone4kNumber` or `assignedPhone8kNumber` respectively.

For example, to modify the `assignedPhone4kAltId` attribute which corresponds to `assignedPhone4kNumber` 302101003037, from the initial value to “My Phone”, you must do the following changes:

```
<modifications xmlns="urn:oasis:name:tc:SPML:1:0">
 <modification name="assignedPhone4kNumber" operation="replace"
 xmlns="urn:oasis:name:tc:SPML:1:0">
 <value xmlns="urn:oasis:name:tc:SPML:1:0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:nil="true">+302101003037</value>
 </modification>
 <modification name="assignedPhone4kAltId" operation="replace"
 xmlns="urn:oasis:name:tc:SPML:1:0">
 <value xmlns="urn:oasis:name:tc:SPML:1:0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:nil="true">My Phone</value>
 </modification>
 </modifications>
```

3) Be sure to close the SPML request properly.
```
"</modifyRequest>" +
"</soapenv:Body>" +
"</soapenv:Envelope>";
```
4) Click **Run** > **Run As** > **Java Application**.
5) Click the **OK** button.
6) The following output is created at the SPML client when everything went well.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
<soapenv:Body>
<ns1:modifyResponse result="urn:oasis:names:tc:SPML:1:0#success"
xmlns="urn:oasis:names:tc:SPML:1:0" xmlns:ns1="urn:oasis:names:tc:SPML:1:0"/>
</soapenv:Body>
</soapenv:Envelope>
```

7) Refresh the browser opened in step 16 on page 25. The users’s display name has changed.
8) Click the modified user. The default language and the time zone have changed.

3.3.4 Export Use Case: Retrieving a User

NOTICE: The administrator of the OpenScape UC Application cannot be retrieved by using SPML.

The Java source code for retrieving all users is the same as in Section 2.3.2, “Import Use Case: Adding a User”, on page 21 but with one exception. The `spmlRequest` variable must be modified.

1) Create a <searchRequest> and give it a requestID attribute. This is a unique identifier for this request. It will be stated in the response so that you can identify which response corresponds to which request.

```
String spmlRequest= "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
"<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
"<soapenv:Body>" +
"<spml:searchRequest xmlns=\"urn:oasis:names:tc:SPML:1:0\"
xmlns:spml=\"urn:oasis:names:tc:SPML:1:0\" " +
"xmlns:dsml=\"urn:oasis:names:tc:DSML:2:0:core\" requestID=\"searchUserId\">" +
"<spml:searchBase type=\"urn:oasis:names:tc:SPML:1:0#DN\">" +
"<identifierAttributes>" +
```

2) Now specify that you want to retrieve a user whose **id** attribute has the value sven.svensson and whose **domain** attribute has the value **system**.

```
"<spml:attr name=\"objectclass\">" +
"<dsml:value>User</dsml:value>" +
"</spml:attr>" +
"<spml:attr name=\"domain\">" +
"<dsml:value>system</dsml:value>" +
"</spml:attr>" +
"<spml:attr name=\6"id\">" +
"<dsml:value>sven.svensson</dsml:value>" +
"</spml:attr>" +
"</identifierAttributes>" +
"</spml:searchBase>" +
```

3) Be sure to close the SPML request properly.

```
"</spml:searchRequest>" +
"</soapenv:Body>" +
"</soapenv:Envelope>";
```

4) Click **Run** > **Run As** > **Java Application**.
5) Click the **OK** button.
6) The following output at the SPML client is an example when everything went well.

   The requestID of the request is repeated.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:searchResponse requestID="searchUserId"
result="urn:oasis:names:tc:SPML:1:0#success" xmlns="urn:oasis:names:tc:SPML:1:0"
```

7) The search filter is repeated first.

```
<ns1:searchResultEntry>
<ns1:identifier type="urn:oasis:names:tc:SPML:1:0#DN">
<ns1:id>sven.svensson</ns1:id>
<ns1:identifierAttributes>
<ns1:attr name="objectclass">
<ns67:value type="string" xmlns:ns67="urn:oasis:names:tc:DSML:2:0:core">User</
ns67:value>
</ns1:attr>
<ns1:attr name="domain">
<ns68:value type="string" xmlns:ns68="urn:oasis:names:tc:DSML:2:0:core">system</
```

```
ns68:value>
</ns1:attr>
<ns1:attr name="id">
<ns69:value type="string"
xmlns:ns69="urn:oasis:names:tc:DSML:2:0:core">sven.svensson</ns69:value>
</ns1:attr>
</ns1:identifierAttributes>
</ns1:identifier>
```

8) Then, the non empty user attributes are delivered.

```
<ns1:attributes>
<ns1:attr name="domain">
<ns70:value type="string" xmlns:ns70="urn:oasis:names:tc:DSML:2:0:core">system</
ns70:value>
</ns1:attr>
<ns1:attr name="displayName">
<ns71:value type="string" xmlns:ns71="urn:oasis:names:tc:DSML:2:0:core">Sven
 Svensson
display name</ns71:value>
</ns1:attr>
<ns1:attr name="currentTimeZone">
<ns72:value type="string" xmlns:ns72="urn:oasis:names:tc:DSML:2:0:core">Asia/
Shanghai</
ns72:value>
</ns1:attr>
<ns1:attr name="homeTimeZone">
<ns73:value type="string" xmlns:ns73="urn:oasis:names:tc:DSML:2:0:core">Asia/
Shanghai</
ns73:value>
</ns1:attr>
<ns1:attr name="defaultLocale">
<ns74:value type="string" xmlns:ns74="urn:oasis:names:tc:DSML:2:0:core">en_US</
ns74:value>
</ns1:attr>
<ns1:attr name="objectclass">
<ns75:value type="string" xmlns:ns75="urn:oasis:names:tc:DSML:2:0:core">User</
ns75:value>
</ns1:attr>
</ns1:attributes>
</ns1:searchResultEntry>
</ns1:searchResponse>
</soapenv:Body>
</soapenv:Envelope>
```

3.3.5 Export Use Case: Retrieving all Users

All available OpenScape UC Application users can be retrieved by following the instructions in Export Use Case: Retrieving a User on page 24 but removing the following lines in step 2 on page 29:

```
"<spml:attr name=\6"id\">" +
"<dsml:value>sven.svensson</dsml:value>" +
"</spml:attr>" +
```

The response now contains as many <ns1:searchResultEntry> as there are available users.

3.3.6 Import Use Case: Deleting a User

IMPORTANT: Before deleting a user, be sure to have checked whether there are other objects depending on that user. These other objects then might be modified or deleted. Example: A contact contains an `assignedSymUserIdentity` attribute having the value of the `identity` attribute of a user. The `assignedPhoneNumber` attribute of this user has the value of the `id` attribute of a phone. When deleting this user, do the following: - Delete the contact since the information it contains is only useful for the user to be deleted. - For example the `assignedPhoneNumber` attribute of another user might be assigned to the `id` attribute of this phone. If the phone is not used any more, delete it as well. For more details about meaning of a contact, see the introduction of Contact.

The Java source code for deleting an user is the same as in Import Use Case: Adding a User on page 18 but with one exception. The variable `spmlRequest` now has a different contents.

1) The SPML code described in steps 3 on page 21, 4 on page 22 and 5 on page 22 is the same for deleting a user except three items:
   a) The <`addRequest` has to be exchanged by <`deleteRequest`.

```
...
"<deleteRequest xmlns=
\"urn:oasis:names:tc:SPML:1:0\">" +
...
```

   b) There are no , and after since there are no properties to be assigned.
   c) Exchange `</addRequest>` by `</deleteRequest>`.

```
 String spmlRequest= "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
"<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
"<soapenv:Body>" +
"<deleteRequest xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<identifier xmlns=\"urn:oasis:names:tc:SPML:1:0\"
type=\"urn:oasis:names:tc:SPML:1:0#DN\">" +
"<id xmlns=\"urn:oasis:names:tc:SPML:1:0\">peter.petersson@system</id>" +
"<identifierAttributes xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"objectclass\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">User</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"domain\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">system</value>" +
"</attr>" +
"</identifierAttributes>" +
"</identifier>" +
"</deleteRequest>" +
"</soapenv:Body>" +
"</soapenv:Envelope>";
```

2) Click **Run** > **Run As** > **Java Application**.
3) Click the **OK** button.

4) The following output is created at the SPML client when everything went well.

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:deleteResponse result="urn:oasis:names:tc:SPML:1:0#success"
xmlns="urn:oasis:names:tc:SPML:1:0" xmlns:ns1="urn:oasis:names:tc:SPML:1:0"/>
</soapenv:Body>
</soapenv:Envelope>
```

5) Refresh the browser opened in step 16 on page 25. The user has been deleted.

3.3.7 Import Use Case: Adding Contact Data

IMPORTANT: It is important to point to out what is meant when talking about a contact. A contact is a collection of attributes sucha as `gender`, `salutation`, `company`, `department` etc. of an OpenScape UC Application user. These attributes are shown in the CMP in the **Contact Information** tab in the properties dialog of a user. However, for the SPML interface these attributes are not part of the `User` object (see User) but they build a separate object, the `Contact` object (see Contact). A `Contact` is linked to a `User` by the assignedSymUserIdentity attribute having the same value as the identity attribute of this `User`.

The Java source code for adding a contact is the same as in Import Use Case: Adding a User on page 18 but with one exception. The variable `spmlRequest` now has a different contents.

Execute the following steps:

1) Be sure that the user a contact should be linked to exists already or is added by a preceeding `<addRequest>` in the same SPML request.
2) The variable `spmlRequest` contains the SPML command to be executed. `<addRequest>` indicates that a new object is to be added. `<id>` identifies the user object the contact should be linked to. `<attr>` within `<identifierAttributes>` indicates that a contact is to be added since its `name` attribute has the value `objectclass` and it contains a `<value>` having the value `Contact`. No second `<attr>` within `<-identifierAttributes>` is needed.

```
String spmlRequest= "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
"<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
"<soapenv:Body>" +
"<addRequest xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<identifier xmlns=\"urn:oasis:names:tc:SPML:1:0\"
type=\"urn:oasis:names:tc:SPML:1:0#DN\">" +
"<id xmlns=\"urn:oasis:names:tc:SPML:1:0\">Peter Petersson</id>" +
"<identifierAttributes xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"objectclass\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Contact</value>" +
```

```
"</attr>" +"</identifierAttributes>" +
"</identifier>" +
```

3) The following <attr> within <attributes> describe which properties of this user should be set to which values. See Contact for details about the properties that can be set and the meanings they have.

```
"<attributes xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"lastName\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Petersson</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"displayName\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Peter Petersson display name</
value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"assignedSymUserIdentity\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">peter.petersson@system</value>"
 +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"gender\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">1</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"middleName\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Arne</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"givenName\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Peter</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"salutation\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Mr.</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"title\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Dr.</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"alias\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Karl Johann</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"countryCode\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Sweden</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"state\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Smaland</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"postalCode\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">52249</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"city\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Loeneberga</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"street\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Karl Johann Gate</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"imAddress\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">peter.petersson@imdomain</
value>"
+ "</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"companyName\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Lindgren AS</value>" +
```

```
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"department\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Development</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"costLocation\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">10000</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"domain\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">system</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"assignedEmail\">" +
"<value
xmlns=\"urn:oasis:names:tc:DSML:2:0:core
\">peter.petersson@company.com,dev07@compan
y.com</value>" +
"</attr>" +
"</attributes>" +
```

4) Be sure to close the SPML request properly.

```
"</addRequest>" +
"</soapenv:Body>" +
"</soapenv:Envelope>";
```

> **NOTICE:** All SPML requests that are described in detail in this documentation are available as XML files in the `spml_requests` directory in the ZIP file this documentation is part of.

5) Click **Run** > **Run As** > **Java Application**.
6) Click the **OK** button.
7) Open `https://<IP address>` in a browser. Substitute `<IP address>` with the IP address or the computer name of the computer the OpenScape UC Application resides on.
8) Enter the user name and password of a user having the privilege to enter the CMP.
9) Click the **User Management > Administration > Users & Resources** tab.
10) Double click the user entry you have added the contact for and click the **Contact Information** tab.

    The **Link to Sym User**, **Sym User ID** and **Vantiy Code** fields at the bottom of this tab are set automatically.

3.3.8 Import Use Case: Deleting Contact Data

The Java source code for deleting an user is the same as in Import Use Case: Adding a User on page 18 but with one exception. The variable `spmlRequest` now has a different contents.

1) The SPML code described in steps 2 on page 33, 3 on page 33 and 4 on page 35 is the same for deleting a contact except three items:

   a) The `<addRequest` has to be exchanged by `<deleteRequest`.

```
...
"<deleteRequest xmlns=
\"urn:oasis:names:tc:SPML:1:0\">" +
...
```

b) There are no <attributes>, <attr> and <value> **after** </identifier> since there are not properties to be assigned.

c) Exchange </addRequest> by </deleteRequest>.

```
String spmlRequest= "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
"<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
"<soapenv:Body>" +
"<deleteRequest xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<identifier xmlns=\"urn:oasis:names:tc:SPML:1:0\"
type=\"urn:oasis:names:tc:SPML:1:0#DN\">" +
"<id xmlns=\"urn:oasis:names:tc:SPML:1:0\">peter.petersson@system</id>" +
"<identifierAttributes xmlns=\"urn:oasis:names:tc:SPML:1:0\">" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"objectclass\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">Contact</value>" +
"</attr>" +
"<attr xmlns=\"urn:oasis:names:tc:SPML:1:0\" name=\"domain\">" +
"<value xmlns=\"urn:oasis:names:tc:DSML:2:0:core\">system</value>" +
"</attr>" +
"</identifierAttributes>" +
"</identifier>" +
"</deleteRequest>" +
"</soapenv:Body>" +
"</soapenv:Envelope>";
```

2) Click **Run** > **Run As** > **Java Application**.

3) Click the **OK** button.

4) The following output is created at the SPML client when everything went well.

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:deleteResponse result="urn:oasis:names:tc:SPML:1:0#success"
xmlns="urn:oasis:names:tc:SPML:1:0" xmlns:ns1="urn:oasis:names:tc:SPML:1:0"/>
</soapenv:Body>
</soapenv:Envelope>
```

5) Refresh the browser opened in step 10 on page 35. The contact data has been deleted.

3.3.9 Import Use Case: Setting a preferred Device when using a private Numbering Plan

If a private numbering plan is used for phones connected to an OpenScape Voice and you want to set a phone to a preferred device or ONS device by using SPML, execute the following steps:

1) Retrieve all Phone objects by sending a phone search SPML request according to the spml_messages/Phone-Search_all.xml file in this SDK.

2) The response to this request contains for each Phone object a section analog to the following one:

```
<ns1:attr name="normalizedNumber">
<ns4294:value type="string"xmlns:ns4294="urn:oasis:names:tc:DSML:2:0:core">
302107710316;phone-context=private.private.3021077@system
```

```
</ns4294:value>
</ns1:attr>
```

> **NOTICE:** Consider that the phone number `302107710316;phone-context=private.private.3021077@system` has no leading + because a private numbering plan is used.

3) Add or modify the `User` object that should be assigned to this phone number. In doing so, set the `assignedPhone8KNumber`, `publishedResource` and `preferredDeviceAndTarget` attributes of this `User` object in the SPML request to the value of the returned normalized number as mentioned above.
4) If the phone number (here: `302107710316;phone-context=private.private.3021077@system`) should be an ONS number, the `isONS` attribute of the corresponding `Phone` object must be set to `true`.

3.4 Object Reference

This section describes in detail the objects of the OpenScape UC Application being exposed via the SPML interface.

3.4.1 SPML Usage

The tables in the following sections describe the attributes an OpenScape UC Application object has (for example Phone object). The SPML usage column in these tables shows the SPML element of an SPML request referring to an attribute of an OpenScape UC Application object. The SPML usage column may have the following values:

- `<id>`

  The `<id>` element in an SPML request specifies the identifier of an object that an SPML request refers to.

  For example the alternativeId attribute of the Phone object is marked as `<id>` in the SPML usage column, i e. the value of the alternativeId attribute should appear in an `<id>` element of an SPML request.

  > **IMPORTANT:** The alternativeId attribute (i.e. a line in the tables in the following sections) is to be understood as an OpenScape UC Application attribute here, not as an SPML attribute or XML attribute.

  > **NOTICE:** An OpenScape UC Application attribute that is used in the `<id>` element of an SPML request, for example the alternativeId attribute, is not mentioned by its name in this SPML request.

  Example:

```
<identifier xmlns="urn:oasis:names:tc:SPML:1:0"
 type="urn:oasis:names:tc:SPML:1:0#DN">
<id xmlns="urn:oasis:names:tc:SPML:1:0">15619231001</id>
<identifierAttributes xmlns="urn:oasis:names:tc:SPML:1:0">
…
</identifierAttributes>
</identifier>
```

- `<identifierAttributes>`

  The value of an attribute that is marked as `<identifierAttributes>` should appear inside an `<identifierAttributes>` element of an SPML request. The `<identifierAttributes>` element contains additional attributes to further identification of an object. The attributes are specified as subelements using `<attr>` elements.

  The `domain` and `objectclass` attributes are specified in the `<identifierAttributes>` element of almost all SPML requests.

  Example:

```
<identifier xmlns="urn:oasis:names:tc:SPML:1:0"
 type="urn:oasis:names:tc:SPML:1:0#DN">
<id xmlns="urn:oasis:names:tc:SPML:1:0">15619231001</id>
<identifierAttributes xmlns="urn:oasis:names:tc:SPML:1:0">
<attr xmlns="urn:oasis:names:tc:SPML:1:0" name="objectclass">
<value xmlns="urn:oasis:names:tc:DSML:2:0:core">Phone</value>
</attr>
<attr xmlns="urn:oasis:names:tc:SPML:1:0" name="domain">
<value xmlns="urn:oasis:names:tc:DSML:2:0:core">system</value>
</attr>
</identifierAttributes>
</identifier>
```

- `<attributes>`

  The `<attributes>` element is used in SPML add requests and SPML responses in order to specify the attribute values of the newly created object or the retrieved object. Each attribute is specified as an `<attr>` subelement of the `<attributes>` tag.

  Example for a User whose `displayName` attribute is `Pier` and whose `defaultLocale` attribute is `fr_FR`:

```
<identifier xmlns="urn:oasis:names:tc:SPML:1:0"
 type="urn:oasis:names:tc:SPML:1:0#DN">
<id xmlns="urn:oasis:names:tc:SPML:1:0">pier-ons</id>
<identifierAttributes xmlns="urn:oasis:names:tc:SPML:1:0">
…
</identifierAttributes>
</identifier>
<attributes xmlns="urn:oasis:names:tc:SPML:1:0">
<attr xmlns="urn:oasis:names:tc:SPML:1:0" name="displayName">
<value xmlns="urn:oasis:names:tc:DSML:2:0:core">Pier</value>
</attr>
<attr xmlns="urn:oasis:names:tc:SPML:1:0" name="defaultLocale">
<value xmlns="urn:oasis:names:tc:DSML:2:0:core">fr_FR</value>
</attr>
</attributes>
```

- <modifications>

  The <modifications> element is used in SPML modify requests in order to specify which attributes should be modified. Each attribute that should be modified is specified as a <modification> subelement of the <modifications> element.

  Example for changing the assignedPhone8kNumber attribute of Pier to +302108045014:

```
<identifier xmlns="urn:oasis:names:tc:SPML:1:0"
 type="urn:oasis:names:tc:SPML:1:0#DN">
<id xmlns="urn:oasis:names:tc:SPML:1:0">pier</id>
<identifierAttributes xmlns="urn:oasis:names:tc:SPML:1:0">
</identifierAttributes>
</identifier>
<modifications xmlns="urn:oasis:name:tc:SPML:1:0">
<modification name="assignedPhone8kNumber"
operation="replace"
xmlns="urn:oasis:name:tc:SPML:1:0">
<value xmlns="urn:oasis:names:tc:DSML:2:0:core">+302108045014</value>
</modification>
…
</modifications>
```

- <operationalAttributes>

  The value of an attribute marked as <operationalAttributes> should appear in an <operationalAttributes> element. An <operationalAttributes> element augments an SPML request by specifying extra operations to take place along with the SPML request.

  Example:

```
<deleteRequest xmlns="urn:oasis:names:tc:SPML:1:0">
<identifier type="urn:oasis:names:tc:SPML:1:0#DN"
xmlns="urn:oasis:names:tc:SPML:1:0">
<id xmlns="urn:oasis:names:tc:SPML:1:0">302108045009</id>
<identifierAttributes xmlns="urn:oasis:names:tc:SPML:1:0">
…
</identifierAttributes>
</identifier>
<operationalAttributes>
<attr name="is8KProvisioning" xmlns=" urn:oasis:names:tc:SPML:1:0">
<value xmlns="urn:oasis:names:tc:DSML:2:0:core">true</value>
</attr>
</operationalAttributes>
</deleteRequest>
```

3.4.2 Domain

The Domain object represents the domain of OpenScape UC Application, see in the CMP under **Configuration** > **CMP** > **General** > **Domains**. It cannot be created or modified (imported), only read (exported).

Table 2: Attributes of the `Domain` Object

Attribute	Description	Necessity	Example
`parentDomainName`	The name of the parent domain. This attribute is always empty and for future purpose	Mandatory	
objectclass	The constant value of the object that is provisioned. Its value is always Domain.	Mandatory	Domain
name	Symphonia domain name **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`
description	The description of the domain		`'system' is the default domain.`

attributes	Comma separated list of key-value pairs to set customized attributes, for example `key1=value1, key2=value2`.		`workgroupPurgeTime=0,`
			`journalPurgeTime=0,`
			`RingNoAnswer=0,`
			`conferencePurgeTime=0`

3.4.3 `User`

Table 3: Attributes of the `User` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	Unique identifier for an OpenScape UC Application user. This attribute does not contain the domain. This is a mandatory attribute, if the attribute `identity` is not used. Use this `id` attribute as the value of the `<id>` element in the `<identifier>` element of the SPML request (see step 3 on page 21). Do not use this `id` attribute in the `<attributes>` element of the SPML request (see step 4 on page 22).	Mandatory	`Peter Petersson`	`<id>`
`objectclass`	The constant value of the object that is provisioned. Its value is always `User`.	Mandatory	User	`<identifierAttributes>`

Attribute	Description	Necessity	Example	SPML usage
`domain`	Symphonia domain the user belongs to. Use this `domain` attribute in the `<identifierAttributes>` element of the SPML request (see step 3 on page 21). **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`. This is a mandatory attribute, if the attribute `identity` is not used. `domain` must match the value of `<identifier> <idententifier Attributes><attr ... name= ”domain”> <value>` of the SPML request.If `domain` and `identity` are not empty, the value of `domain` is ignored.	Mandatory	`system`	`<identifierAttributes>`
`identity`	OpenScape UC application user identity including the domain. This attribute can be used instead of `id` and `domain`. If `id` and `identity` are not empty, the value of `id` is ignored. If `domain` and `identity` are not empty, the value of `domain` is ignored.	Mandatory	`peter. petersson @system`	`<attributes>, <modifications>`
`displayName`	Display name of the user. It is for example used in the CMP.	Mandatory	`Peter Petersson`	`<attributes>, <modifications>`

Attribute	Description	Necessity	Example	SPML usage
`publishedResource`	This corresponds to the identifier of the corresponding resource object. `publishedResource` either is the user's ONS number or an alternative identifier. If `publishedResource` is a user's ONS number, it is a normalized number. If the Phone object is the corresponding object, `publishedRessource` corresponds to the alternativeId attribute of that object. If the Phone4k object is the corresponding object, `publishedRessource` corresponds to the alternativeId attribute of that object.	Mandatory[1]	`+49897221001`	`<attributes>`, `<modifications>`
`assignedVoicemail`	This attribute corresponds to the id attribute in global number form of the corresponding VoiceMail object.	Optional	`+49897223101`	`<attributes>`, `<modifications>`

1 Only if `callForwardingDependableTarget` is set.

Attribute	Description	Necessity	Example	SPML usage
`assignedProfile`	A comma separated list of assigned profiles. A single value or a comma separated list of values are allowed. Allowed values: • `Professional: OpenScape UC App` • `Essential: OpenScape UC App` • `Standard: E/A-Cockpit` • `Team: OpenScape UC App` • `Premium: E/A-Cockpit` • `Customer Administrator: H8kAst` • `Basic Administrator: H8kAst` • `Super Administrator: H8kAst` • `Configuration: RGAssistant` • `Supervision: RGAssistant` • `Administration: RGAssistant` • `Administrator: Symphonia` • `CustomerAdministrator: Symphonia` • `ServiceTechnician: Symphonia`	Optional	`Professional: OpenScape UC App , Super Administrator: H8kAst, ServiceTechnician: Symphonia`	`<attributes>, <modifications>`

Attribute	Description	Necessity	Example	SPML usage
password	User password in plain text. The password must comply to the password policy. If password is not provided in an add request, the user will obtain the default domain password. If there is no default domain password either, the user's configuration request will be rejected. If password is not provided in a modify request, the user's password will remain unchanged. If the pwdNeedChange attribute or the pwdNeverExpires attribute is set to 1, the password attribute must be set. **NOTE:** The password attribute cannot be exported.	Optional	Gair2_xrVWX	<attributes>, <modifications>
homeTimeZone	Home time zone. This time zone must have the format of an Olson ID as used by Java, GNU etc. (see Table 4 on page 48).	Mandatory	Europe/ Berlin	<attributes>, <modifications>
currentTimeZone	Current time zone. This time zone must have the format of an Olson ID as used by Java, GNU etc. (see Table 4 on page 48).	Optional	Asia/ Shanghai	<attributes>, <modifications>
defaultLocale	The locale must have the format <language_code >_<country_code>, for example de_DE or en_US with the language code following ISO 639-1 and the country code following ISO 3166-1. defaultLocale controls the language of the conference announcement the user is going to listen to when entering a conference not by dialing the conference bridge.	Mandatory	de_DE	<attributes>, <modifications>

Attribute	Description	Necessity	Example	SPML usage
`vanityCodeString`	The vanity code in string representation. Telephone numbers represented as letters are called vanity code. In addition to digits, the single telephone keys are assigned to letters and special characters. Thus, you can either keep in mind for example the phone number 0700-74636 or its vanity code, i.e. 0700-SIMEN.	Optional	`1`	`<attributes>`, `<modifications>`
`pin`	The PIN in plain text. The PIN must comply to the PIN policy (see section “Changing Policies for User PINs” in administrator documentation *OpenScape UC Application Configuration and Administration*). **NOTE:** The `pin` attribute cannot be exported.	Optional		`<attributes>`, `<modifications>`
`allocatedPhone`	This should be a comma separated list of allocated phone IDs (id attribute of the AllocatedPhone object).	Optional	`49897229001,` `491706315034`	`<attributes>`, `<modifications>`

`assignedPhone8kNumber`	Comma separated list of the normalized numbers of Phone objects that are assigned to this user.
	If a public numbering plan is used, the normalized numbers start with a ‘+’ character. If a private numbering plan is used, they do not.
		2	`+49897221001,`
			`+49897221002,`
			`+49897221003`
				`<attributes>`, `<modifications>`

`assignedPhone8kAltId`	Comma separated list of alternativeId attribute of the Phone objects that are assigned to this user.	2	`Mike’s Phone,`
			`John’s Phone,`
			`+49897221001`
				`<attributes>`, `<modifications>`

2 Either `assignedPhone8kNumber` or `assignedPhone8kAltId`must be set, if a Phone object is assigned to this user.

Attribute	Description	Necessity	Example	SPML usage

`assignedPhone4kNumber`	Comma separated list of normalizedNumber attribute of the Phone4k objects that are assigned to this user.
	If a public numbering plan is used, the normalized numbers start with a '+' character. If a private numbering plan is used, they do not.
		3	`+49897221001,`
			`+49897221002,`
			`+49897221003`
				`<attributes>,`
				`<modifications>`

`assignedPhone4kAltId`	Comma separated list of alternativeId attribute of the Phone4k objects that are assigned to this user.	3	`Mike's Phone,` `John's Phone`	`<attributes>,` `<modifications>`
`assignedCallQueue`	id attribute of the of corresponding QueueDevice object	Optional	`+49897223100`	`<attributes>,` `<modifications>`
`preferredDeviceAndTarget`	The identifier of an assigned phone device (alternativeId of a Phone object, alternativeId of a Phone4k object or the id of an AllocatedPhone object) to be set as the preferred device and target. A "desk phone" could be the preferred target.	Optional	`49897221001@1`	`<attributes>,` `<modifications>`
`callForwarding` `DependableTarget`	The number of the dependable forwarding target to be provisioned into the OpenScape Voiceand which will be used by OpenScape UC Voice Server in case of OpenScape UC Application server response timeout. A "desk phone" could be the call forward dependable target. **NOTE:** The `callForwarding DependableTarget` attribute cannot be exported.	Optional	`498972229001`	`<attributes>,` `<modifications>`

3 Either `assignedPhone4kNumber` or `assignedPhone4kAltId`must be set, if a Phone4k object is assigned to this user.

Attribute	Description	Necessity	Example	SPML usage
`pwdNeedChange`	pwdNeedChange indicates whether the `password` has to be changed when the user logs in the next time (value: `1`) or not (value: `0`). If the pwdNeedChange attribute is set to `1`, the `password` attribute must be set. **NOTE:** The pwdNeedChange attribute cannot be exported.	Optional	`0` or `1` Default: `1`	`<attributes>`, `<modifications>`
`pwdNeverExpire`	pwdNeverExpire indicates whether the password provided in the password attribute never expires (value: `1`) or not (value: `0`). If the pwdNeverExpire attribute is set to `1`, the `password` attribute must be set. **NOTE:** The pwdNeverExpire attribute cannot be exported.	Optional	`0` or `1` Default: `1`	`<attributes>`, `<modifications>`
`translatorContextId`	This attribute designates the ID of the translator context that the user is associated with.	Optional		`<attributes>`, `<modifications>`
`WebRTCEnabled`	This attribute indicates whether the user has WebRTC functionalities enabled.	Optional	`0` or `1` Default: `0`	`<attributes>`, `<modifications>`
`webRTCResource`	This attribute indicates the device resource with the WebRTC functionality enabled.	Optional	`+49897223101`	`<attributes>`, `<modifications>`
`mlhgBusyAddress`	This attribute indicates the Toggle MLHG Make Busy subscription code, associated with a user's WebRTC device.	Optional	*6	`<attributes>`, `<modifications>`
`mlhgStopAddress`	This attribute indicates the Toggle MLHG Stop Hunt subscription code, associated with a user's WebRTC device.	Optional	*7	`<attributes>`, `<modifications>`
`pugAddress`	This attribute indicates the Group Call Pickup subscription code, associated with a user's WebRTC device.	Optional	*8	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
homeServer	This attribute is the IP address or computer name of the application computer (UC backend computer), i.e. the computer that has the type attribute with the value BACKEND in the <Node> XML element in the system.xml file: <SystemData displayname= "Symphonia System" ...> <SystemProperties> <Property name="Hotline" value=""/> ..... </SystemProperties> <NodeList> <Node displayname= "offboard" ...... type="BACKEND"/> ..... </NodeList> </SystemData> If the homeServer attribute is not specified, the user will be imported. Otherwise, the user is only imported if the value of the homeServer attribute is identical to the IP address or computer name of the application computer (type=BACKEND) the SMPL message is sent to.	Optional	cluster1	<attributes>, <modifications>
keycloak2FAEnabled	This attribute indicates whether Keycloak 2FA is enabled.	Optional	0 or 1	<attributes>, <modifications>

The Olson IDs in the following table can be used for the `homeTimeZone` and `currentTimeZone` attributes.

Table 4: Examples for Olson IDs of Time Zones

Time Zone	Olson ID	Offset to UTC
Honolulu	`Pacific/Honolulu`	UTC-10
Anchorage	`America/Anchorage`	UTC-9
Pacific Standard Time	`US/Pacific`	UTC-8
Moutain Standard Time	`US/Mountain`	UTC-7
Central Standard Time	`US/Central`	UTC-6
Eastern Standard Time	`US/Eastern`	UTC-5
Atlantic Standard Time	`Canada/Atlantic`	UTC-4
Sao Paulo	`America/Sao_Paulo`	UTC-3
South Georgia	Atlantic/South_Geogia	UTC-2
Azores	Atlantic/Azores	UTC-1
Reykjavik	`Atlantic/Reykjavik`	UTC
London	`Europe/London`	UTC
Lisbon	`Europe/Lisbon`	UTC
Berlin	`Europe/Berlin`	UTC+1
Madrid	`Europe/Madrid`	UTC+1
Paris	`Europe/Paris`	UTC+1
Rome	`Europe/Rome`	UTC+1
Cairo	Africa/Cairo	UTC+2
Johannesburg	Africa/Johannesburg	UTC+2
Moscow	`Europe/Moscow`	UTC+3
Tehran	`Asia/Tehran`	UTC+3:30
Yerevan (Armenia)	`Asia/Yerevan`	UTC+4
Calcutta	`Asia/Calcutta`	UTC+5:30
Dhaka	Asia/Dhaka	UTC+6
Ho Chi Minh	`Asia/Saigon`	UTC+7
Hong Kong	`Asia/Hong_Kong`	UTC+8
Shanghai	`Asia/Shanghai`	UTC+8
Taipei	`Asia/Taipei`	UTC+8

Time Zone	Olson ID	Offset to UTC
Tokyo	`Asia/Tokyo`	UTC+9
Sydney (New South Wales)	Australia/Sydney	UTC+10
Salomon Islands	`Pacific/Guadalcanal`	UTC+11
Auckland	`Pacific/Auckland`	UTC+12

The following table lists all locales that can be used in the OpenScape database. You need to enter the prefix of the desired locale in the respective database field.

Table 5: Values for the `defaultLocale` attribute

Locale	Prefix
Albanian (Albania)	sq_AL
Albanian	sq
Arabic (Algeria)	ar_DZ
Arabic (Bahrain)	ar_BH
Arabic (Egypt)	ar_EG
Arabic (Iraq)	ar_IQ
Arabic (Jordan)	ar_JO
Arabic (Kuwait)	ar_KW
Arabic (Lebanon)	ar_LB
Arabic (Libya)	ar_LY
Arabic (Morocco)	ar_MA
Arabic (Oman)	ar_OM
Arabic (Qatar)	ar_QA
Arabic (Saudi Arabia)	ar_SA
Arabic (Sudan)	ar_SD
Arabic (Syria)	ar_SY
Arabic (Tunisia)	ar_TN
Arabic (United Arab Emirates)	ar_AE
Arabic (Yemen)	ar_YE
Arabic	ar
Belarusian (Belarus)	be_BY
Belarusian	be
Bulgarian (Bulgaria)	bg_BG
Bulgarian	bg

Locale	Prefix
Catalan (Spain)	ca_ES
Catalan	CA
Chinese (China)	zh_CN
Chinese (Hong Kong)	zh_HK
Chinese (Singapore)	zh_SG
Chinese (Taiwan)	zh_TW
Chinese	zh
Croatian (Croatia)	hr_HR
Croatian	hr
Czech (Czech Republic)	cs_CZ
Czech	cs
Danish (Denmark)	da_DK
Danish	da
Dutch (Belgium)	nl_BE
Dutch (Netherlands)	nl_NL
Dutch	nl
English (Australia)	en_AU
English (Canada)	en_CA
English (India)	en_IN
English (Ireland)	en_IE
English (Malta)	en_MT
English (New Zealand)	en_NZ
English (Philippines)	en_PH
English (Singapore)	en_SG
English (South Africa)	en_ZA
English (United Kingdom)	en_GB
English (United States)	en_us
English	en
Estonian (Estonia)	et_EE
Estonian	et

Locale	Prefix
Finnish (Finland)	fi_FI
Finnish	fi
French (Belgium)	fr_BE
French (Canada)	fr_CA
French (France)	fr_FR
French (Luxembourg)	fr_LU
French (Switzerland)	fr_CH
French	fr
German (Austria)	de_AT
German (Germany)	de_de
German (Luxembourg)	de_LU
German (Switzerland)	de_CH
German	de
Greek (Cyprus)	el_CY
Greek (Greece)	el_GR
Greek	el
Hebrew (Israel)	iw_IL
Hebrew	iw
Hindi (India)	hi_IN
Hungarian (Hungary)	hu_HU
Hungarian	hu
Icelandic (Iceland)	is_IS
Icelandic	is
Indonesian (Indonesia)	in_ID
Indonesian	In
Irish (Ireland)	ga_IE
Irish	ga
Italian (Italy)	it_IT
Italian (Switzerland)	it_CH
Italian	it
Japanese (Japan)	ja_JP
Japanese (Japan,JP)	ja_JP_JP
Japanese	ja
Korean (South Korea)	ko_KR
Korean	ko

Locale	Prefix
Latvian (Latvia)	lv_LV
Latvian	lv
Lithuanian (Lithuania)	lt_LT
Lithuanian	lt
Macedonian (Macedonia)	mk_MK
Macedonian	mk
Malay (Malaysia)	ms_MY
Malay	MS
Maltese (Malta)	mt_MT
Maltese	mt
Norwegian (Norway)	no_NO
Norwegian (Norway,Nynorsk)	no_NO_NY
Norwegian	no
Polish (Poland)	pl_PL
Polish	pl
Portuguese (Brazil)	pt_BR
Portuguese (Portugal)	pt_PT
Portuguese	pt
Romanian (Romania)	ro_RO
Romanian	ro
Russian (Russia)	ru_RU
Russian	ru

Locale	Prefix
Serbian (Bosnia and Herzegovina)	sr_BA
Serbian (Montenegro)	sr_ME
Serbian (Serbia and Montenegro)	sr_CS
Serbian (Serbia)	sr_RS
Serbian	sr
Slovak (Slovakia)	sk_SK
Slovak	sk
Slovenian (Slovenia)	sl_SI
Slovenian	sl
Spanish (Argentina)	es_AR
Spanish (Bolivia)	es_BO
Spanish (Chile)	es_CL
Spanish (Colombia)	es_CO
Spanish (Costa Rica)	es_CR
Spanish (Dominican Republic)	es_DO
Spanish (Ecuador)	es_EC
Spanish (El Salvador)	es_SV
Spanish (Guatemala)	es_GT
Spanish (Honduras)	es_HN
Spanish (Mexico)	es_MX
Spanish (Nicaragua)	es_NI
Spanish (Panama)	es_PA
Spanish (Paraguay)	es_PY
Spanish (Peru)	es_PE
Spanish (Puerto Rico)	es_PR
Spanish (Spain)	es_ES
Spanish (United States)	es_US
Spanish (Uruguay)	es_UY
Spanish (Venezuela)	es_VE
Spanish	es
Swedish (Sweden)	sv_SE
Swedish	sv

Locale	Prefix
Thai (Thailand)	th_TH
Thai (Thailand,TH)	th_TH_TH
Thai	th
Turkish (Turkey)	tr_TR
Turkish	tr
Ukrainian (Ukraine)	uk_UA
Ukrainian	uk
Vietnamese (Vietnam)	vi_VN
Vietnamese	vi

IMPORTANT:

For existing users, adding a new WebRTC resource via `modifyRequest` is currently **not possible**. This change requires to delete the user first and then add it again from scratch via `addRequest`, with both the ONS and WebRTC resources.

3.4.4 `Contact`

IMPORTANT: A contact is a collection of attributes such a as `gender`, `salutation`, `company`, `department` etc. which might be associated to an OpenScape UC Application user. A contact associated to a user is shown in the CMP in the **Contact Information** tab in the properties dialog of a user. However, these attributes are not part of the `User` object (see User) but they build a separate object, the `Contact` object (see Contact). A `Contact` can be linked to a `User` by the assigning his `UserId` to the assignedSymUserIdentity attribute.

A `Contact` can be linked to only one `User`, i. e. the `assignedSymUserIdentity` attribute of a `Contact` object contains only one `UserId`. This reflects the situation in the CMP where an OpenScape UC Application user only has one **Contact Information** tab. A `User` object is not linked to a `Contact` object, i. e. there is no attribute of the `User` object containing a contact ID.

If you assign a contact to a user by using the assignedSymUserIdentity attribute, be sure to have created this user before. This might be done in the same SPML request (see step 1 on page 33). This association implicitly creates an `externalId` attribute in the `User` object. After deleting a `Contact`, the `externalId` associated to the `Contact` must be set deleted as well.

The contact object is a contact in the global contact list. Most of the fields need no explanation.

Table 6: Attributes of the `Contact` Object

Attribute	Description	Necessity	Example	SPML usage
`assignedSymUserIdentity`	`assignedSymUserIdentity` corresponds to the identity attribute of the User object this contact belongs to. `assignedSymUserIdentity` neither corresponds to id nor to domain of the User object. `assignedSymUserIdentity` should be a fully qualified user ID (userId@domain). Use this `assignedSymUserIdentity` attribute as the value of the `<id>` element in the `<identifier>` element of the SPML request (see step 2 on page 33). Do not use this `assignedSymUserIdentity` attribute in the `<attributes>` element of the SPML request (see step 2 on page 33). It would be ignored. Maximum length: 255 characters		`peter.petersson@system`	`<id>`
`objectclass`	The constant value of the object that is provisioned. Its value is always `Contact`. Maximum length: 255 characters	Mandatory	`Contact`	`<identifierAttributes>`
`domain`	The domain the contact belongs to. **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Optional	`system`	`<identifierAttributes>`
`externalId`	Unique identifier identifying the contact in the external system Maximum length: 255 characters	Optional		`<attributes>, <modifications>`
`middleName`	User's middle name Maximum length: 64 characters	Optional	`Arne`	`<attributes>, <modifications>`
`lastName`	User's family name Maximum length: 64 characters	Optional	`Petersson`	`<attributes>, <modifications>`

Attribute	Description	Necessity	Example	SPML usage
`givenName`	User's first name Maximum length: 64 characters	Optional	`Peter`	`<attributes>`, `<modifications>`
`displayName`	The name being displayed, for example in the CMP Maximum length: 128 characters	Mandatory	Peter Petersson	`<attributes>`, `<modifications>`
`salutation`	Salutation Maximum length: 64 characters	Optional	`Mr.`	`<attributes>`, `<modifications>`
`title`	Title Maximum length: 64 characters	Optional	`Prof.`	`<attributes>`, `<modifications>`
`alias`	Alias Maximum length: 64 characters	Optional		`<attributes>`, `<modifications>`
`gender`	Gender	Optional	`1` (male) or `2` (female) or `0` (unknown)	`<attributes>`, `<modifications>`
`countryCode`	Maximum length: 64 characters	Optional		`<attributes>`, `<modifications>`
`stateOrProvince`	Maximum length: 64 characters	Optional	North-Rhine Westphalia	`<attributes>`, `<modifications>`
`postalCode`	Postal code Maximum length: 32 characters	Optional	`52477`	`<attributes>`, `<modifications>`
`city`	City Maximum length: 64 characters	Optional	`Berlin`	`<attributes>`, `<modifications>`
`street`	Street Maximum length: 128 characters	Optional	`Main Street`	`<attributes>`, `<modifications>`
`building`	Building number Maximum length: 32 characters	Optional	`G`	`<attributes>`, `<modifications>`
`room`	Room number Maximum length: 32 characters	Optional	`R.E.04`	`<attributes>`, `<modifications>`
`homeURL`	Home URL Maximum length: 2048 characters	Optional	http://www.unify.com	`<attributes>`, `<modifications>`
`companyName`	Company name Maximum length: 64 characters	Optional	`Unify`	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
`department`	Department Maximum length: 64 characters	Optional	`Development`	`<attributes>, <modifications>`
`costLocation`	Cost ID Maximum length: 32 characters	Optional	`10000`	`<attributes>, <modifications>`
`assignedPhoneNumber`	Comma separated list of business phone numbers. A maximum of two business phones is allowed. Use localized numbers only, do not use numbers in global number format (GNF).	Optional	`02404901450,` `02404901172`	`<attributes>, <modifications>`
`assignedEmail`	Comma separated list of email addresses. A maximum of two email addresses is allowed.	Optional	`peter.petersson` `@company.com, dev07@company.com`	`<attributes>, <modifications>`
`assignedFax`	Fax number. Use a localized number only, do not use a number in global number format (GNF).	Optional	`02404901250`	`<attributes>, <modifications>`
`assignedMobile`	Mobile phone number. Use a localized number only, do not use a number in global number format (GNF).	Optional	`0171123456`	`<attributes>, <modifications>`
`assignedHomePhone`	Home phone number. Use a localized number only, do not use a number in global number format (GNF).	Optional	`02403123456`	`<attributes>, <modifications>`
`assignedExternalSystemId`	externalId attribute of ExternalIdentifier object the contact comes from	Optional		`<attributes>, <modifications>`
`imAddress`	The instant message ID of the contact Maximum length: 129 characters		`peter.petersson` `@imdomain`	`<attributes>, <modifications>`
`vanityCode`	Vanity code Maximum length: 129 characters	Optional		`<attributes>, <modifications>`
`description`	Description of the contact	Optional	Contact person for sales	`<attributes>, <modifications>`
`notice`	Notice for the contact	Optional	Person to contact for sales	`<attributes>, <modifications>`

3.4.5 Phone

The Phone object describes the ONS number the user will be assigned to, if the user is connected to an OpenScape Voice PBX. Use a Phone4k object instead, if the user is connected to a HiPath 4000 PBX.

When importing Phone objects to the OpenScape Voice directly, the corresponding resources in the OpenScape UC Application database are created by the DSA Service, which establishes an automatic synchronization mechanism between OpenScape Voice and the OpenScape UC Application. The DSA Service is not part of the import export service. Furthermore, the OpenScape Voice extension service enhances the data of the synchronized resources in the OpenScape UC Application data base

Table 7: Attributes of the Phone Object

Attribute	Description	Necessity	Example	SPML usage

`alternativeId`	The alternative unique identifier of the phone. It can be arbitrary text, if the `is8KProvisioning` attribute is set to `false`.
	
	This can be used in cases where an identifier generated by Symphonia (i.e. the UUID key or the normalized number) is not usable.
	
	If `is8KProvisioning` is set to `true` in SPML add requests, the `alternativeId` is ignored. However, it must be specified as the `<id>` of the SPML request. An `alternativeId` is composed in the background by either the public numbering plan and the public extension or by the private numbering plan and the private extension.
	
	Example:
	
	If `assignedNumbering`
	
	`PlanPublicId` is set to `public.30210804@system` and `publicExtension` is set to `5014`, `alternativeId` will be set to `302108045014` and the normalized number will be set to `+301208045014`.
	
	If `is8KProvisioning` is set to `false` and `alternativeId` is empty in SPML add requests, the `alternativeId` is be composed from the numbering plan and the public or private extension.
	
	Example:
	
	If `assignedNumbering`
	
	`PlanPublicId` is set to `public.30210804@system`
		Mandatory	`Mike's Phone`
			
			or
			
			`public.30210804`
			
			`@system`
			
			or
			
			`492404123456100`
				`<id>`

Attribute	Description	Necessity	Example	SPML usage
	and `publicExtension` is set to `5014`, `alternativeId` will be set to `+302108045014`. The normalized number will have the same value. Use this `alternativeId` attribute as the value of the `<id>` element in the `<identifier>` element of the SPML request (`<id>alternativeId</id>`). Do not use the `alternativeId` attribute in the `<attributes>` element of the SPML request. It would be ignored.			
`objectclass`	The constant value of the object that is provisioned. Its value is always `Phone`.	Mandatory	`Phone`	`<identifierAttributes>`
`domain`	The domain the phone belongs to. It has to be the same domain as in the associated number plans and dial plans. **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory for creation	`system`	`<identifierAttributes>`
`switchId`	id attribute of the Switch8k object this phone is assigned to. In order to assign the phone to a Switch8k object, this Switch8k object has to be created before. This might have been done in the same SPML request.	Mandatory	`1`	`<identifierAttributes>`

Attribute	Description	Necessity	Example	SPML usage
publicExtension	Public number extension of the telephone without overlapping digits. It is only used, if a public numbering plan is used.	4	1001 for the normalized number +49897221001	<attributes>, <modifications>
privateExtension	Private number extension of the telephone without overlapping digits. It is only used, if a private numbering plan is used.	4	1234 for 2221201234	<attributes>, <modifications>
normalizedNumber	This is a phone number internally built out of extension and numbering plan attributes. The normalizedNumber attribute cannot be set in a SPML add or modify request but can it be retrieved by using an SPML search request. **NOTE:** The normalizedNumber attribute does not start with + if a private numbering plan is used.	Optional	When using a public numbering plan: +492404673407160 When using a private numbering plan: 302107710316; phone-context=private.private.3021077@system	<attributes>
officeCode	The officeCode attribute will be used, if the is8KProvisioning attribute is set to true. If it is set, the assignedNumberingPlanPublicId attribute and the assignedNumberingPlanPrivate attribute will be ignored.	4	492404105	<attributes>, <modifications>

4 You either use a public numbering plan or a private numbering plan. If you use a public numbering plan, the following attributes are mandatory: publicExtension either officeCode (only possible if is8KProvisioning attribute is set to true) or assignedNumberingPlanPublicId assignedDialPlanPublicId In this case, the following attributes must not be set: privateExtension assignedNumberingPlanPrivateId assignedDialPlanPrivateId If you use a private numbering plan, the following attributes are mandatory: privateExtension officeCode (only if the is8KProvisioning attribute is set to true) or assignedNumberingPlanPrivateId (only if the is8KProvisioning attribute is not available or set to false) assignedDialPlanPrivateId In this case, the following attributes must not be set: publicExtension assignedNumberingPlanPublicId assignedDialPlanPublicId

Attribute	Description	Necessity	Example	SPML usage
`assignedNumbering PlanPulicId`	id attribute of a NPPublic object, if a public numbering plan is used. This attribute is only valid, if the `officeCode` attribute is not set.	4	`public.492404105 @system`	`<attributes>, <modifications>`
`assignedNumbering PlanPrivateId`	id attribute of a NPPrivate object, if a private numbering plan is used. This attribute is only valid, if the `officeCode` attribute is not set.	4	`private.3344555 @system`	`<attributes>, <modifications>`
`assignedNumbering PlanUnknownId`	id attribute of a NPUnknown object, if an unknown numbering plan is used.	4		`<attributes>, <modifications>`
`assignedDial PlanPublicId`	id attribute of a DPPublic object this phone is assigned to, if a public numbering plan is used. In case of using a public numbering plan, the GNF format is used. In order to assign the public dial plan to a phone, the public dial plan has to be defined in a previous administration action.	4	`public.492404105 @system`	`<attributes>, <modifications>`
`assignedDial PlanPrivateId`	id attribute of the DPPrivate object this phone is assigned to, if a private numbering plan is used. In order to assign the private dial plan to a phone, the private dial plan has to be defined in a previous administration action.	4	`private.3344555 @system`	`<attributes>, <modifications>`
`orphan`	Flag set by the OpenScape Voice Assistant indicating that the subscriber line to this phone has been deleted on the OpenScape Voice.	Optional	true	`<attributes>, <modifications>`

Attribute	Description	Necessity	Example	SPML usage

`BCFedId`	id attribute of the BCFederation object (BCOM federation) that this phone is assigned to. This attribute only is set, if the phone is associated to a BC federation.
	In order to assign the BC federation to a phone, the federation has to be defined in a previous administration action.
		Optional	`1@TAS Node`
			or
			`2@offboard`
				`<attributes>, <modifications>`

isONS	Flag indicating whether this phone is the one number service device of the user	Optional	true	`<attributes>, <modifications>`
`assignedBusinessGroup`	ID of the assigned business group used by the OpenScape Voice Assistant	Mandatory, if the `propagateToSwitch` or - `is8kProvisioning` attribute is set to true	`BG1`	`<attributes>, <modifications>`
`propagateToSwitch`	This attribute is for backward compatibility reasons only. Use `is8KProvisioning` instead. If this attribute is set to `true`, the phone object is not only provisioned into the OpenScape UC Application but it is also provisioned directly into the OpenScape Voice. If this attribute is set to true, the assigned DPPublic, the switch8k and the business group have to be defined in the OpenScape Voice before. **NOTE:** The propagateToSwitch attribute cannot be exported. See the `attributes` attribute for further information.	Optional	`true` Default value: `false`	`<attributes>, <modifications>`

Attribute	Description	Necessity	Example	SPML usage
`is8KProvisioning`	If this attribute is set to `true`, the phone object is not only provisioned into the OpenScape UC Application but it is also provisioned directly into the OpenScape Voice. If this attribute is set to true, the assigned DPPublic, the switch8k and the business group have to be defined in the OpenScape Voice before. **NOTE:** The `is8KProvisioning` attribute cannot be exported. See the `attributes` attribute for further information.	Optional	`true` Default value: `false`	`<attributes>`[5], <modifications>[5]
`displayName`	The display name of the subscriber in OpenScape Voice. Use this property only if the `is8KProvisioning` attribute is set to `true`.	Optional	Peter Petersson	`<attributes>`, `<modifications>`

5 `<identifierAttributes>` in an SPML modify request, `<operationalAttributes>` in an SPML delete request

Attribute	Description	Necessity	Example	SPML usage

`attributes`	Comma separated list of key-value pairs to set customized attributes, for example “key1=value1, key2=value2”.
	NOTE: The `attributes` attribute is not exported and the normalizedNumber is not imported via attributes.
	If the `is8KProvisioning` attribute is set to true, the following keys are reasonable for provisioning the data of a business group line (BGL). A BGL is the OpenScape Voice representation of a device.
	`externalUnicodeName`: This key is the caller ID for external calls. It can only be used for single step provisioning. It cannot be exported. Single step provisioning means one data import at once, the other one is bulk provisioning more than one data import at once.
	`internalUnicodeName`: This key is the caller ID for internal calls. It can only be used for single step provisioning. It cannot be exported.
	`featureProfileID`: This is the feature profile name to be assigned to the business group line. It can only be used for single step provisioning. It cannot be exported. If the feature profile is defined in terms of a business group, `featureProfileId` should be set to <BusinessGroup> #<FeatureProfile>. Otherwise (i e. the feature profile is a global one), it should be set to the ID of feature profile.
			`featureProfile= ONS_BG`
			`#FP_ONS_BG, language=French, numberingPlan= NP_ONS_BG`
			or
			`normalizedNumber=`
			`+492404673406227`
				`<attributes>, <modifications>`

Attribute	Description	Necessity	Example	SPML usage
	`language`: This is the language the media server announcements are played in. It can only be used for single step provisioning. It cannot be exported. `numberingPlan`: This is the numbering plan the business group line will be created in. It can only be used for single step provisioning. It cannot be exported. `normalizedNumber`: This is the phone number in global number format, for example +49897221001 `rateAreaID`: This value of this key is an already existing routing area. It can only be used for single step provisioning, i. e. it cannot be used for bulk provisioning. It cannot be exported. `externalDisplayName`: This key is the caller's display name for external calls. It can only be used for single step provisioning. It cannot be exported.			

Attribute	Description	Necessity	Example	SPML usage
`attributes` (continued)	• `branchOfficeID`: This key is the ID of the branch office that the subscriber will be moved to. If no `branchOfficeID` is specified the resource is provisioned in the default "Main Office" branch. It cannot be exported. The values of externalUnicodeName, internalUnicodeName, featureProfileID, language, numberingPlan, rateAreaID, externalDisplayName and branchOfficeID are stored in the database to handle the phone's attributes.			

3.4.6 Phone4k

The `Phone4k` object represents a phone connected to a HiPath 4000 PBX. Use a `Phone` object instead, if the phone is connected to an OpenScape Voice PBX.

None of the attributes can be modified

Table 8: Attributes of the Phone4k Object

Attribute	Description	Necessity	Example	SPML usage

`alternativeId`	The alternative unique identifier of the phone. It is arbitrary text.	Mandatory	`altId1`
			or
			`Mike's Phone`
				`<id>`

`objectclass`	The constant value of the object that is provisioned. Its value is always `Phone4k`.	Mandatory	`Phone4k`	`<identifierAttributes>`

Attribute	Description	Necessity	Example	SPML usage
`domain`	The name of the domain the phone belongs to. It has to be the same domain as in the associated number plans and dial plans. **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttributes>`
`normalizedNumber`	The normalized number. It is the the phone's unique identifier. The `normalizedNumber` is constructed internally out of extension and numbering plan attributes.	Mandatory	+49897223109	`<attributes>`, `<modifications>`
`isUnique`	Flag indicating whether this phone is unique or not within the numbering scheme of the entire HiPath 4000 network	Optional	`true` or `false`	`<attributes>`, `<modifications>`
`switchId`	The id attribute of a Switch4k object representing the PBX this numbering plan belongs to	Mandatory	`switchtest1`	`<attributes>`, `<modifications>`
`publicExtension`	Public extension, e. g. `12345` for `+498972212345`, if a public numbering plan is used	7896	12345	`<attributes>`, `<modifications>`
`privateExtension`	Private extension, for example `1324` for `2221201234`, if a private numbering plan is used	6	1234	`<attributes>`, `<modifications>`

6 Either you use a public numbering plan or a private numbering plan. If you use a public numbering plan, the `publicExtension` attribute, the `assignedNumbering`

7 `PlanPublicId` attribute and the `assignedDialPlan`

8 `PublicId` attribute are mandatory and the `privateExtension` attribute, the `assignedNumberingPlan`

9 `PrivateId` attribute and the `assignedDial PlanPrivateId` attribute must not be set. If you use a private numbering plan, the `privateExtension` attribute, the `assignedNumbering PlanPrivateId` attribute and the `assignedDialPlanPrivateId` attribute are mandatory and the `publicExtension` attribute, the `assignedNumberingPlan PublicId` attribute and the `assignedDialPlan PublicId` attribute must not be set.

Attribute	Description	Necessity	Example	SPML usage
`virtualNodeId`	The virtual node ID is the representation of the VNR property of the HiPath 4000. If the HiPath 4000 is a VNR, the `virtualNodeId` attribute is `true`, otherwise it is - false.	Optional	false	`<attributes>, <modifications>`
`assignedNumbering PlanPublicId`	The id attribute of the assigned NPPublic object.	6	nppublic2	`<attributes>, <modifications>`
`assignedNumbering PlanPrivateId`	The id attribute of the assigned NPPrivate object.	6	npprivate2	`<attributes>, <modifications>`
`assignedNumbering Plan Unknown`	The id attribute of the assigned NPUnknown object.	6	npunknown2	`<attributes>, <modifications>`
`assignedDialPlan PublicId`	The id attribute of the assigned DPPublic object.	6	dppublic2	`<attributes>, <modifications>`
`assignedDialPlan PrivateId`	The id attribute of the assigned DPPrivate object.	Mandatory, if `assignedNumbering PlanPublicId` is not set	dpprivate2	`<attributes>, <modifications>`
`BCFedId`	The id attribute of the assigned BCFederation object (BCOM federation). This attribute only is set, if the phone is associated to a BC federation. You have a BC federation, if you use more than one BCOM.	Optional	1@offboard	`<attributes>, <modifications>`

3.4.7 `AllocatedPhone`

The `AllocatedPhone` object (foreign phone) describes a phone of an unknown PBX, i e. without monitoring via BCOM, for example for IBM.

Each allocated phone can be assigned to multiple users and each user can be assigned to multiple allocated phones.

Table 9: Attributes of the `AllocatedPhone` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	Unique ID of the allocated phone. A phone number in the global number format is recommended. Use this `id` attribute as the value of the `<id>` element in the `<identifier>` element of the SPML request (`<id>id</id>`). Do not use this `id` attribute in the `<attributes>` element of the SPML request.	Mandatory[10]	`491706315034`	`<id>`[10]
objectclass	The constant value of the object that is provisioned. Its value is always `AllocatedPhone`.	Mandatory	`AllocatedPhone`	`<identifierAttribute>`
`normalized Number`	A dialable phone number in global number format. It is a unique identifier.	Optional[10]	`+491706315034` or `91706315035`	`<attributes>`[10], `<modifications>`[10]
`domain`	The domain the allocated phone belongs to. **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttributes>`
`displayName`	A user defined name for the allocated phone	Optional	`Mobile phone`	`<attributes>`, `<modifications>`

A softphone on a computer device or a desk phone should be an allocated device.

3.4.8 `VoiceMail`

The `VoiceMail` object gives information about a voicemail box a user has.

10 If the `id` attribute is specified and the `normalizedNumber` attribute is not specified, the `id` attribute is used as the `<id>` SPML element of the SPML add request. If the `id` attribute and the `normalizedNumber` attribute are specified, the `id` attribute is ignored and the `normalizedNumber` attribute is used instead as the `<id>` SPML element of the SPML add request. Be sure to use the same attribute as an `<id>` SPML element in an SPML modify or delete request, if you want to modify or delete exactly this `AllocatedPhone` object instance later.

Table 10: Attributes of the `VoiceMail` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	The unique phone number in global number format identifies the voicemail box a user has. This phone number must describe the access number of a voicemail system. Use this `id` attribute as the value of the `<id>` element in the `<identifier>` element of the SPML request (`<id>id</id>`). Do not use this `id` attribute in the `<attributes>` element of the SPML request.	Mandatory	`+49897223101`	`<id>`
objectclass	The constant value of the object that is provisioned. Its value is always `VoiceMail`.	Mandatory	`VoiceMail`	`<identifierAttribute>`
`domain`	The Symphonia domain that the voicemal number belongs to. **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttribute>`
`displayName`	A user defined name for the voicemail system	Optional	OpenScape Xpressions	`<attributes>`, `<modifications>`
`normalizedNumber`	A dialable phone number in global number format. It is a unique identifier.	Mandatory	`+491706315034` or `491706315035`	`<attributes>`, `<modifications>`

3.4.9 `DPPublic`

DPPublic stands for dial plan public.

NOTICE: A numbering plan describes how a phone number is composed, a dial plan describes how a phone number is used.

In case of using a public numbering plan, the GNF format and the `DPPublic` object are used instead of a private numbering plan and the `DPPrivate` object.

The `DPPublic` object describes the office code a phone device has. The office code is a phone number in global number format without the + character and

without the extension number, for example 492404901 is the office code of +492404901450.

IMPORTANT: None of the attributes can be modified after creation of the `DPPublic` object. At least the `subscriberExitCode` should be set. The rest of the layers can be omitted.

Table 11: Attributes of the `DPPublic` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	The unique ID for the dial plan public	Mandatory	`public.492404901` `@system`	<id>
`objectclass`	The constant value of the object that is provisioned. Its value is always `DPPublic`.	Mandatory	`DPPublic`	`<identifierAttribute>`
`domain`	The Symphonia domain that the public dial plan belongs to. **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttribute>`
`internationalPrefix`	International prefix	Mandatory	`00`	`<attributes>`, `<modifications>`
`internationalExitCode`	International exit code	Optional		`<attributes>`, `<modifications>`
`nationalPrefix`	National prefix	Mandatory	49	`<attributes>`, `<modifications>`
`nationalExitCode`	National exit code. If the `nationalExitCode` attribute is set in a `DPPublic` object and if this plan is assigned to at least one phone, the corresponding assigned numbering planof this phone must have set the `areaCode` attribute.	Optional	0	`<attributes>`, `<modifications>`
`subscriberPrefix`	Subscriber prefix	Mandatory	2404	`<attributes>`, `<modifications>`
`subscriberExitCode`	Subscriber exit code	Optional		`<attributes>`, `<modifications>`
`switchId`	The `id` attribute of the switch (Switch8k object or Switch4k objec) the phone belongs to	Mandatory		`<attributes>`, `<modifications>`

3.4.10 DPPrivate

DPPrivate stands for dial plan private.

NOTICE: A numbering plan describes how a phone number is composed, a dial plan describes how a phone number is used.

In case of using a private numbering plan, the DPPrivate object is used instead of a public numbering plan, the GNF format and the DPPublic object.

IMPORTANT: INone of the attributes can be modified after creation of the DPPrivate object. All attributes should be set to have a complete private dial plan.

Table 12: Attributes of the DPPrivate Object

Attribute	Description	Necessity	Example	SPML usage
id	The unique ID for the private dial plan	Mandatory	private.556677 @system	<id>
objectclass	The constant value of the object that is provisioned. Its value is always DPPrivate.	Mandatory	DPPrivate	<identifierAttribute>
domain	The OpenScape UC Application domain the private dial plan belongs to. **NOTE:** OpenScape UC Application can currently only be operated with the domain system.	Mandatory	system	<identifierAttribute>
level2Prefix	Layer 2 prefix as a number	Optional	2	<attributes>, <modifications>
level2ExitCode	Layer 2 exit code as a number.	Optional	9	<attributes>, <modifications>
level1Prefix	Layer 1 prefix as a number	Optional	1	<attributes>, <modifications>
level1ExitCode	Layer 1exit code as a number	Optional	9	<attributes>, <modifications>
level0Prefix	Level 0 prefix as a number	Optional	8	<attributes>, <modifications>
level0ExitCode	Layer 0 exit code as a number	Optional	9	<attributes>, <modifications>

Attribute	Description	Necessity	Example	SPML usage
switchId	The id attribute of the switch (Switch8k object or Switch4k objec) the phone belongs to	Mandatory	2	<attributes>, <modifications>

3.4.11 NPPublic

NPPublic stands for numbering plan public.

NOTICE: A numbering plan describes how a phone number is composed, a dial plan describes how a phone number is used.

coutryCode, areaCode and localDestinationCode should be set in order to have a complete numbering plan.

IMPORTANT: None of the attributes can be modified.

Table 13: Attributes of the NPPublic Object

Attribute	Description	Necessity	Example	SPML usage
id	The unique identifier of the public numbering plan	Mandatory	public.492404901@system	<id>
objectclass	The constant value of the object that is provisioned. Its value is always NPPublic.	Mandatory	NPPublic	<identifierAttribute>
domain	The name of the domain the numbering plan belongs to **NOTE:** OpenScape UC Application can currently only be operated with the domain system.	Mandatory	system	<identifierAttribute>
countryCode	Country code as a number, for example 49 for Germany	Mandatory	49	<attributes>, <modifications>
areaCode	Area code as a number, for example 89 for Munich	Optional	89	<attributes>, <modifications>

Attribute	Description	Necessity	Example	SPML usage
`localDestinationCode`	Local DPPublic (office code)	Mandatory	`722`	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
`overlap`	Overlap as a number. The overlap indicates the amount of digits in a phone number that are not only used as a part of the local destination code but also used as a part of the extension. Set the `overlap` attribute to its default value `0`, if you are not sure whether you need an overlap or when you are sure not to use an overlap. The overlap originates from the US phone number scheme with a fix length of 10 digits. Example: `1-aaa-bbb-cccc` with • `1`: Country code (CC) • `aaa`: Area code (AC) • `bbb`: Local destination code (LDC) • `cccc`: Extension `cccc` only supports 9999 extensions per default. In order to support more extensions, the last digit of the local destination codes could not only be used as the last digit of the local destination code but also be used as the first digit of the extension. Thus,	Mandatory	Default: `0`	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
	you can support 99999 extensions. In this case the overlap is 1. Example: `1-503-736-1234` With an overlap of 1, the local destination code is 736 and the extension is 61234. With an overlap of 2, you could support 999999 extensions.			
`switchId`	The `id` attribute of the switch (Switch8k object or Switch4k objec) the phone belongs to	Mandatory	`switchtest1`	`<attributes>`, `<modifications>`

3.4.12 `NPPrivate`

NPPrivate stands for numbering plan private.

NOTICE: A numbering plan describes how a phone number is composed, a dial plan describes how a phone number is used.

`layer0Code`, `layer1Code` and `layer2Code` should be set in order to have a complete numbering plan.

IMPORTANT: At least `layer0Code` should be set and the rest of the layers could be omitted. However, if they are set, they have to be in an ascending order from level 0 to level 2.None of the attributes can be modified.

Table 14: Attributes of the `NPPrivate` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	The unique identifier of the private numbering plan	Mandatory	`private.556677@system`	<id>

Attribute	Description	Necessity	Example	SPML usage
`objectclass`	The constant value of the object that is provisioned. Its value is always `NPPrivate`.	Mandatory	`NPPrivate`	`<identifierAttribute>`
`domain`	The name of the domain the numbering plan belongs to **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttribute>`
`level2Code`	Layer 2 code as a number	Optional	`22`	`<attributes>`, `<modifications>`
`level1Code`	Layer 1 code as a number	Optional, but mandatory if `layer2Code` is set	`21`	`<attributes>`, `<modifications>`
`level0Code`	Layer 0 code as a number	Mandatory	`20`	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
`overlap`	Overlap as a number. The overlap indicates the amount of digits in a phone number that are not only used as a part of the local destination code but also used as a part of the extension. Set the `overlap` attribute to its default value `0`, if you are not sure whether you need an overlap or when you are sure not to use an overlap. The overlap originates from the US phone number scheme with a fix length of 10 digits. Example: `1-aaa-bbb-cccc` with • `1`: Country code (CC) • `aaa`: Area code (AC) • `bbb`: Local destination code (LDC) • `cccc`: Extension `cccc` only supports 9999 extensions per default. In order to support more extensions, the last digit of the local destination codes could not only be used as the last digit of the local destination code but also be used as the first digit of the extension. Thus,	Mandatory	Default: `0`	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
	you can support 99999 extensions. In this case the overlap is 1. Example: `1-503-736-1234` With an overlap of 1, the local destination code is 736 and the extension is 61234. With an overlap of 2, you could support 999999 extensions.			
`switchId`	The `id` attribute of the switch (Switch8k object or Switch4k objec) the phone belongs to	Mandatory	`switchtest1`	`<attributes>`, `<modifications>`

3.4.13 NPUnknown

NPUnknown stands for numbering plan unknown.

NOTICE: A numbering plan describes how a phone number is composed, a dial plan describes how a phone number is used.

IMPORTANT: None of the attributes can be modified.

Table 15: Attributes of the `NPUnknown` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	The unique identifier of the unknown numbering plan	Mandatory	`npunknown1`	`<id>`
`objectclass`	The constant value of the object that is provisioned. Its value is always `NPUnknown`.	Mandatory	`NPUnknown`	`<identifierAttribute>`

Attribute	Description	Necessity	Example	SPML usage
`domain`	The name of the domain the numbering plan belongs to **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttribute>`
`nodeCD`	Node CD number	Mandatory	`2245`	`<attributes>`, `<modifications>`
`overlap`	Overlap as a number	Optional	Default: `0`	`<attributes>`, `<modifications>`
`switchId`	The `id` attribute of the switch (Switch8k object or Switch4k objec) the phone belongs to	Mandatory	`switchtest1`	`<attributes>`, `<modifications>`

3.4.14 `Switch8k`

All attributes in the following table can be changed.

Table 16: Attributes of the `Switch8k` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	The unique ID for the OpenScape Voice PBX. .	Mandatory	`HP8k_V50`	`<id>`
`objectclass`	The constant value of the object that is provisioned. Its value is always `Switch8k`.	Mandatory	`Switch8k`	`<identifierAttribute>`
`version`	Version of the OpenScape Voice	Mandatory	`2`	`<attributes>`, `<modifications>`
`name`	Name of the OpenScape Voice that is shown for example in the CMP	Mandator	s8ktest	`<attributes>`, `<modifications>`
`CSTAAddressMain`	IP address under which the CSTA interface of the OpenScape Voice can be reached.	Mandatory	`10.0.3.0`	`<attributes>`, `<modifications>`
`CSTAPortMain`	Port that is used for the CSTA communication with the OpenScape Voice.	Mandatory	`1040`	`<attributes>`, `<modifications>`

Attribute	Description	Necessity	Example	SPML usage
CSTAAddressAlternative	Alternative IP address under which the CSTA interface of the OpenScape Voice can be reached.	Optional	10.0.3.1	<attributes>, <modifications>
CSTAPortAlternative	Alternative Port that is used for the CSTA communication with the OpenScape Voice.	Optional	1041	<attributes>, <modifications>
AdminAddressMain	IP address under which the administration interface of the OpenScape Voice can be reached.	Optional	10.0.3.2	<attributes>, <modifications>
AdminPortMain	Port that is used for the administration communication with the OpenScape Voice.	Optional	1042	<attributes>, <modifications>
AdminAddressAlternative	Alternative IP address under which the administration interface of the OpenScape Voice can be reached.	Optional	10.0.3.3	<attributes>, <modifications>
AdminPortAlternative	Alternative Port that is used for the administration communication with the OpenScape Voice.	Optional	1043	<attributes>, <modifications>

3.4.15 Switch4k

IMPORTANT: The Switch4k and id attributes cannot be modified after creation of the Switch4k object.

Table 17: Attributes of the Switch4k Object

Attribute	Description	Necessity	Example	SPML usage
id	The unique identifier of the HiPath 4000	Mandatory	switchtest1	<id>
objectclass	The constant value of the object that is provisioned. Its value is always Switch4k.	Mandatory	Switch4k	<identifierAttribute>
version	Version of the HiPath 4000	Mandatory	5.0	<attributes>, <modifications>

Attribute	Description	Necessity	Example	SPML usage
`name`	Name of the HiPath 4000	Mandatory	`HiPath_Testlab`	`<attributes>, <modifications>`
`CSTAAddressMain`	Computer name or IP address of the HiPath 4000	Mandatory	`swi.company.com`	`<attributes>, <modifications>`
`CSTAPortMain`	Port of the HiPath 4000	Mandatory	`900`	`<attributes>, <modifications>`
`applId`	Application identifier	Mandatory	`5`	`<attributes>, <modifications>`
`subApplId`	Subapplication identifier	Mandatory	`29`	`<attributes>, <modifications>`

3.4.16 `BCFederation`

A BC federation is an assignment of resources to instances of services. A BC federation is used to assign a device (logical or physical phone number) to BCOM instances processing these phone numbers. On one hand, this allows to created specific instances serving specifically an OpenScape Voice, a HiPath 4000 or a media server. On the other hand, this is used to create an assignment for several similar instances running on different computers like in the large deployment where several media servers might be used.

A BC federation is called federation or BCom federation as well.

IMPORTANT: None of the attributes can be modified.

Table 18: Attributes of the `BCFederation` Object

Attribute	Description	Necessity	Example	SPML usage
`id`	The unique ID for the federation. .	Mandatory	`BC1` or `2@offboard`	`<id>`
`objectclass`	The constant value of the object that is provisioned. Its value is always `BCFederation`.	Mandatory	`BCFederation`	`<identifierAttribute>`
`bcomHomeReponsibility`	ID of the BCOM service instance defined by BCOM	Mandatory	The value always is `1`.	`<attributes>, <modifications>`
`switchId`	ID of the PBX the phone belongs to	Mandatory	switchtest2	`<attributes>, <modifications>`

3.4.17 ExternalIdentifier

A user may be represented not only in an OpenScape UC Application system but also in an external system, for example a Microsoft Exchange system or a Lotus Domino system. The ExternalIdentifier object is a link between the user's representation in these two systems. The externalId attribute is the users's ID in the external system and the assignedUserIdentity attribute is the user's ID in the OpenScape UC Application (see identity attribute in User).

NOTICE: The ExternalIdentifier object can only be used to identify users but no items (contacts, phones, etc.).

NOTICE: Several ExternalIdentifier objects can be assigned to the same OpenScape UC Application user. This is needed if the user is for example in an OpenScape UC Application, a Microsoft Exchange and a Lotus Domino system.

Table 19: Attributes of the ExternalIdentifier Object

Attribute	Description	Necessity	Example	SPML usage
externalId	The unique ID of a user in the external system, for example a Lotus Domino system. If an SPML request tries to change attributes of an ExternalIdentifier object, the externalId attribute is not changed, if an ExternalIdentifier object with the same assignedUserIdentity and the same assignedExternalSystem already is in the database. However, any other attributes can be changed.	Mandatory	CN= Peter Petersson/ O= Simen	<id>
objectclass	The constant value of the object that is provisioned. Its value is always ExternalId. Mandatory value: ExternalId	Mandatory	ExternalId	<identifierAttribute>

Attribute	Description	Necessity	Example	SPML usage
`assignedUserIdentity`	identity attribute of a User object representing an OpenScape UC Application user. This User object must have been created before the `assignedUserIdentity` attribute of an `ExternalIdentifier` object can correspond to the `User` object. This might be done in the same SPML request.	Mandatory	`peter.petersson@system`	11
`assignedExternalSystemId`	ID of the external system the user with the external identifier belongs to for example a groupware system such as Microsoft Exchange. This attribute has to be defined by a previous administration action.	Mandatory	`Sys2`	11

11 `<identifierAttributes>` in an SPML delete request, `<attributes>` or `<modifications>` in all other SPML requests

Attribute	Description	Necessity	Example	SPML usage
`isContactMaster`	This attribute defines whether the external system identified by the `externalId` attribute is the source system for contact information of the OpenScape UC Application user identified by the `assignedUserIdentity` attribute. See Contact for more information about the `Contact` object. If several `ExternalIdentifier` objects are assigned to an OpenScape UC Application user, only one of these `ExternalIdentifier` objects can be the link to the source system for contact information of this user. Possible options are: • `true`: The external system identified by the `externalId` attribute is the source system for contact information of the OpenScape UC Application user identified by the `assignedUserIdentity` attribute. • `false`: The external system identified by the `externalId` attribute is not the source system for contact information of the OpenScape UC Application user identified by the `assignedUserIdentity` attribute.	Optional	`false`	`<attributes>`, `<modifications>`
`isProvisioningMaster`	This attribute is completely analog to the `isContactMaster` attribute but it does not indicate the source system for contact information (see Contact) but user information (see User).	Optional	`true`	`<attributes>`, `<modifications>`
`additionalInfo`	Comma separated list of key-value pairs to set customized attributes, for example key1=value1, key2=value2	Optional		`<attributes>`, `<modifications>`

3.4.18 QueueDevice

The QueueDevice object represents a call queue.

Table 20: Attributes of the QueueDevice Object

Attribute	Description	Necessity	Example	SPML usage
id	The unique identifier of the queue	Mandatory[12]	+90312292138	<id>[12]
objectclass	The constant value of the object that is provisioned. Its value is always ExternalIdentifier.	Mandatory	QueueDevice	<identifierAttribute>
domain	Symphonia domain the queue belongs to **NOTE:** OpenScape UC Application can currently only be operated with the domain system.	Mandatory	system	<identifierAttribute>
displayName	The display name of the queue in the CMP	Mandatory	queueName	<attributes>, <modifications>
normalizedNumber	The normalized number of the queue device.	Optional[12]		<attributes>[12], <modifications>[12]

3.4.19 Role

A Role object describes the profiles a user is assigned to. A Role object cannot be changed after creation, imported or updated. It can only be exported by a search request.

12 If the id attribute is specified and the normalizedNumber attribute is not specified, the id attribute is used as the <id> SPML element of the SPML add request. If the id attribute and the normalizedNumber attribute are specified, the id attribute is ignored and the normalizedNumber attribute is used instead as the <id> SPML element of the SPML add request. Be sure to use the same attribute as an <id> SPML element in an SPML modify or delete request, if you want to modify or delete exactly this QueueDevice object instance later.

Table 21: Attributes of the `Role` Object

Attribute	Description	Necessity	Example	SPML usage
name	Profile name Allowed values: • `Standard: E/A-Cockpit` • `Premium: E/A-Cockpit` • `Professional: OpenScape UC App` • `Essential: OpenScape UC App` • `Team: OpenScape UC App` • `Customer Administrator: OpenScapeVoice` • `Basic Administrator: OpenScapeVoice` • `Super Administrator: OpenScapeVoice` • `Configuration: RGAssistant` • `Supervision: RGAssistant` • `Administration: RGAssistant` • `Administrator: Symphonia` • `CustomerAdministrator: Symphonia` • `ServiceTechnician: Symphonia`	Optional	`Team:OpenScape UC App`	`<id>`
objectclass	The constant value of the object that is provisioned. Its value is always `Role`.	Mandatory	`Role`	`<identifierAttribute>`
domain	The name of the domain the role belongs to **NOTE:** OpenScape UC Application can currently only be operated with the domain `system`.	Mandatory	`system`	`<identifierAttribute>`
application	Application	Optional	`OpenScape UC Application V`	`<attributes>, <modifications>`
permissions	Permission of this role	Optional	`OpenScape Enterprise V Team User`	`<attributes>, <modifications>`

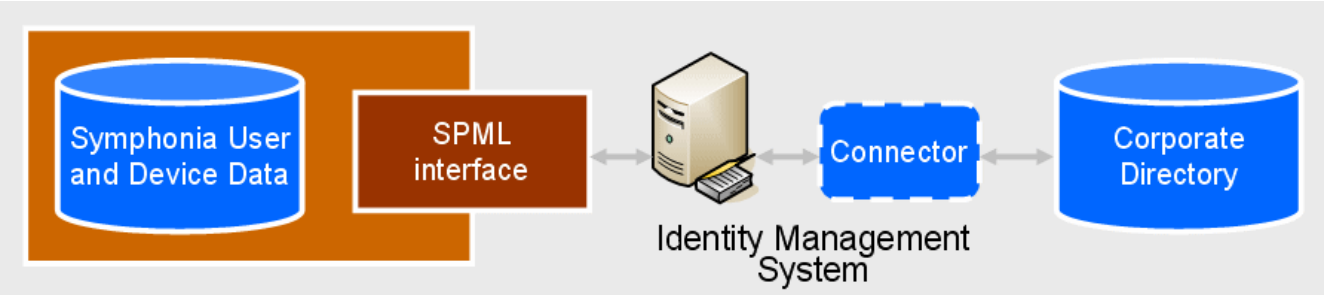
4 Provisioning LDAP Directories

4.1 Overview

4.1.1 General

The Open SOA services used are part of the overall provisioning infrastructure and provide the means to import data to the OpenScape UC Application by using Service Provisioning Markup Language (SPML). SPML is supported by OpenSOA via the incorporation of DirX Identity Light Framework of Unify GmbH & Co. KG.

The OpenScape UC Application does not provide an out of the box connector to the corporate directory. The connector is also responsible for the correct attribute mapping.



Initial or general system configuration (e. g. the CSTA link configuration) is not possible via this interface.

The OpenScape UC Application SPML interface does not extend to the provisioning of users into the communication system or Voice over IP (VOIP) system.

The interface supports SMPL V1.0. It is a request/reply based interface as defined by SPML V1.0.

4.1.2 Workflow

The customer side consists of the LDAP directory (identity domain; data source; source system) and an SPML client (LDAP connector) retrieving data from the LDAP directory and forwarding it to the OpenScape UC Application side.

Data transfer can only be done in this direction, i. e. only import of data to the OpenScape UC Application is possible, no export to the LDAP directory can be done.

The data is received by the SPML responder on the OpenScape UC Application side (TS, target system). It is the SPML endpoint of OpenSOA, which incorporates and extends DirX Identity Light Framework to perform the actual SPML request/response handling. As a servlet it resides within the Tomcat servlet container. The data is forwarded to the user service residing within an OSGi container. It looks for the data in the Symphonia database, the data target. If the data is found, it is modified, if not it is created.

DirX Identity Light Framework can be executed as a stand-alone executable or as a servlet. It is Java-based meaning that the servlet variant needs to be deployed on a Tomcat server while the executable is a JAR file that can run using a Java runtime environment installation 1.5.x or later or a JDK installation 1.5.x or later.

The data to be provisioned is user data, password data and contact data.

The procedure is automatic and cannot be changed by the user. No Add/Modify/Delete/Search/Schema SPML request can be executed directly.

4.1.3 Access to the SPML Interface

Depending on the server configuration the SPML interface is reachable via a HTTP or HTTPS interface at

```
http://<IP address of the OpenScape UC Application
 application computer>:8099/symphonia-
spmlresponder/services/SpmlSoapService
```

The application computer is the computer where the backend services of the OpenScape UC Application have been installed.

The HTTP basic authentication mechanism has to be used (user name and password).

A well formed SPML document is sent via this interface to the OpenScape UC Application executing the specified operation (creation, modification or deletion).

4.1.4 Supported Objects and Requests

This section lists the objects that can be manipulated by the SPML interface and the SPML requests that can be executed using the SPML interface.

Table 22: Supported Requests

Object	Source System/Destination System													
	Any except LDAP Directory							LDAP Directory						
	Add	Modify	Delete	Search	Batch	Status	Cancel	Add	Modify	Delete	Search	Batch	Status	Cancel
`Domain`	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
`User`	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗

Object	Source System/Destination System													
	Any except LDAP Directory							LDAP Directory						
	Add	Modify	Delete	Search	Batch	Status	Cancel	Add	Modify	Delete	Search	Batch	Status	Cancel
Contact	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Phone	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Phone4K	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
AllocatedPhone	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
VoiceMail	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
DPPublic	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
DPPrivate	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
NPPrivate	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
NPPublic	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
NPUnknown	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Switch8k	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Switch4k	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
BCFederation	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
ExternalIdentifier	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
QueueDevice	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Role	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗

4.2 Preconditions

The following objects have to be configured via the CMP before running provisioning:

Mandatory

- User profiles
- Predefined profiles exist e. g. for the system and customer administrator. Typical users do not need a configured profile.
- If voicemails must be assigned to users, one or more voicemails must be defined via the CMP.
- If queue devices are to be assigned to users, one or more queue devices must be defined via the CMP. You can do so by selecting the Administrator user, opening **Configuration** > **Unified Communications** > **Accounts** >

List > **<User>** > **Resources** > **Queue** > **Number** and adding the queue devices you wish to use.

4.3 LDAP Provisioning Tool

The implementation of the DirX Identity Light Framework in its stand-alone version is the LDAP Provisioning Tool. This tool can be used on both a Windows or Linux operating system. It does not have to be located on the OpenScape UC Application installation and it can be run from any system that has direct network access to both the LDAP directory and the OpenScape UC Application. Scheduling capabilities are not provided as part of the tool but being a command line tool it can be used as part of a cronjob on Linux or scheduled task on Windows to achieve periodic synchronization.

4.4 Installation on the OpenScape UC Application Side

The required components of the DirX Identity Lightweight Framework are installed by executing the following command on the application computer (OpenScape UC Applicaton back-end):

```
zypper in scs_provisioning-ldapprov<Version>
```

You find the scs_provisioning-ldapprov<Version> RPM package on the installation DVD of the OpenScape UC Application. See the installation guide *OpenScape UC Application, Installation and Upgrade* for details about the installation and using zypper.

This installation creates the `/opt/siemens/sw_repository/scs_-provisioning-ldapprov.zip` file. The contents of this file is autmatically extracted into the `/opt/siemens/servicetools` directory.

4.5 Configuration and Execution of the Import on SLES

If you want to run the import on a computer with a SLES operating system, execute the following steps:

4.5.1 Preconditions

1) Execute the following command, to check whether JRE 1.5.x or higher or JDK 1.5.x or higher is installed.

```
zypper se --match-any
```

A succesful output is e. g.:

```
S | Name | Summary | Type
--+---------------------------
+-----------------------------------------------------
| java-1_4_2-ibm | IBM(R) Runtime Environment for Linux, Java(TM) 2 Technol
| java-1_4_2-ibm | IBM(R) Runtime Environment for Linux, Java(TM) 2 Technol
| java-1_6_0-ibm | Java(TM) 6 Runtime Environment
| java-1_6_0-ibm | Java(TM) 6 Runtime Environment
| java-1_6_0-ibm-fonts | Java(TM) 2 Runtime Environment
| java-1_6_0-ibm-jdbc | JDBC/ODBC bridge driver for java-1.6.0-ibm
```

```
| sblim-cim-client2-javadoc | Javadoc for sblim-cim-client2
| timezone-java | Timezone Descriptions
| timezone-java | Timezone Descriptions s
```

A letter `i` in the first column indicates that an RPM package is installed. Use the `rpm in <Package name>` command to install an RPM package.

2) If step 1 did not succeed, execute the `echo $PATH` command to check whether the path variable is set correctly.

4.5.2 Configuration

1) Copy the configuration XML files from
`/opt/siemens/servicetools/ldapprovisioning/configurations`

to the following directory:
`/opt/siemens/servicetools/ldapprovisioning`

Copy only those configuration files that are needed for the specific data that you want to import from the LDAP directory:

a) `conf-AllocatedPhone.xml` for importing mobile phones

IMPORTANT: This file must be copied unless you do not want to import data about the users' mobile phones **and**

- you either change the provided `conf-Users.xml` file or
- you are **absolutely sure** that **none** of the users in the LDAP directory you want to import has a mobile phone.

The mentioned modification of the `conf-Users.xml` file is the removal of the attribute mapping for the `mobile` attribute.

If you do not copy and use this file (i. e. you do not execute step 2a on page 93) but mobile phones are found in the LDAP directory, the following error message is shown:

2010-11-25 16:08:23,312 ERROR
[com.siemens.dxm.join.util.Connection] ERR(JOIN535): Creation of

object "peterone" failed; error: SpmlException: AddRequest for peterone failed: Resource '+306900000012' does not exist

b) `conf-Users.xml` for importing users (e. g. home time zone, language, resources etc.)

NOTICE: It is `conf-User`**`s`**`.xml`, **not** `conf-User`**`.`**`xml`!

Among other things, the provided `conf-Users.xml` file is configured to check whether there is information about mobile phones in the LDAP directory. If there is a mobile phone, it is imported.

If a user in the LDAP directory has a mobile phone number that does not exist in the OpenScape UC Application, the usage of the `conf-Users.xml` file (see step 2b on page 93) fails.

The attributes of a user are e. g. shown at **User Management > Administration > Users & Resources** > <**User> > General** in the CMP.

If a user's phone number that is used in the LDAP directory exists as a resource in the CMP, this phone number is set as a ONS.

c) `conf-Contacts.xml` for importing a user's contact information such as gender salutation, title, business phones, home phone, mobile phone, fax number etc. This file is e. g. needed, if it should be possible to search with the OpenScape UC Application OpenScape Web Client 1.0 for a user to add your contacts. The attritutes of a contact are shown on the **Contact** tab in the CMP.

d) `conf-ExternalId.xml` is used for assigning an identifier of a user in an external system (here: LDAP directory) to an identifier in the OpenScape UC Application system and vice versa. This is e. g. needed, if users should be able to login to the OpenScape UC Application OpenScape Web Client 1.0 with SSO.

IMPORTANT: The SSO configuration must have been done before.

---

e) `conf-User_Update.xml` for updating user information from the LDAP directory to the OpenScape UC Application

Using `conf-User_Update.xml` (see step 2e on page 93) updates **ALL** users that had been imported from the LDAP directory before. Example: You have 10 users in the LDAP directory. You update one of these users in the LDAP directory. Using `conf-User_Update.xml` updates all 10 users that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory. As a result, the password, locale and other attributes are reset according to the settings in `conf-User_Update.xml`.

f) `conf-Contacts_Update.xml` for updating a user's contact information from the LDAP directory to the OpenScape UC Application.

Using `conf-Contacts_Update.xml` (see step 2f on page 93) updates **ALL** contact information that had been imported from the LDAP directory before. Example: You have 10 contacts in the LDAP directory. You update one of these contacts in the LDAP directory. Using `conf-Contacts_-Update.xml` updates all 10 contacts that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory.

2) Edit the copied configuration files according to the instructions in Basic Mapping on page 97 and Configuration File Reference on page 115.
3) The `/opt/siemens/servicetools/ldapprovisioning/lib/-log4j.xml` file specifies the logging level and the subdirectory where the log files are available, e. g.:

```
/opt/siemens/servicetools/ldapprovisioning/testresources/testSoap/log.txt
```

The `log4j.xml` file uses two appender elements with the attribute `name` having the values `FILE` and `FILE_XML` to define the log files.

Example:

```
<appender name="FILE" class="org.apache.log4j.RollingFileAppender">
<param name="Threshold" value="DEBUG"/>
<param name="File" value="testresources/testSoap/log.txt" />
…
<appender name="FILE_XML" class="org.apache.log4j.RollingFileAppender">
<param name="Threshold" value="DEBUG"/>
<param name="File" value="testresources/testSoap/log.xml" />
```

The `RollingFileAppender` value causes the log files to be backed up when they reach a certain size.

The following log levels are supported:

- `DEBUG`

  The `DEBUG` level designates fine-grained informational events that are most useful to debug an application.
- `INFO`

  The `INFO` level designates informational messages that highlight the progress of the application at coarse-grained level.
- `WARN`

  The `WARN` level designates potentially harmful situations.
- `ERROR`

  The `ERROR` level designates error events that might still allow the application to continue running.
- `FATAL`

  The `FATAL` level designates very severe error events that will presumably lead the application to abort.

4.5.3 Execution

1) Execute the following command:
   `cd /opt/siemens/servicetools/ldapprovisioning`

2) For each configuration file you have copied and edited in step 1 on page 90 and 2 on page 92, the corresponding import command stated below must be executed.

IMPORTANT: Execute the commands **in exactly the following order**. Omit the commands that are not needed.

---

a) `bash ldapprov.sh conf-AllocatedPhone.xml`
b) `bash ldapprov.sh conf-Users.xml`

Use `conf-User`**`s`**`.xml`, **not** `conf-User`**`.`**`xml`!

c) `bash ldapprov.sh conf-Contacts.xml`
d) `bash ldapprov.sh conf-ExternalId.xml`
e) `bash ldapprov.sh conf-User_Update.xml`

IMPORTANT: Using `conf-User_Update.xml` updates **ALL** users that had been imported from the LDAP directory before. Example: You have 10 users in the LDAP directory. You update one of these users in the LDAP directory. Using `conf-User_Update.xml` updates all 10 users that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory. As a result, the password, locale and other attributes are reset according to the settings in `confUser_Update.xml`.

---

f) `bash ldapprov.sh conf-Contacts_Update.xml`

IMPORTANT: Using `conf-Contacts_Update.xml` updates **ALL** contact information that had been imported from the LDAP directory before. Example: You have 10 contacts in the LDAP directory. You update one of these contacts in the LDAP directory. Using `conf-Contacts_Update.xml` updates all 10 contacts that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory.

---

3) Output

When the import is executed, the full log appears on the screen identifying problems and giving detailed information. E. g. the following message indicates that the import is complete and all entries have been imported from the LDAP directory (Identity domain; source system) to the OpenScape UC Application (TS; target system):

```
****** Synchronisation 'Identity Domain --> TS'
 successfully terminated."
```

You see e. g. a new user or new contact information at **Users & Resources** > **User Administration** > **Users** in the CMP.

4.6 Configuration and Execution of the Import on Windows

If you want to run the import on a computer with a Windows operating system, execute the following steps:

4.6.1 Preconditions

1) Execute the following command, to check whether JRE 1.5.x or higher or JDK 1.5.x or higher is installed.
   ```
   java -version
   ```
   A successful output is e. g.:
   ```
   Java version "1.6.0_17"
   Java(TM) SE Runtime Environment (build 1.6.0_17-b04)
   Java HotSpot(TM) Client VM (build 14.3-b01, mixed mode,
    sharing)
   ```
2) If step 1 did not succeed, execute the following sub-steps to add the Java executables to the **Path** variable.
   a) Open **Start** > **Settings** > **Control Panel** > **System**.
   b) Click the **Advanced** tab.
   c) Click the **Environment Variables** button.
   d) Click the **Path** variable in the **System variables** area.
   e) Click the **Edit** button.
   f) Add the path to your EXE file, e. g.:
      ```
      C:\Program Files\Java\jdk1.6.0_17\bin;
      ```
   g) Click the **OK** button.
   h) Click the **OK** button.
   i) Click the **OK** button.

4.6.2 Configuration

1) Having executed the zypper command for installing the RPM package (see above), copy the
   `/opt/siemens/sw_repository/scs_provisioning-ldapprov.zip`
   file from the application computer to a local windows machine by using e. g. Secure Shell.
2) Extract the `scs_provisioning-ldapprov.zip` file to `C:\LDAP`.

3) Copy the configuration XML files from `C:\LDAP\configurations` to `C:\LDAP`. Copy only those configuration files that are needed for the specific data that you want to import from the LDAP directory:

   a) `conf-AllocatedPhone.xml` for importing mobile phones associated to a user.

      IMPORTANT: This file must be copied unless you do not want to import data about the users' mobile phones **and**

      - you either change the provided `conf-Users.xml` file or
      - you are **absolutely sure** that **none** of the users in the LDAP directory you want to import has a mobile phone.

      The mentioned modification of the `conf-Users.xml` file is the removal of the attribute mapping for the `mobile` attribute.

      If you do not copy and use this file (i. e. you do not execute step 2a on page 99) but mobile phones are found in the LDAP directory, the following error message is shown:

      ```
      2010-11-25 16:08:23,312 ERROR
      [com.siemens.dxm.join.util.Connection] ERR(JOIN535):

      Creation of object "peterone" failed; error:
      SpmlException: AddRequest for peterone

      failed: Resource '+306900000012' does not exist
      ```

   b) `conf-Users.xml` for importing users (e. g. home time zone, language, resources etc.)

      NOTICE: It is `conf-User`**s**`.xml`, **not** `conf-User.xml`!

      Among other things, the provided `conf-Users.xml` file is configured to check whether there is information about mobile phones in the LDAP directory. If there is a mobile phone number, it is imported.

      If a user in the LDAP directory has a mobile phone number that does not exist in the OpenScape UC Application, the usage of the `conf-Users.xml`f file (see 2b on page 99) fails.

      The attributes of a user are e. g. shown at **User Management > Administration > Users & Resources** > <**User> > General** in the CMP.

      If a user's phone number that is used in the LDAP directory exists as a resource in the CMP, this phone number is set as a ONS.

   c) `conf-Contacts.xml` for importing a user's contact information such as gender salutation, title, business phones, home phone, mobile phone, fax number etc. This file is e. g. needed, if it should be possible to search with the OpenScape UC Application OpenScape Web Client 1.0 for a user to add your contacts. The attritutes of a contact are shown at **User Management > Administration > Users & Resources** > <**User> > General > Contact** in the CMP.

   d) `conf-ExternalId.xml` is used for assigning the identifier of a user in an external system (here: LDAP directory) to the identifier of the OpenScape UC Application system and vice versa. This is e. g. needed,

if users should be able to login to the OpenScape UC Application OpenScape Web Client 1.0 with SSO.

IMPORTANT: The SSO configuration must have been done before.

---

e) `conf-User_Update.xml` for updating user information from the LDAP directory to the OpenScape UC Application

---

NOTICE: Using `conf-Contacts_Update.xml` (see step 2e on page 99) updates **ALL** contact information that had been imported from the LDAP directory before. Example: You have 10 contacts in the LDAP directory. You update one of these contacts in the LDAP directory. Using `conf-Contacts_Update.xml` updates all 10 contacts that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory.

---

f) `conf-Contacts_Update.xml` for updating a user's contact information from the LDAP directory to the OpenScape UC Application.

IMPORTANT: Using `conf-Contacts_Update.xml` (see step 2f on page 99) updates **ALL** contact information that had been imported from the LDAP directory before. Example: You have 10 contacts in the LDAP directory. You update one of these contacts in the LDAP directory. Using `conf-Contacts_Update.xml` updates all 10 contacts that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory.

---

4) Edit the copied configuration files according to the instructions in Basic Mapping on page 97 and Configuration File Reference on page 115.

5) Create the `C:\LDAP\logs` directory.

6) Create the `C:\LDAP\logs\log.txt` file.

7) Open `C:\LDAP\lib\log4j.xml` in an editor.

8) Replace

```
value="${SYMPHONIA_LOG}/LDAPProvisioning.log"
```

in line 31 by the following text:

```
value="logs/log.txt"
```

You have to do this in order to get logs during import execution and use `C:\LDAP\logs\log.txt` that you have created in step 6.

9) The `log4j.xml` file uses two appender elements with the attribute `name` having the values `FILE` and `FILE_XML` to define the log files.

Example:

```
<appender name="FILE" class="org.apache.log4j.RollingFileAppender">
<param name="Threshold" value="DEBUG"/>
<param name="File" value="testresources/testSoap/log.txt" />
…
<appender name="FILE_XML" class="org.apache.log4j.RollingFileAppender">
<param name="Threshold" value="DEBUG"/>
```

```
<param name="File" value="testresources/testSoap/log.xml" />
```

The `RollingFileAppender` value causes the log files to be backed up when they reach a certain size.

The following log levels are supported:

- `DEBUG`

  The `DEBUG` level designates fine-grained informational events that are most useful to debug an application.
- `INFO`

  The `INFO` level designates informational messages that highlight the progress of the application at coarse-grained level.
- `WARN`

  The `WARN` level designates potentially harmful situations.
- `ERROR`

  The `ERROR` level designates error events that might still allow the application to continue running.
- `FATAL`

  The `FATAL` level designates very severe error events that will presumably lead the application to abort.

4.6.3 Execution

1) Open **Start** > **Run**.

a) Enter `cmd` to open a command shell.

b) Enter `cd C:\LDAP`, i. e. change to the directory `ldapprov.bat` is in.

2) For each configuration file you have copied and edited in step 3 and step 4, the corresponding import command stated below must be executed in this command shell.

IMPORTANT: Execute the commands **in exactly the following order**. Omit the commands with the configuration files that are not needed.

---

a) `ldapprov.bat conf-AllocatedPhone.xml`

b) `ldapprov.bat conf-Users.xml`

Use `conf-Use`**`rs`**`.xml`, **not** `conf-User.xml`!

c) `ldapprov.bat conf-Contacts.xml`

d) `ldapprov.bat conf-ExternalId.xml`

e) `ldapprov.bat conf-User_Update.xml`

IMPORTANT: Using `conf-User_Update.xml` updates **ALL** users that had been imported from the LDAP directory before. Example: You have 10 users in the LDAP directory. You update one of these users in the LDAP directory. Using `conf-User_Update.xml` updates all 10 users that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory. As a

result, the password, locale and other attributes are reset according to the settings in `conf-User_Update.xml`.

f) `ldapprov.bat conf-Contacts_Update.xml`

IMPORTANT: Using `conf-Contacts_Update.xml` updates **ALL** contact information that had been imported from the LDAP directory before. Example: You have 10 contacts in the LDAP directory. You update one of these contacts in the LDAP directory. Using `conf-Contacts_Update.xml` updates all 10 contacts that had been imported before from the LDAP directory into the OpenScape UC Application not only the one you have updated in the LDAP directory.

3) You can open the `C:\LDAP\logs\log.txt` file and delete its content, if the logs are not interesting any more for debugging.

4) Output

When the import is executed, the full log appears on the screen identifying problems and giving detailed information. E. g. the following message indicates that the synchroniziation is complete and all entries have been imported from the LDAP directory (Identity domain; source system) to the OpenScape UC Application (TS; target system):

```
****** Synchronisation 'Identity Domain --> TS'
 successfully terminated."
```

You see e. g. a new user or new contact information at **Users & Resources** > **User Administration** > **Users** in the CMP.

NOTICE: If the import is not successful, execute the instructions described in Preconditions on page 93.

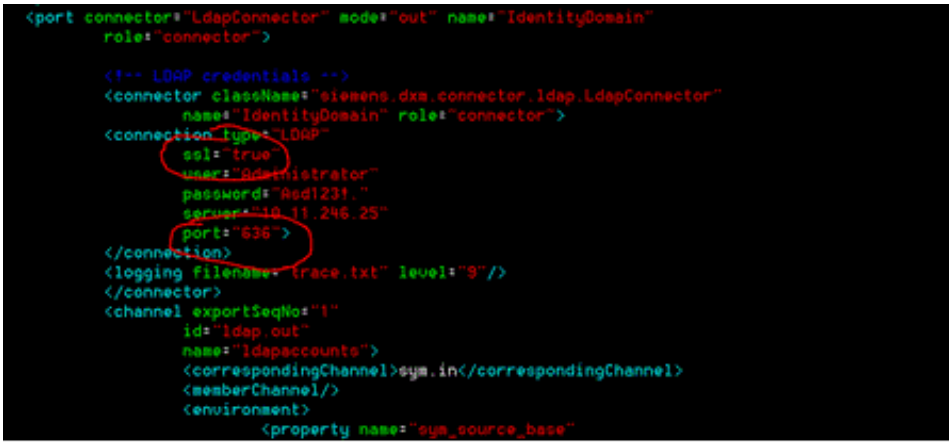
4.7 Basic Mapping

The basic mapping is determined by the configuration file, an XML file. In this section, only the most relevant XML elements are described. Please read Configuration File Reference on page 115 for more details.

4.7.1 Configuring a secure LDAP connection

In order to make an LDAP connection secure there are two flags that need to be changed in the active configuration file (e.g. conf-Users.xml). The SSL flag should be true as well as the port of the LDAP server should point to the correct (secure one, usually 636).

Please consult the screenshot below:



That should suffice for making the connection work securely (This is valid for Provisioning LDAP versions 9.0.4.13.0 and upwards).

4.7.1.1 Configuring LDAP with provided certificates

The LDAP provisioning tool offers the option to import you own certificates provided. These certificates should be copied under the certs folder contained in the LDAP Provisioning tool application folder. After copying the certificates (the extension should be.cer) you can execute the `importCustomCertificate.sh` script contained in the LDAP Provisioning tool root folder.

4.7.2 User Mapping Configuration Example

Overview

1) `<port>`

```
<?xml version="1.0" encoding="UTF-8"?>
<job xmlns="urn:siemens:dxm:configuration:1:0"
    xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
    xmlns:spml="urn:oasis:names:tc:SPML:1:0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <controller
        className="com.siemens.dxm.join.SyncOneWay"
        notifyOnAdd="false"
        notifyOnDelete="false">
        <logging filename="trace.txt" level="9"/>
    </controller>
    <port
        connector="LdapConnector"
        mode="out"
        name="IdentityDomain"
        role="connector">
    <port
        connector="SOAPProxy"
        mode="in"
        name="TS"
        role="connector">
    <connector
        role="responseWriter"
        name="SPML File writer"
        className="siemens.dxm.connector.framework.test.SpmlTestWriter">
        <connection type="SPML" filename="receivedRsp.xml"></connection>
    </connector>
</job>
```

The value `com.siemens.dxm.join.SyncOneWay` of the `className` attribute of the `<controller>` element indicates that data can only be

imported from the LDAP directory to the Symphonia database but it cannot be exported from the Symphonia database to the LDAP directory.

The first <port> element describes the connection of the LDAP Provisioning Tool to the LDAP directory named IdentityDomain. Since this is the data source, is attribute mode has the value out.

The second <port> element describes the connection of the LDAP Provisionig tool to the OpenScape UC Application named TS, target system. Since it is the target, the attribute mode has the value in.

Connection to the source system

Expand the first <port> in an editor.

1) <connector><connection>

```
<port
    connector="LdapConnector"
    mode="out"
    name="IdentityDomain"
    role="connector">
    <connector
        className="siemens.dxm.connector.ldap.LdapConnector"
        name="IdentityDomain"
        role="connector">
        <connection
            type="LDAP"
            user=""
            password=""
            server="139.25.228.165"
            port="389">
        </connection>
        <logging filename="trace.txt" level="9"/>
    </connector>
    <channel
```

<connection> in the first <port> provides connection information to the data source. The attributes user and password are the credentials of the LDAP directory user, e. g. domain\user in a Microsoft Active Directory. server is the IP address of the LDAP server.

2) <channel><environment>

```
    </connector>
    <channel
        exportSeqNo="1"
        id="ldap.out"
        name="ldapaccounts">
        <correspondingChannel>sym.in</correspondingChannel>
        <memberChannel/>
        <environment>
            <property
                name="sym_source_base"
                value="l=KPT,ou=SEN,o=SEN,c=DE"/>
        </environment>
        <export
```

<channel><environment><property> defines the LDAP query to be used for finding the source entries that need to be considered and included in the current job. The sym_source_base value of name attribute will be used in <spml:id>${env.sym_source_base}</spml:id> in the next step.

1)

1)

2) `<export>...<spml:attr>`

```
</environment>
<export
    pageSize="50"
    pagedRead="false"
    pagedTimeLimit="0"
    sortAttribute=""
    sortOrder="ASCENDING"
    xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
    xmlns:spml="urn:oasis:names:tc:SPML:1:0">
    <operationalAttributes>
        <spml:attr name="scope">
            <dsml:value>subtree</dsml:value>
        </spml:attr>
    </operationalAttributes>
    <searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
        <spml:id>${env.sym_source_base}</spml:id>
            <spml:identifierAttributes >
                <spml:attr name="objectclass">
                    <dsml:value type="string">User</dsml:value>
                </spml:attr>
                <!-- Symphonia Domain Name-->
                <spml:attr name="domain">
                    <dsml:value type="string">system</dsml:value>
                </spml:attr>
            </spml:identifierAttributes>
    </searchBase>
    <filter>
        <dsml:equalityMatch name="objectClass">
            <!-- Search only for "person" objects
            of IdentityDomain LDAP source -->
            <dsml:value>person</dsml:value>
        </dsml:equalityMatch>
    </filter>
</export>
<joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
```

The `<spml:attr>` with the value `objectclass` in the attribute `name` specifies the type of object handlers to be handled by the provisioning services. It receives values according to the available object, e. g. `User`, `Contact`, `Phone` etc. in its subelement `<dsml:attr>`. It has to be specified as an `identifierAttribute` of every request as well as the equalityMatch DSML attribute of the `<filter>` configuration.

3) `<filter>`

`<filter>` limits the types of entries being retrieved and processed from the data source. This limitation comes in form of an objectclass value. Please check your source system with an LDAP browser to see the available object classes.

4) `<joins><join><filterExtension>`

```
</export>
<joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
    <join>
        <filterExtension>
            <dsml:equalityMatch name="dn">
                <dsml:value>${source.id}</dsml:value>
            </dsml:equalityMatch>
        </filterExtension>
    </join>
</joins>
<import
```

`<filterExtension>` specifies the equalityMatch criteria for matching entries between the data source and the data target. The value is a simple expression (see simpleexpression).

5) <attributes>

```
        </joins>
        <import
            createDespiteOfMultipleMatch="true"
            idInAddRequest="true"
            notifyOnAdd="false"
            notifyOnDelete="false"/>
        <attributes>
            <attribute name="sn"/>
            <attribute name="givenName"/>
            <attribute name="dn"/>
            <attribute name="mail"/>
        </attributes>
        <mappingDefinition
```

<attributes> defines some attributes that are created by the DirX Identity Light Framework engine for being used throughout the process.

6) <mappingDefinition>

```
        </attributes>
        <mappingDefinition
            packagename=""
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <idMapping
                classname="Id"
                coderef="idMapping"
                mappingType="direct"
                type="urn:oasis:names:tc:SPML:1:0#DN">
                <value>dn</value>
            </idMapping>
            <attrMapping mappingType="direct" name="sn">
                <value>lastName</value>
            </attrMapping>
            <attrMapping mappingType="direct" name="givenname">
                <value>firstName</value>
            </attrMapping>
            <attrMapping mappingType="direct" name="mail">
                <value>mail</value>
            </attrMapping>
        </mappingDefinition>
        <userhook implementationLanguage="java"/>
    </channel>
</port>
<port
```

This is the first core part of the entire XML file. <mappingDefinition> identifies the type and potential value of the <attributes> in step 7.

As an example, the specified mail attribute (name="mail") will be created by the engine and the LDAP attribute “mail” (<value>mail</value>) will be directly mapped to its value for each entry to be provisioned. These attributes will then become available by the engine to the next section, the target system (see section Connection to the Target System on page 101).

There is an analog second core part in the second <port> element (see step 14 on page 108).

Connection to the Target System

The second <port> element is analog to the Connection to the source system on page 99 section but it contains the configuration of the LDAP Provisioning Tool for accessing and storing the data retrieved from the source system, LDAP directory, into the target system (TS), the OpenScape UC Application.

1) <connector><connection>

```
</port>
<port
    connector="SOAPProxy"
    mode="in"
    name="TS"
    role="connector">
    <connector
        className="siemens.dxm.connector.framework.soap.SpmlSoapProxy"
        mode="in"
        name="TS"
        role="connector">
        <connection
            type="SOAP"
            server="localhost"
            port="8080"
            ssl="false"
            user="administrator@system"
            password="01*Test!">
            <property
                name="path"
                value="symphonia-spmlresponder/services/SpmlSoapService"/>
        </connection>
    </connector>
    <channel
```

<connection> defines the connection options to the SPML responder, the interface of the OpenScape UC Application. Change the following values:

- server attribute: IP address or DNS resolvable name of the application computer
- user attribute: Adminstrator access to the OpenScape UC Application
- password attribute: The administrator's password

2) <channel><environment>

```
    </connector>
    <channel
        createDespiteOfMultipleMatch="true"
        exportSeqNo="1"
        id="sym.in"
        name="symaccounts">
        <correspondingChannel>ldap.out</correspondingChannel>
        <memberChannel/>
        <environment>
            <property name="sym_basenode" value="system"/>
            <!-- Path below the root path of the LDAP
            Provisioning Tool, e. g. "D:\ldapprov" -->
            <property
                name="sourcePath"
                value="testresources\siemens\dxm\map\ldap\accounts\to"/>
            <property name="objectclass" value="User"/>
            <property name="domain" value="system"/>
        </environment>
        <export
```

a) sym_basenode: The <property> with the name sym_basenode defines the root under which to look for OpenScape UC Application users. sym_basenode will be used in <spml:id> ${env.sym_basenode}</spml:id> in step 11 on page 107.

b) sourcePath: The <property> with the name sourcePath defines the path to the Java classes if using Java for mapping attributes that mostly are defined in <attributes> in the <port> with the name identityDomain (see step 7 on page 105).

3) `<export><searchBase>`

```
        </environment>
        <export
            pageSize="50"
            pagedRead="false"
            pagedTimeLimit="0"
            sortAttribute=""
            sortOrder="ASCENDING"
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
                <spml:id>${env.sym_basenode}</spml:id>
                <spml:identifierAttributes >
                    <spml:attr name="objectclass">
                        <dsml:value type="string">User</dsml:value>
                    </spml:attr>
                    <!-- Symphonia Domain Name-->
                    <spml:attr name="domain">
                        <dsml:value type="string">system</dsml:value>
                    </spml:attr>
                </spml:identifierAttributes>
            </searchBase>
        </export>
        <joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core" xmlns:spr
```

The `<export><searchBase>` defines how to forward data containing the ID and the identifier attributes of the request to the OpenScape UC Application.

4) `<joins>`

```
        </export>
        <joins
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <join>
                <searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
                    <spml:id>${source.givenName}+"."+${source.sn}</spml:id>
                    <spml:identifierAttributes>
                        <spml:attr name="objectclass">
                            <dsml:value type="string">User</dsml:value>
                        </spml:attr>
                        <!-- Symphonia Domain Name-->
                        <spml:attr name="domain">
                            <dsml:value type="string">system</dsml:value>
                        </spml:attr>
                    </spml:identifierAttributes>
                </searchBase>
            </join>
        </joins>
        <import
```

`<joins>` specifies the `<spml:id>` and the identifierAttributes of each joined entry, i. e. of each entry to be found in the target system matching the search criteria.

5) <attributes>

```
            <import
                createDespiteOfMultipleMatch="false"
                idInAddRequest="true"
                notifyOnAdd="false"
                notifyOnDelete="false"/>
            <attributes>
                <attribute name="userId"/>
                <attribute name="objectclass"/>
                <attribute name="domain"/>
                <attribute name="displayName"/>
                <attribute name="homeTimeZone"/>
                <attribute name="defaultLocale"/>
                <attribute name="vanityCodeString"/>
                <attribute name="password"/>
                <attribute name="pin"/>
            </attributes>
            <mappingDefinition
```

<attributes> and <mappingDefinition> in the next step are the most important elements for the creation of entries in the target system.

<attributes> defines the attributes of the data being forwarded to the target system.

6) <mappingDefinition>

```
            </attributes>
            <mappingDefinition
                packagename="siemens.dxm.map.ldap.accounts.to"
                xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
                xmlns:spml="urn:oasis:names:tc:SPML:1:0">
                <idMapping
                    classname="IdMapper"
                    mappingType="javasource"
                    type="urn:oasis:names:tc:SPML:1:0#DN">
                    <value>distinguishedName</value>
                </idMapping>
                <attrMapping name="objectclass" mappingType="constant">
                    <value>User</value>
                </attrMapping>
                <attrMapping name="domain" mappingType="constant">
                    <value>system</value>
                </attrMapping>
                <attrMapping mappingType="direct" name="displayName">
                    <value>sn</value>
                </attrMapping>
                <attrMapping name="homeTimeZone" mappingType="constant">
                    <value>Europe/Paris</value>
                </attrMapping>
                <attrMapping name="defaultLocale" mappingType="constant">
                    <value>en_US</value>
                </attrMapping>
                <attrMapping name="vanityCodeString" mappingType="constant">
                    <value>1</value>
                </attrMapping>
                <attrMapping name="password" mappingType="constant">
                    <value>!Test01?</value>
                </attrMapping>
                <attrMapping name="pin" mappingType="constant">
                    <value>12345678</value>
                </attrMapping>
            </mappingDefinition>
        </channel>
    </port>
    <connector
```

This is the second core part of the entire XML file (see step 8 on page 105 for the first core part). It defines the values for the attributes being defined in <attributes> in step 13. Some of them get constant values, some of them get values being determined in <attributes> of the IdentityDomain port (see step 7 on page 105) and some of them get values being determined by Java code.

a) IdMapper: <idMapping> has a <value> subelement with the value distinguishedName. This indicates that the default value of the

<userId> in step 13 is the distinguishedName attribute dn of entry in the source system (see step 7).

<idMapping> has the attribute mappingType with its value javasource. This indicates that the default value of <userId> is overwritten by Java code. The name of the Java class doing so is indicated by the value IdMapper of the attribute classname.

b) objectclass: The object class always is User since this configuration file is for user provisioning.

c) domain: The domain always is system.

d) displayName: The display name always is the value of the entry's attribute sn in the source system (see step 7).

e) homeTimeZone: This attribute indicates the time zone. It is only used for display purposes so that any value is accepted. Please assign the appropriate time zone for your situation. Example: Europe/Paris.

f) defaultLocale: This attribute indicates the used language. It is only used for display purposes so that any value is accepted. Example: en_US.

g) vanityCodeString: The vanity code always is 1.

h) password: Enter the default password for your OpenScape UC Application system.

i) pin: Enter the default PIN for your OpenScape UC Application system.

4.7.3 Contact Mapping Configuration Example

This section describes the changes that have to be done to the configuration file described in User Mapping Configuration Example on page 98 in order to provision contact data from the source system to the target system.

Overview

1) <port> remains unchanged.

```
<?xml version="1.0" encoding="UTF-8"?>
<job xmlns="urn:siemens:dxm:configuration:1:0"
    xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
    xmlns:spml="urn:oasis:names:tc:SPML:1:0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <controller
        className="com.siemens.dxm.join.SyncOneWay"
        notifyOnAdd="false"
        notifyOnDelete="false">
        <logging filename="trace.txt" level="9"/>
    </controller>
    <port
        connector="LdapConnector"
        mode="out"
        name="IdentityDomain"
        role="connector">
    <port
        connector="SOAPProxy"
        mode="in"
        name="TS"
        role="connector">
    <connector
        role="responseWriter"
        name="SPML File writer"
        className="siemens.dxm.connector.framework.test.SpmlTestWriter">
        <connection type="SPML" filename="receivedRsp.xml"></connection>
    </connector>
</job>
```

Connection to the source system

1) <connector><connection> remains unchanged.

```
<port
    connector="LdapConnector"
    mode="out"
    name="IdentityDomain"
    role="connector">
    <connector
        className="siemens.dxm.connector.ldap.LdapConnector"
        name="IdentityDomain"
        role="connector">
        <connection
            type="LDAP"
            user=""
            password=""
            server="139.25.228.165"
            port="389">
        </connection>
        <logging filename="trace.txt" level="9"/>
    </connector>
    <channel
```

2) <channel><environment> remains unchanged.

```
    </connector>
    <channel
        exportSeqNo="1"
        id="ldap.out"
        name="ldapaccounts">
        <correspondingChannel>sym.in</correspondingChannel>
        <memberChannel/>
        <environment>
            <property
                name="sym_source_base"
                value="l=KPT,ou=SEN,o=SEN,c=DE"/>
        </environment>
        <export
```

3) <export>...<spml:attr>

```
    </environment>
    <export
        pageSize="50"
        pagedRead="false"
        pagedTimeLimit="0"
        sortAttribute=""
        sortOrder="ASCENDING"
        xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
        xmlns:spml="urn:oasis:names:tc:SPML:1:0">
        <operationalAttributes>
            <spml:attr name="scope">
                <dsml:value>subtree</dsml:value>
            </spml:attr>
        </operationalAttributes>
        <searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
            <spml:id>${env.sym_source_base}</spml:id>
                <spml:identifierAttributes >
                    <spml:attr name="objectclass">
                    <dsml:value type="string">Contact</dsml:value>
                    </spml:attr>
                    <!-- Symphonia Domain Name-->
                    <spml:attr name="domain">
                        <dsml:value type="string">system</dsml:value>
                    </spml:attr>
                </spml:identifierAttributes>
        </searchBase>
        <filter>
            <dsml:equalityMatch name="objectClass">
                <dsml:value>scdPerson</dsml:value>
            </dsml:equalityMatch>
        </filter>
    </export>
    <joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
```

Since now contacts are provisioned, change the value of <dsml:value> from User to Contact. This does not affect the source system but the search for existing entries in the target system.

4) <filter>

Change the value of <dsml:value> from person to scdPerson.

5) <joins>...<filterExtension>

```
</export>
<joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
    <join>
        <filterExtension>
            <dsml:equalityMatch name="dn">
                <dsml:value>${target.externalId}</dsml:value>
            </dsml:equalityMatch>
        </filterExtension>
    </join>
</joins>
<import
```

Change the value of <dsml:value> from ${source.id} to ${target.externalId}.

6) <attributes>

```
<import
    createDespiteOfMultipleMatch="true"
    idInAddRequest="true"
    notifyOnAdd="false"
    notifyOnDelete="false"/>
<attributes>
    <attribute name="sn"/>
    <attribute name="givenName"/>
    <attribute name="mail"/>
    <attribute name="displayName"/>
</attributes>
<mappingDefinition
```

a) Remove <attribute name="dn"/>.

b) Add the following elements:

```
<attribute name="displayName"/>

<attribute name="department"/>
```

7) <mappingDefinition>

```
</attributes>
<mappingDefinition
    packagename=""
    xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
    xmlns:spml="urn:oasis:names:tc:SPML:1:0">
    <idMapping
        classname="Id"
        coderef="idMapping"
        mappingType="direct"
        type="urn:oasis:names:tc:SPML:1:0#DN">
        <value>sn</value>
    </idMapping>
    <attrMapping mappingType="direct" name="sn">
        <value>lastName</value>
    </attrMapping>
    <attrMapping mappingType="direct" name="givenname">
        <value>firstName</value>
    </attrMapping>
    <attrMapping mappingType="direct" name="displayname">
        <value>displayName</value>
    </attrMapping>
    <attrMapping mappingType="direct" name="mail">
        <value>mail</value>
    </attrMapping>
</mappingDefinition>
<userhook implementationLanguage="java"/>
</channel>
</port>
<port
```

a) In <idMapping>, replace <value>dn</value> by <value>sn</value>.

b) Add the following elements:

```
<attrMapping mappingType="direct" name="displayname">
<value>displayName</value>
</attrMapping>
```

```
<attrMapping mappingType="direct" name="department">
<value>department</value>
</attrMapping>
```

Connection to Target System

1) `<connector><connection>` remains unchanged.

```
</port>
<port
    connector="SOAPProxy"
    mode="in"
    name="TS"
    role="connector">
    <connector
        className="siemens.dxm.connector.framework.soap.SpmlSoapProxy"
        mode="in"
        name="TS"
        role="connector">
        <connection
            type="SOAP"
            server="localhost"
            port="8080"
            ssl="false"
            user="administrator@system"
            password="01*Test!">
            <property
                name="path"
                value="symphonia-spmlresponder/services/SpmlSoapService"/>
        </connection>
    </connector>
    <channel
```

2) `<channel><environment>`

```
    </connector>
    <channel
        createDespiteOfMultipleMatch="true"
        exportSeqNo="1"
        id="sym.in"
        name="symaccounts">
        <correspondingChannel>ldap.out</correspondingChannel>
        <memberChannel/>
        <environment>
            <property name="sym_basenode" value="system"/>
            <!-- Path below the root path of the LDAP
            Provisioning Tool, e. g. "D:\ldapprov" -->
            <property
                name="sourcePath"
                value="testresources\siemens\dxm\map\ldap\accounts\to"/>
            <property
                name="objectclass"
                value="Contact"/>
            <property
                name="domain"
                value="system"/>
        </environment>
```

Replace `<property name="objectclass" value="User"/>` by `<property name="objectclass" value="Contact"/>`.

3) `<export><searchBase>`

```
        </environment>
        <export
            pageSize="50"
            pagedRead="false"
            pagedTimeLimit="0"
            sortAttribute=""
            sortOrder="ASCENDING"
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
                <spml:id>${env.sym_basenode}</spml:id>
                    <spml:identifierAttributes >
                        <spml:attr name="objectclass">
                            <dsml:value type="string">Contact</dsml:value>
                        </spml:attr>
                        <!-- Symphonia Domain Name-->
                        <spml:attr name="domain">
                            <dsml:value type="string">system</dsml:value>
                        </spml:attr>
                    </spml:identifierAttributes>
            </searchBase>
        </export>
        <joins
```

Change the value of `<dsml:value>` from `User` to `Contact`.

4) <joins>

```
</export>
<joins
    xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
    xmlns:spml="urn:oasis:names:tc:SPML:1:0">
    <join>
        <searchBase
            mappingType="simpleexpression"
            type="urn:oasis:names:tc:SPML:1:0#DN" >
            <spml:id>${source.givenName}+"."+${source.sn}</spml:id>
            <spml:identifierAttributes >
                <spml:attr name="objectclass">
                    <dsml:value type="string">Contact</dsml:value>
                </spml:attr>
                <!-- Symphonia Domain Name-->
                <spml:attr name="domain">
                    <dsml:value type="string">system</dsml:value>
                </spml:attr>
            </spml:identifierAttributes>
        </searchBase>
    </join>
</joins>
<import
```

Change the value of <dsml:value> from User to Contact.

5) <attributes>

```
<import
    createDespiteOfMultipleMatch="false"
    idInAddRequest="true"
    notifyOnAdd="false"
    notifyOnDelete="false"/>
<attributes>
    <attribute name="externalId" />
    <attribute name="objectclass" />
    <attribute name="domain" />
    <attribute name="displayName"/>
    <attribute name="givenName"/>
    <attribute name="lastName"/>
    <attribute name="assignedEmail"/>
    <attribute name="assignedSymUserIdentity"/>
</attributes>
<mappingDefinition
```

<attribute> and <attrMapping> (see step 14 on page 116) in <port> with the name TS specify the attributes being needed in the SPML requests.

a) Exchange <attribute name="userId"/> by <attribute name="externalId"/>.

b) Remove the <attribute> elements having the following values of the name attributes:

- homeTimeZone
- defaultLocale
- vanityCodeString
- password
- pin

c) Add <attribute> elements having the following values of the name attributes:

- displayName; If displayName is not provided, displayName is composed of givenName and lastName
- givenName
- lastName
- assignedEmail
- assignedSymUserIdentity

6) <mappingDefinition>

```
        </attributes>
        <mappingDefinition
            packagename="siemens.dxm.map.ldap.accounts.to"
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <idMapping
                classname="IdMapper"
                mappingType="javasource"
                type="urn:oasis:names:tc:SPML:1:0#DN">
                <value>distinguishedName</value>
            </idMapping>
            <attrMapping name="objectclass" mappingType="constant">
                <value>Contact</value>
            </attrMapping>
            <attrMapping name="domain" mappingType="constant">
                <value>system</value>
            </attrMapping>
            <attrMapping name="displayName" mappingType="direct">
                <value>displayName</value>
            </attrMapping>
            <attrMapping name="givenName" mappingType="direct">
                <value>givenName</value>
            </attrMapping>
            <attrMapping name="lastName" mappingType="direct">
                <value>sn</value>
            </attrMapping>
            <attrMapping name="assignedEmail" mappingType="direct">
                <value>mail</value>
            </attrMapping>
            <attrMapping
                name="assignedSymUserIdentity"
                mappingType="simpleexpression">
                <value>${source.givenName} + "." + ${source.sn}
                + "@" + ${env.domain}</value>
            </attrMapping>
        </mappingDefinition>
    </channel>
</port>
<connector
```

a) In the <attrMapping> having objectclass as value of the name attribute, replace the value User of the <value> subelement by Contact.

b) In the <attrMapping> having displayName as value of the name attribute, replace the sn value of the <value> subelement by displayName.

c) Remove the <attrMapping> elements having the following values of the name attributes:

- homeTimeZone
- defaultLocale
- vanityCodeString
- password
- pin

d) Add the following elements:

```
<attrMapping mappingType="direct" name="givenName">
<value>givenName</value>
</attrMapping>
<attrMapping mappingType="direct" name="lastName">
<value>sn</value>
</attrMapping>
<attrMapping mappingType="direct"
 name="assignedEmail">
<value>mail</value>
</attrMapping><attrMapping mappingType="direct"
 name="givenName">
<value>givenName</value>
</attrMapping>
```

```
<attrMapping mappingType="direct" name="lastName">
<value>sn</value>
</attrMapping>
<attrMapping mappingType="direct"
 name="assignedEmail">
<value>mail</value>
</attrMapping>
```

4.7.4 Adding a new <attribute> to Contacts

If an new <attribute>, e. g. department, is to be added, both <port> XML elements have to be extended, see subsection Connection to the Source System and subsection Connection to Target System.

Connection to the Source System

1) Look for <attributes> in the <port> XML element having the name attribute with the value IdentityDomain.
2) Add the following element:
   <attribute name="department"/>

3) Look for <mappingDefinition> in the same <port>.

4) Add the following element:

```
<attrMapping mappingType="direct" name="department">
<value>department</value>
</attrMapping>
```

```
        </attributes>
        <mappingDefinition
            packagename=""
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <idMapping
                classname="Id"
                coderef="idMapping"
                mappingType="direct"
                type="urn:oasis:names:tc:SPML:1:0#DN">
                <value>sn</value>
            </idMapping>
            <attrMapping mappingType="direct" name="sn">
                <value>lastName</value>
            </attrMapping>
            <attrMapping mappingType="direct" name="givenname">
                <value>firstName</value>
            </attrMapping>
            <attrMapping mappingType="direct" name="displayname">
                <value>displayName</value>
            </attrMapping>
            <attrMapping mappingType="direct" name="mail">
                <value>mail</value>
            </attrMapping>
            <attrMapping  mappingType="direct" name="department">
                <value>department</value>
            </attrMapping>
        </mappingDefinition>
        <userhook implementationLanguage="java"/>
    </channel>
</port>
<port
```

Connection to Target System

1) Look for <attributes> in the <port> element having the name attribute with the value TS (target system).

2) Add the following element:

```
<attribute name="department"/>
```

```
        <import
            createDespiteOfMultipleMatch="false"
            idInAddRequest="true"
            notifyOnAdd="false"
            notifyOnDelete="false"/>
        <attributes>
            <attribute name="externalId" />
            <attribute name="objectclass" />
            <attribute name="domain" />
            <attribute name="displayName"/>
            <attribute name="givenName"/>
            <attribute name="lastName"/>
            <attribute name="assignedEmail"/>
            <attribute name="assignedSymUserIdentity"/>
            <attribute name="department"/>
        </attributes>
        <mappingDefinition
```

3) Look for <mappingDefinition> in the same <port>.

4) Add the following element:

```
<attrMapping mappingType="direct" name="department">
<value>department</value>
</attrMapping>
```

```
            </attributes>
            <mappingDefinition
                packagename="siemens.dxm.map.ldap.accounts.to"
                xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
                xmlns:spml="urn:oasis:names:tc:SPML:1:0">
                <idMapping
                    classname="IdMapper"
                    mappingType="javasource"
                    type="urn:oasis:names:tc:SPML:1:0#DN">
                    <value>distinguishedName</value>
                </idMapping>
                <attrMapping name="objectclass" mappingType="constant">
                    <value>Contact</value>
                </attrMapping>
                <attrMapping name="domain" mappingType="constant">
                    <value>system</value>
                </attrMapping>
                <attrMapping name="displayName" mappingType="direct">
                    <value>displayName</value>
                </attrMapping>
                <attrMapping name="givenName" mappingType="direct">
                    <value>givenName</value>
                </attrMapping>
                <attrMapping name="lastName" mappingType="direct">
                    <value>sn</value>
                </attrMapping>
                <attrMapping name="assignedEmail" mappingType="direct">
                    <value>mail</value>
                </attrMapping>
                <attrMapping name="department" mappingType="direct">
                    <value>department</value>
                </attrMapping>
                <attrMapping
                    name="assignedSymUserIdentity"
                    mappingType="simpleexpression">
                    <value>${source.givenName} + "." + ${source.sn}
                    + "@" + ${env.domain}</value>
                </attrMapping>
            </mappingDefinition>
        </channel>
    </port>
    <connector
```

4.7.5 Using simple Expressions for Contacts

If you want to send an instant messaging address of a contact, e. g. `peter.petersson@company.com` via SPML to the OpenScape UC application, this address has to be assigned to the `imAddress` field of a contact.

The simpleexpression language of DirX Identity Light Framework is used to construct this address from the data being available in the source system (see step 6 on page 121).

Connection to Target System

1) Look for `<environment>` in the `<port>` XML element having the `name` attribute with the value `TS` (target system).

2) Add an XML element with your IM server address according to the following example:

```
<property name="imServer" value="company.com"/>
```

```
        </connector>
        <channel
            createDespiteOfMultipleMatch="true"
            exportSeqNo="1"
            id="sym.in"
            name="symaccounts">
            <correspondingChannel>ldap.out</corre
            <memberChannel/>
            <environment>
                <property name="sym_basenode" va
                <!-- Path below the root path of
                Provisioning Tool, e. g. "D:\ldap
                <property
                    name="sourcePath"
                    value="testresources\siemens\
                <property
                    name="objectclass"
                    value="Contact"/>
                <property
                    name="domain"
                    value="system"/>
                <property
                    name="imServer"
                    value="company.com" />
            </environment>
            <export
```

3) Look for <attributes> in the same <port> XML element.

4) Add the following element:

```
<attribute name="imAddress"/>
```

```
            <import
                createDespiteOfMultipleMatch="false"
                idInAddRequest="true"
                notifyOnAdd="false"
                notifyOnDelete="false"/>
            <attributes>
                <attribute name="externalId" />
                <attribute name="objectclass" />
                <attribute name="domain" />
                <attribute name="displayName"/>
                <attribute name="givenName"/>
                <attribute name="lastName"/>
                <attribute name="assignedEmail"/>
                <attribute name="assignedSymUserIdentity"/>
                <attribute name="imAddress"/>
                <attribute name="department"/>
            </attributes>
            <mappingDefinition
```

5) Look for <mappingDefinition> in the same <port>.

6) Add the following element:

```
<attrMapping name="imAddress"
 mappingType="simpleexpression">
```

```
<value>${source.givenName} + "." + ${source.sn} + "@" +
 ${env.imServer}</value>
</attrMapping>
```

```
        </attributes>
        <mappingDefinition
            packagename="siemens.dxm.map.ldap.accounts.to"
            xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
            xmlns:spml="urn:oasis:names:tc:SPML:1:0">
            <idMapping
                classname="IdMapper"
                mappingType="javasource"
                type="urn:oasis:names:tc:SPML:1:0#DN">
                <value>distinguishedName</value>
            </idMapping>
            <attrMapping name="objectclass" mappingType="constant">
                <value>Contact</value>
            </attrMapping>
            <attrMapping name="domain" mappingType="constant">
                <value>system</value>
            </attrMapping>
            <attrMapping name="displayName" mappingType="direct">
                <value>displayName</value>
            </attrMapping>
            <attrMapping name="givenName" mappingType="direct">
                <value>givenName</value>
            </attrMapping>
            <attrMapping name="lastName" mappingType="direct">
                <value>sn</value>
            </attrMapping>
            <attrMapping name="assignedEmail" mappingType="direct">
                <value>mail</value>
            </attrMapping>
            <attrMapping name="department" mappingType="direct">
                <value>department</value>
            </attrMapping>
            <attrMapping
                name="assignedSymUserIdentity"
                mappingType="simpleexpression">
                <value>${source.givenName} + "." + ${source.sn}
                + "@" + ${env.domain}</value>
            </attrMapping>
        </mappingDefinition>
    </channel>
</port>
<connector
```

The instant messaging address consists of several parts, the first name of the user, a dot, the family name of the user, the '@' character and instant messaging server name. The construction of the entire address is done by using the simpleexpression language of DirX Identity Light Framework.

The `${source.givenName}` expression gets the first name of the user givenName from `<attribute name="givenName">` being defined in the `<port>` connecting to the source system (see step 7 on page 105).

The `${source.sn}` expression is retrieved in the same way.

The `${env.imServer}` expression retrieves the name of the instant messaging server being defined in step 2 on page 120.

4.8 Configuration File Reference

The configuration for the mapping component is based on an XML schema and is part of a DirX Identity framework's `<job>` configuration. The following figure gives an example of the most important elements in the configuration file user mapping.

```
<?xml version="1.0" encoding="UTF-8"?>
<job xmlns="urn:siemens:dxm:configuration:1:0"
    xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
    xmlns:spml="urn:oasis:names:tc:SPML:1:0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <controller
        className="com.siemens.dxm.join.SyncOneWay"
        notifyOnAdd="false"
        notifyOnDelete="false">
        <logging filename="trace.txt" level="9"/>
    </controller>
    <port
        connector="LdapConnector"
        mode="out"
        name="IdentityDomain"
        role="connector">
        <connector
            className="siemens.dxm.connector.ldap.LdapConnector"
            name="IdentityDomain"
            role="connector">
        <channel
            exportSeqNo="1"
            id="ldap.out"
            name="ldapaccounts">
            <correspondingChannel>sym.in</correspondingChannel>
            <memberChannel/>
            <environment>
            <export>
                <operationalAttributes>
                    <spml:attr name="scope">
                </operationalAttributes>
                <searchBase type="urn:oasis:names:tc:SPML:1:0#DN">
                <filter>
                    <dsml:equalityMatch name="objectClass">
                </filter>
            </export>
            <joins xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
                <join>
                    <filterExtension>
                        <dsml:equalityMatch name="dn">
                    </filterExtension>
                </join>
            </joins>
            <import
                createDespiteOfMultipleMatch="true"
                idInAddRequest="true"
                notifyOnAdd="false"
                notifyOnDelete="false"/>
            <attributes>
            <mappingDefinition>
                <idMapping
                    classname="Id"
                    coderef="idMapping"
                    mappingType="direct"
                    type="urn:oasis:names:tc:SPML:1:0#DN">
                <attrMapping mappingType="direct" name="sn">
                <attrMapping mappingType="direct" name="givenname">
                <attrMapping mappingType="direct" name="mail">
            </mappingDefinition>
            <userhook implementationLanguage="java"/>
        </channel>
    </port>
    <port
        connector="SOAPProxy"
        mode="in"
        name="TS"
        role="connector">
    <connector
        role="responsewriter"
        name="SPML File writer"
        className="siemens.dxm.connector.framework.test.SpmlTestWriter">
</job>
```

4.8.1 <job>

A <job> describes the synchronization between various systems, typically a source and a target system.

<job> has no attributes beside name space definitions.

Table 23: Attributes of <controller>

Attribute	Description	Value
`xmlns`	Namespace	`urn:siemens:dxm:configuration:1:0`
`xmlns:dsml`	DSML namespace	`urn:oasis:names:tc:DSML:2:0:core`
`xmlns:spml`	SPML namespace	`urn:oasis:names:tc:SPML:1:0`
`xmlns:xsi`	XSI namespace	`http://www.w3.org/2001/XMLSchema-instance`

Allowed subelements of <job>

- <controller>
- <port>
- <connector>

4.8.2 <controller>

<controller> defines basic settings needed for provisioning data from the source system to the target system.

Table 24: Attributes of <controller>

Attribute	Description	Example Value
`className`		`com.siemens.dxm.join.SyncOneWay`
`notifyOnAdd`		`false` and `true`
`notifyOnDelete`		`false` and `true`

The value `com.siemens.dxm.join.SyncOneWay` of the `className` attribute of the <controller> element indicates that data can only be imported from source system to the target system but it cannot be exported from the target system back to the source system. The LDAP directory is not yet defined as the source system and the Symphonia database in the OpenScape UC Application is not yet defined as the target system. This will be done later.

Allowed subelements of <controller>

None

4.8.3 <port>

The LDAP Provisioning Tool is in the middle between the source system and the target system. A <port> element gives information about a system the LDAP Provisioning Tool is connected to, i. e. you need a <port> element describing the source system (LDAP directory) and a <port> element describing the target system (OpenScape UC Application).

Table 25: Attributes of <port>

Attribute	Description	Example Value
`connector`		`LdapConnector`
`mode`	`mode` indicates whether this port describes a source system or a target system.	`out` and `in` The value `out` indicates a source system and the value `in` indicates a target system. Since data only can be provisioned from the LDAP directory (source system) to the OpenScape UC Application (target system), the port describing the LDAP directory must have a `mode` attribute having the value `out` and the port describing the OpenScape UC Application must have a `mode` attribute having the value `in`.
`name`	`name` is the port's name being used a reference in subelements of `<port>`.	`IdentityDomain` and `TS`
`role`		`connector`

Example

The following code shows a sample usage of a `<port>`.

```
<port
connector="LdapConnector"
mode="out"
name="IdentityDomain"
role="connector">…
<connector>
…
</connector>
<channel>
…
</channel>
</port>
<port
connector="SOAPProxy"
mode="in"
name="TS"
role="connector">
<connector>
…
</connector>
<channel>
…
</channel>
</port>
```

Allowed subelements of `<port>`

- <connector>

- <channel>

4.8.4 <connector>

The connector identifies the method of connection between different systems. It provides an interface for DirX.

Table 26: Attributes of <connector>

Attribute	Description	Example Value
classname		siemens.dxm.connector.ldap.LdapConnector in the <connector> in the <port> of the source system and siemens.dxm.connector.framework.soap.SpmlSoapProxy in the <connector> in the <port> of the target system.
mode	mode indicates whether this connector describes a source system or a target system.	out and in The value out indicates a source system and the value in indicates a tar system. Since data only can be provisioned from the LDAP directory (sou system) to the OpenScape UC Application (target system), the connector describing the LDAP directory must have a mode attribute having the valu out and the connector describing the OpenScape UC Application must ha a mode attribute having the value in.
name	name is the connector's name	IdentityDomain and TS
role		connector

Allowed subelements of <connector>

- <connection>

4.8.5 <connection>

<connection> provides the information about the endpoints of the communication (computer name or IP address, port, SSL) and the user account try to establish this communication.

Table 27: Attributes of <connection>

Attribute	Description	Example Value
type		LDAP and SOAP
user	The user name to be used for logging into the source system or the target system	<domain name>\<user name> for a Microsoft Active Directory as the source system and administrator@system for the target system
password	The password the user uses for logging in	01*TEST!

Attribute	Description	Example Value
server	The name or the IP address of the computer where the source system or target system resides. Note: If using a computer name, be sure to use a resolvable name.	139.25.228.165 or localhost
port	The destination port numberof the communication from the LDAP Provisioning Tool to the computer either the source system or the target sysetm resides on.	389 in the <port> of the source system since this is the standard LDAP port number and 8080 in the <port> of the target system.
ssl	ssl indicates whether secure socket layer is used.	SSL is available for both LDAP & SOAP (SPML) connections. By enabling SSL in LDAP connections the appropriate secure port should be used as well

Allowed subelements of <connection>

- <property>

4.8.6 <property>

<property> states the path to be used in the HTTP address or HTTPS address to access the OpenScape UC Application.

The usage of <property> only is allowed in the <port>of the target system.

Table 28: Attributes of <property>

Attribute	Description	Example Value
name	The name of the property	path
value	The value of the property	symphonia-spmlresponder/services/SpmlSoapService

Allowed subelements of <property>

None

4.8.7 <channel>

Each system, e. g. a source system or a target system, can contain a number of object types, e. g. users, groups and roles. They are configured in <channel> elements beneath a <port>. A <channel> describes the properties of the object type, how to export the objects from the target system and map an entry from the corresponding source channel.

Most child elements of the <channel> are ignored in the property mapper. They are only relevant for the join engine. <mappingDefinition>, <attributes> and <environment> are important for mapping.

Table 29: Attributes of <channel>

Attribute	Description	Example Value
name	Name of the channel, such as “accounts” or “groups”. It is unique among all channels of the same port.	ldapaccounts or symaccounts
id	An identifier for this channel, which is unique among all channels of all ports.	ldap.out for the <port> of the source system and sym.in for the <port> of the target system
exportSeqNo		1
createDespiteOfMultipleMatch		true or false

Allowed subelements of <channel>

- <correspondingChannel>
- <memberChannel>
- <environment>
- <export>
- <joins> and <join>
- <import>
- <attributes>
- <mappingDefinition>

4.8.8 <correspondingChannel>

The <correspondingChannel> element contains the value of the id attribute of the corresponding channel, i. e. in a <channel> to the source it describes the channel to the target and in a <channel> to the target it describes the channel to the source. In mapping, only the attribute names are evaluated.

Example

```
<port
connector="LdapConnector"
mode="out"
name="IdentityDomain"
role="connector">
…
<channel
id="ldap.out"
…
<correspondingChannel>sym.in</correspondingChannel>
…
```

```
</channel>
</port>
<port
connector="SOAPProxy"
mode="in"
name="TS"
role="connector">
…
<channel
id="sym.in"
…
<correspondingChannel>ldap.out</correspondingChannel>
…
</channel>
</port>
```

Allowed subelements of `<correspondingChannel>`

None

4.8.9 `<memberChannel>`

The member channel configures the mapping for relationships between entries. Typically these are the memberships between a user and groups.

The various systems to be synchronized store memberships very differently: some at the user, some at the group or role, others like databases in separate tables. When synchronizing two such systems, all combinations are possible. The source system might store the memberships at the user, while the target stores it at the group or vice versa.

To allow for more flexibility in mapping configuration, the attribute(s) holding the entry relationships are configured in a separate channel, the member channel. The `<memberChannel>` subelement tells the attribute mapper, where the memberships are stored.

Example: If the group entries hold the identifiers of the users in their `member` attribute, the group channel needs the `<memberChannel>` subelement and the `member` attribute has to be defined in the appropriate channel. In this case, the join engine considers the attributes of the member channel, when constructing the group entry.

Allowed subelements of `<memberChannel>`

None

4.8.10 `<environment>`

The `<environment>` element contains a list of properties that represent the channel environment. They can be used anywhere in the mapping, especially for simple expression mapping and Java mapping functions. Typically, they contain properties that might be used at several locations. Samples are base nodes for users and groups in the source system and the target system.

Each property is defined with a `<property>` element, which consists of a `name` and a `value` attribute.

The following example shows the definition of sym_source_base in a <property> and its usage in a <searchBase>. The name attribute of <property> contains the property name. The value of the <value> contains the corresponding value. This value is a distinguished name (DN) identifying the search base. The LDAP API references an LDAP object by its distinguished name. A distinguished name is a sequence of relative distinguished names (RDN) connected by commas, e. g. l=KPT,ou=UNIFY,o=UNIFY,c=DE meaning that KPT is the locality name, UNIFY is the organization unit name, UNIFY ist the organization name and DE is the country name. Be sure to use $, { and } according to the example when using the items you defined in a <property>.

LDAP provisioning is designed to work with the email mailbox name as the name for the login name. If none is found or it is not declared that this value exists in the appropriate config.xml file, it will fallback to the distinguished name of the active directory record. Therefore, a user must have a mail value at his record.

Example

```
<channel
…
<environment>
<property
name="sym_source_base"
value="l=KPT,ou=UNIFY,o=UNIFY,c=DE"/>
</environment>
<export
…
<searchBase
type="urn:oasis:names:tc:SPML:1:0#DN">
<spml:id>${env.sym_source_base}</spml:id>
…
</searchBase>
```

Allowed subelements of <environment>

- <property>

4.8.11 <export>

<export> is used to define how the information will be transferred to an external system.

Table 30: Attributes of <export>

Attribute	Description	Example Value
pageSize		50
pagedRead		false
pagedTimeLimit		0
sortAttribute		
sortOrder	Sort order	ASCENDING or DESCENDING

Allowed subelements of <export>

- <operationalAttributes>
- <searchBase>
- <filter>

4.8.12 <operationalAttributes>

<operationsAttributes> provides the receiver of an SPML request with additional information that is needed for the correct processing of the request, e. g. if the SPML request contains a deletion of an email address, <operationsAttributes> may specify an archival policy for the deleted mailbox and a time at which the request should be executed.

Allowed subelements of <operationalAttributes>

- <spml:attr>

4.8.13 <spml:attr>

<spml:attr> has to be used in some SPML tags in order to describe a property and its value. The name of the property is given by the name attribute and the value of the property is given by the name of a <dsml:value> subelement.

Table 31: Attributes of <spml:attr>

Attribute	Description	Example Value
name	The name of an attr. The value of name is used a variable name. The name of a <dsml:value> subelement is used as the value of this variable.	scope, objectClass or domain objectClass is used to specify which type of object handlers will be handled by the provisioning services. It will receive values according to the available objects (e. g. User, Contact, Phone).

Example

```
<spml:attr
  name="objectclass">
  <dsml:value type="string">User</dsml:value>
</spml:attr>
<!-- Symphonia Domain Name-->
<spml:attr name="domain">
  <dsml:value type="string">system</dsml:value>
</spml:attr>
```

Allowed subelements of <spml:attr>

- <dsml:value>

4.8.14 <dsml:value>

The name of the <dsml:value> is used as a value of a variable. The value of the name attribute of an <spml:attr> is used as the name of this variable.

Table 32: Attributes of <dsml:value>

Attribute	Description	Example Value
type	The type of the element	string

Allowed subelements of <dsml:value>

None

4.8.15 <searchBase>

<searchBase> is used to specify the property that is used as primary search criterion for the search of objects. A <spml:id> subelement is used to specify the start point for the search operation and <spml:identifierAttributes> is used to specify in more detail what type of objects are to be searched.

Table 33: Attributes of <searchBase>

Attribute	Description	Example Value
type	This attribute contains a Uniform Resource Name (URN) indicating the property type that should be searched, e. g. urn:oasis:names:tc:SPML:1:0#DN means that the distinguished name is the primary search criterion and urn:oasis:names:tc:SPML:1:0#EmailAddress means that an email address is the primary search criterion.	urn:oasis:names:tc:SPML:1:0#D urn:oasis:names:tc:SPML:1:0#EmailAd urn:oasis:names:tc:SPML:1:0#OID urn:oasis:names:tc:SPML:1:0#URN

Example

```
<searchBase
 type="urn:oasis:names:tc:SPML:1:0#DN">
 <spml:id>${env.sym_source_base}</spml:id>
 <spml:identifierAttributes >
  <spml:attr
  name="objectclass">
  <dsml:value type="string">User</dsml:value>
  </spml:attr>
  <!-- Symphonia Domain Name-->
  <spml:attr name="domain">
  <dsml:value type="string">system</dsml:value>
  </spml:attr>
 </spml:identifierAttributes>
</searchBase>
```

Allowed subelements of <searchBase>

- <spml:id>
- <spml:identifierAttributes>

4.8.16 <spml:id>

The name of <spml:id> is used here to identify the start point of a search with <searchBase> in a data tree structure. No attributes are used.

Example

<spml:id>${env.sym_source_base}</spml:id>

Allowed subelements of <spml:id>

None

4.8.17 <spml:identifierAttributes>

<spml:identifier> is used to contain a collection of <spml:attr> subelements. These subelements speficy in more detail which objects are to be searched. No attributes are used.

Allowed subelements of <spml:identifierAttributes>

None

4.8.18 <filter>

This element only is used in the source system <port>.

<filter> uses a <dsml:equalityMatch> to limit the types of entries being retrieved and processed from the data source.

Example

```
<filter>
  <dsml:equalityMatch name="objectClass">
    <!-- Search only for "person" objects of IdentityDomain
 LDAP source -->
    <dsml:value>person</dsml:value>
  </dsml:equalityMatch>
</filter>
```

The name attribute indicates to search for entries of the objectClass type. They must have the value person indicated by the <dsml:value> subelement in order to pass the filter. Please check your LDAP directory with an LDAP browser to see the available object classes.

Allowed subelements of <filter>

- <dsml:equalityMatch>

  This element only contains the name attribute having the value objectclass and a <dsml:value> subelement indicating the type of entries to retrieve.

4.8.19 <dsml:equalityMatch>

<dsml:equalityMatch> limits the types of entries being retrieved and processed from the data source. The name attribute states the name of entry type that pass the filter. The value that this entry type must fullfil in order to pass the filter is determined by the <dsml:value> subelement.

Example

```
<dsml:equalityMatch name="objectClass">
  <!-- Search only for "person" objects of IdentityDomain
 LDAP source -->
  <dsml:value>person</dsml:value>
</dsml:equalityMatch>
```

The name attribute indicates to search for entries of the objectClass type. They must have the value person indicated by the <dsml:value> subelement in order to pass the filter. Please check your LDAP directory with an LDAP browser to see the available object classes.

Table 34: Attributes of <dsml:equalityMatch>

Attribute	Description	Example Value
name	The name attribute states the name of entry type that may pass the filter. The value that this entry type must fullfil in order to pass the filter is determined by the <dsml:value> subelement.	objectClass or dn If <dsml:equalityMatch> is used in a <filter>, its name attribute must have the value objectClass. If <dsml:equalityMatch> is used in a <filterExtension>, its name attribute must indicate the search criterion of the filter, e. g. dn (distinguished name) if persons are searched.

Allowed subelements of <dsml:equalityMatch>

- <dsml:value>

4.8.20 <joins> and <join>

<joins> contains a <join> containing a <filterExtension> specifying the <equalityMatch> criteria for matching entries between the source system and the target system.

4.8.21 <filterExtension>

<filterExtension> specifies the filter criteria in more detail. <filter> specifies e. g. to filter persons and <filterExtension> specifies that only persons whose distinguished name matches a certain value are chosen for being provisioned to the target system.

This more detailed filtering is done by using <dsml:equalityMatch> (see <dsml:equalityMatch>) as a subelement in <filter>. The value of the used <dsml:value> is a simple expression (see simpleexpression).

<filterExtension> is used without any attribute.

Example

```
<filterExtension>
  <dsml:equalityMatch name="dn">
  <dsml:value>${source.id}</dsml:value>
  </dsml:equalityMatch>
</filterExtension>
```

Allowed subelements of <filterExtension>

- <dsml:equalityMatch>

4.8.22 <import>

<import> specifies the import configuration.

Table 35: Attributes of <import>

Attribute	Description	Example Value
createDespiteOfMultipleMatch		true
idInAddRequest		true
notifyOnAdd		false
notifyOnDelete		false

Allowed subelements of <import>

None

4.8.23 <attributes>

<attributes> in the <port> of the source system contains some <attribute> elements that are created by the DirX engine for being used as input for being mapped to the <attribute> elements in the <attribute> of the <port> of the target system.

Example

In the <port> of the source system:

```
<attributes>
  <attribute name="sn"/>
  <attribute name="givenName"/>
  <attribute name="dn"/>
  <attribute name="mail"/>
</attributes>
```

```
In the <port> of the target system:

<attributes>
  <attribute name="userId"/>
  <attribute name="objectclass"/>
  <attribute name="domain"/>
  <attribute name="displayName"/>
  <attribute name="homeTimeZone"/>
  <attribute name="defaultLocale"/>
  <attribute name="vanityCodeString"/>
  <attribute name="password"/>
  <attribute name="pin"/>
</attributes>
```

Allowed subelements of `<attributes>`

None

4.8.24 <attribute>

In the `<port>` of the source system, the name of `<attribute>` only indicates the name of a property that should be retrieved from the source system. In the `<port>` of the target system, the name of `<attribute>` only indicates the name of a property that should be provisioned to the target system. <mappingDefinition> and its subelements specify how to map from the `<attribute>` in the `<port>` of the source system to the `<port>` of the target system.

Table 36: Attributes of <attributes>

Attribute	Description	Example Value
`name`	The name indicates the name of a property, e. g. a given name, a family name or an email address.	`dn`, `givenName`, `dn`, `mail`, `userId`, `objectClass`, `domain`, `displayName`, `homeTimeZone`, `defaultLocale`, `password` or `pin`
`readOnly`	This attribute is used in retrieval operations only. It is ignored in update operations. It is even forbidden to update this attribute.	Default: `false`
`onAddOnly`	This attribute is only used when creating a new object; it is not used in MODIFY operations.	Default: `false`

Attribute	Description	Example Value
`checkModifications`	This flag is useful for handling multi-value attributes with a huge number of values, e. g. large groups. This attribute will not be read when searching the joined entry. Nevertheless it needs to be updated. The flag value false guarantees that the update is always done without comparing against existing values in the joined entry. In this case you need Java classes, i. e. mappingType (see mappingType) must have the value javaclass or javasource. They typically set the modify operation to “add” or “delete” instead of “replace”, which is automatically set for the other mapping types.	Default: `true`
`modifyAlways`	This flag is to be set to true, if the connector always needs the current value(s) for this `<attribute>`, whether they were changed or not. If this flag is set to false, the join engine puts a modification for this `<attribute>` into the modify request only if the value changed compared with the joined entry.	Default: `false`
`notInSchema`	A flag indicating that the `<attribute>` does not exist in the database. Such an <attribute> will be treated as “specific `<attribute>`” (type / value pair) and stored into the dxrOptions attribute when updating the identity domain.	Default: `false`
`matchRule`	A flag defining the way values of the `<attribute>` are compared for calculating differences.	`caseExact` and `caseIgnore` Default: `caseIgnore`

Attribute	Description	Example Value
retrievable	This flag defines whether the <attribute> should be passed as “requestedAttribute” to the search operation. If set to false, the <attribute> will not be requested in the search operation. That flag is mainly used with the Lotus Notes Connector. The <attribute> is updated in the Notes target system, but on the way back it is not read as the <attribute> does not exist in or cannot be read from the target system.	Default: true

Allowed subelements of <attribute>

None

4.8.25 <mappingDefinition>

Subelements of <mappingDefinition> specify how the mapping of an <attribute> in the <port> of the source system to an <attribute> in the <port> of the target system is done. Only an <attribute> that is not flagged readOnly is allowed to be mapped. The mapping of an identifier is specified in an <idMapping>, the mapping of another property is specified in an <attrMapping>. You may also define a <postMapping>. It is processed after all properties and the identifier were mapped. All these mapping elements follow more or less the same structure: a name and a mappingType attribute, a <value> and optionally a <code> element or a Java class name for complex mappings.

Table 37: Attributes of <mappingDefinition>

Attribute	Description	Example Value
packagename		siemens.dxm.map.ldap.accounts.to
xmlns:dsml	DSML namespace	urn:oasis:names:tc:DSML:2:0:core
xmlns:spml	SPML namespace	urn:oasis:names:tc:SPML:1:0

Example

```
<mappingDefinition
  packagename=""
  xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core"
  xmlns:spml="urn:oasis:names:tc:SPML:1:0">
  <idMapping
```

```
    classname="Id"
    coderef="idMapping"
    mappingType="direct"
    type="urn:oasis:names:tc:SPML:1:0#DN">
    <value>dn</value>
  </idMapping>

  <attrMapping mappingType="direct" name="sn">
    <value>lastName</value>
  </attrMapping>

  <attrMapping mappingType="direct" name="givenname">
    <value>firstName</value>
  </attrMapping>

  <attrMapping mappingType="direct" name="mail">
    <value>mail</value>
  </attrMapping>
</mappingDefinition>
```

Allowed subelements of `<mappingDefinition>`

- <idMapping>
- <attrMapping>
- <postMapping>

4.8.26 <idMapping>

An `<idMapping>` calculates the identifier of the mapped target entry. The `name` and `mappingType` attributes are identical to those in the `<attrMapping>` element.

The `<idMapping>` element requires the additional `type` attribute to denote the type of identifier as requested in SPMLv1.

Table 38: Attributes of <idMapping>

Attribute	Description	Example Value
`type`	`type` denotes the type of identifier as requested in SPML V1.	`urn:oasis:names:tc:SPML:1:0#DN`
mappingType	Mapping type describing how the value of a property is determined, i. e. whether it is constant, a direct assignment or calculated	See mappingType
`classname`	The name of the Java class realizing the mapping. It is only evaluated for the mapping types `javaclass` and `javasource` (see below).	`Id` or `IdMapper`
coderef	This is just for administration purposes at design time. It is not evaluated at runtime.	`idMapping`

Example

```
<idMapping
  type="urn:oasis:names:tc:SPML:1:0#DN"
  mappingType="direct">
  <value>dxrPrimaryKey</value>
</idMapping>
```

This mapping definition produces an identifier of type `DN` and simply takes the `dxrPrimaryKey` property from the source entry.

Allowed subelements of `<idMapping>`

- <value>

4.8.27 <value>

`<value>` only is used to indicate the value of a `<idMapping>` or `<-attrMapping>`. No attributes are used.

Allowed subelements of `<value>`

None

4.8.28 <attrMapping>

The `<attrMapping>` element specifies how the mapping for an `<attribute>` in the `<port>` of the source system to an `<attribute>` in the `<port>` of the target system is done. Only an `<attribute>` that is not flagged `readOnly` is allowed to be mapped.

Table 39: Attributes of <attrMapping>

Attribute	Description	Example Value
`name`	The name of the target `<attribute>`. This attribute only is used in the `<port>` of the target system.	`objectClass`, `domain`, `displayName`, `homeTimeZone`, `defaultLocale`, `vanityCodeString`, `password` or `pin`
mappingType	Mapping type describing how the value of a property is determined, i. e. whether it is constant, a direct assignment or calculated	See mappingType
`classname`	The name of the Java class realizing the mapping. It is only evaluated for the mapping types `javaclass` and `javasource` (see mappingType).	`Id` or `IdMapper`
coderef	This is just for administration purposes at design time. It is not evaluated at runtime.	

Example

The following XML elements in the <port> of the source system specify that the lastName LDAP property of the source system is mapped via the sn property in the DirX engine to the displayName property in the target system:

```
<attributes>
 <attribute name="sn"/>
 ...
</attributes>
<mappingDefinition ...
 <attrMapping mappingType="direct" name="sn">
  <value>lastName</value>
 </attrMapping>
</mappingDefinition>

and the following XML elements in the <port> of the target
 system

<attributes>
 <attribute name="displayName"/>
 ...
</attributes>
<mappingDefinition
 ...
 <attrMapping mappingType="direct" name="displayName">
  <value>sn</value>
 </attrMapping>
</mappingDefinition>
```

Allowed subelements of <attrMapping>

- <value>: See mappingType for the values.
- <code>: This is only evaluated if mappingType has the value "javasource".

For examples see <mappingDefinition>.

4.8.29 mappingType

The mappingType attribute is mandatory and describes the type of mapping. It accepts one of the following values:

- constant

  The target property is set to the constant value(s) listed in the <value> subelements. This is typically applied to the objectclass attribute in LDAP systems.

  Example:

  ```
  <attrMapping
    name="objectclass"
    mappingType="constant">
    <value>User</value>
  </attrMapping>
  ```

- `direct`

  Direct mapping from the source property to the target property. It applies both for single and for multi-value attributes. The <value> subelement contains the name of the source attribute.

  Example:

```
<attrMapping
  mappingType="direct"
  name="givenname">
  <value>firstName</value>
</attrMapping>
```

- `simpleexpression`

  This allows composing strings using simple bean (placeholder) notations instead of Map and DsmlValue Java notation. The attribute mapper evaluates the expression at runtime.

  The following simple expressions are interpreted. They are basically a concatenation of strings:

```
EXPR [+ EXPR] …
```

  with the following definitions:

  – `EXPR ::= VAR | STRING`
  – `VAR ::= ${source.<attrType>} |`

    `${joinedEntry.<attrType>} |`

    `${env.<name>}`
  – `STRING := "..."` (i. e. a constant string)

  where

  – `${source.<attrType>}` is an property taken from the source entry,
  – `${joinedEntry.<attrType>} is an` property taken from the joined entry and
  – `${env.<name>} is a` property taken from the environment.

  Limitations: Only single valued properties are handled. In other words: only the first value is used.

  Examples:

```
${source.dxrName} + ${env.ads_upn_extension};
${source.sn} +  "," + ${source.initials} + "-" +
 ${source.givenName}
```

  The <value> element may occur more than once. This is for situations, where we want to provide a number of alternative mappings. The expressions are evaluated in the configured sequence. The first expression resulting in a non-null value is taken as the mapping result.

  NOTICE: If some substrings are concatenated and one of them is empty, the whole expression results in an empty string.

  Here is an example with two expressions:

```
<attrMapping name="Validity" type="simpleexpression">
  <value>${source.validity}</value>
```

```
  <value>${env.validity}</value>
<attrMapping>
```

First the `validity` attribute of the source entry is evaluated. If it is non-empty, this value is taken as the mapping result. Otherwise, the `validity` property from the environment is taken.

- `javaclass`

  The `classname` attribute contains the full name of a Java class that has to perform the mapping for the attribute. It has to implement one of three java interfaces depending on the parent element:

  – `<attrMapping>`: `com.siemens.dxm.join.api.IMapAttribute`
  – `<idMapping>`: `com.siemens.dxm.join.api.IMapIdentifier`
  – `<postMapping>`: `com.siemens.dxm.join.api.IPostMapping`

  The attribute mapper simply instantiates the class with its default constructor, calls the appropriate interface method and takes the return value as the mapping result.

- `javasource`

  The Java class implementing the mapping interface (see above) is not loaded from a file, but from the XML element.

  The `<code>` subelement contains the compilation unit for the Java class. It is assumed that an appropriate tool at design time (e. g. DirX Identity Manager) compiles some Java source and fills in the compilation unit. The attribute mapper loads the class from the XML content and calls the corresponding mapping interface method.

  It builds the full class name by concatenating the `packagename` attribute of the `<mappingDefinition>` element with the simple name taken from the `classname` attribute of the parent `<*Mapping>` element.

  Here is a sample configuration snippet:

```
<mappingDefinition
 packagename="siemens.dxm.map.ldap.accounts.to">
  <attrMapping
  classname="DxrTSState"
  mappingType="javasource" name="dxrTSState">
  <code>yv66vg … </code>
  </attrMapping>
…
```

  The attribute mapper calls the class `siemens.dxm.map.ldap.accounts.to.DxrTSState`.

  For further examples see <mappingDefinition>.

4.8.30 <postMapping>

The `<postMapping>` element is optional. Post mapping is performed after the identifier and all the properties have been mapped. It only allows a Java implementation which can work on the whole mapped entry produced so far. The post mapping is especially intended to cope with dependencies between the target properties.

Table 40: Attributes of <postMapping>

Attribute	Description	Example Value
mappingType	Mapping type describing how the value of a property is determined	Only `javaclass` and `javasource` are allowed, see mappingType.
`classname`	The name of the Java class realizing the mapping.	`Id` or `IdMapper`

Allowed subelements of `<aattrMapping>`

- <value>
- <code>

4.8.31 <userhook>

Table 41: Attributes of <userhook>

Attribute	Description	Example Value
implementationLanguage	The programming language that might be used for implementation of extensions	`java`

Index

A