HP OpenView Select Identity

**Installation Guide
for the UNIX Connector
for Solaris 8 Systems**

Software Version: 3.0

July 2004

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

Select Identity uses software from the Apache Jakarta Project including:

- Commons-beanutils.
- Commons-collections.
- Commons-logging.
- Commons-digester.
- Commons-httpclient.

- Element Construction Set (ecs).
- Jakarta-poi.
- Jakarta-regexp.
- Logging Services (log4j).

Additional third party software used by Select Identity includes:

- JasperReports developed by SourceForge.
- iText (for JasperReports) developed by SourceForge.
- BeanShell.
- Xalan from the Apache XML Project.
- Xerces from the Apache XML Project.
- Java API for XML Processing from the Apache XML Project.
- SOAP developed by the Apache Software Foundation.
- JavaMail from SUN Reference Implementation.
- Java Secure Socket Extension (JSSE) from SUN Reference Implementation.
- Java Cryptography Extension (JCE) from SUN Reference Implementation.
- JavaBeans Activation Framework (JAF) from SUN Reference Implementation.
- OpenSPML Toolkit from OpenSPML.org.
- JGraph developed by JGraph.
- Hibernate from Hibernate.org.

Support

Please visit the HP OpenView web site at:

http://openview.hp.com/

There you will find contact information and details about the products, services, and support that HP OpenView offers.

You can go directly to the support web site at:

http://support.openview.hp.com/

The support web site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

contents

Installing the UNIX Connector

The UNIX connector enables HP OpenView Select Identity to manage user data in UNIX. It is a one-way connector and pushes changes made to user data in the Select Identity database to a target UNIX server. This connector is generic and can be used to connect to any UNIX data source. The mapping file controls how Select Identity fields are mapped to UNIX fields.

The UNIX connector is packaged in two JAR files: one containing the mapping files called `unixtelschema.jar` and one containing the rest of the source code called `UnixTelnetConnector.rar`. These files are located in the `UNIX` directory on the Select Identity Connector CD.

- Note that this procedure installs a connector that supports UNIX for Solaris 8 running Telnet. The application server used in this procedure is WebLogic 8.1.
- Perform this procedure after the Select Identity product installation.

To install the UNIX connector on the Select Identity server, complete these steps.

1 If necessary, stop the application server.

2 A Select_Identity directory was created on the web server during the product installation. Copy the UnixTelnetConnector.rar file and extract the unixtelschema.jar file from the Select Identity Connector CD to this directory.

After you extract files from the .jar file, remove the unixschema.jar file from the Select_Identity directory. This ensures that the web server references the contents of .jar and not .jar itself.

3 The startweblogic.cmd file was edited during the product installation to specify the location of the Select_Identity directory. The startweblogic.cmd file resides in the *WebLogic_home*/ user_projects/domains/*domain*/ directory on WebLogic 8.1.

Ensure that this line set CLASSPATH=%CLASSPATH% in the startweblogic.cmd file references C:\Select_Identity in the class path.

4 If necessary, start the web server.

5 Log in to the WebLogic Administrator Console.

6 Navigate to *My_domain* –> **Deployments** –> **Connector Modules**.

7 Click **Deploy a New Connector Module**.

8 Locate and select the UnixConnector.rar file from the list. It is stored in the Select_Identity directory.

9 Click **Target Module**.

10 Configure the connector according to your environment. Open **Edit Connector Descriptor**, and add the following connector parameters:

— Connector Name: Specify any name for the Unix connector

— Pool Name: eis/UnixTelnetCon

11 Select the **My Server** (which is your server instance) check box.

12 Click **Continue**. Review your settings.

13 Keep all default settings and click **Deploy**.

14 The Status of Last Action column should display Success.

To test the connector, log in to the Select Identity application and deploy the connector through the Connector pages. See the *HP OpenView Select Identity Administrator Guide* for procedures.

2

Understanding the UNIX Mapping File

Each connector is deployed with an XML mapping file that contains the attributes required by the operating system. This file is used to map user account additions and modifications from Select Identity to the system resource. When you deploy a resource through the Select Identity Resources pages, you can review this file.

You can create attributes that are specific to Select Identity through the Attributes pages in the Select Identity client. These attributes can be used to associate Select Identity user accounts with system resources by mapping them to the connector mapping file described in this chapter. This process becomes necessary because a single attribute "username" can have a different definition on three different resources, such as "login" for UNIX, "UID" for a database, and "userID" on a Windows server.

This file does not need to be edited unless you want to map additional attributes to your resource. If attributes and values are not defined in this mapping file, they cannot be saved to the resource through Select Identity.

The UNIX connector provides the `UnixConnector.xml` mapping file. The file is created in XML, according to SPML standards, and is bundled in a JAR file called `unixschema.jar`.

General Information

The following operations can be performed in the mapping file:

- Add a new attribute mapping
- Delete an existing attribute mapping
- Modify attribute mappings

Here is an explanation of the elements in the XML mapping file:

- **<Schema>**, **<providerID>**, and **<schemaID>**

  Provides standard elements for header information.

- **<objectClassDefinition>**

  Defines the actions that can be performed on the specified object as defined by that name attribute (in the <properties> element block) and the Select Identity-to-resource field mappings for the object (in the <memberAttributes> block). For example, the object class definition for users defines that users can be created, read, updated, deleted, reset, and expired in UNIX.

  - **<properties>**
    Defines the operations that are supported on the object. This can be used to control the operations that are performed through Select Identity. The following operations can be controlled:

    — Create (CREATE)

    — Read (READ)

    — Update (UPDATE)

    — Delete (DELETE)

    — Enable (ENABLE)

    — Disable (DISABLE)

    — Reset password (RESET_PASSWORD)

    — Expire password (EXPIRE_PASSWORD)

    — Change password (CHANGE_PASSWORD)

The operation is assigned as the name of the <attr> element and access to the operation is assigned to a corresponding <value> element. You can set the values as follows:

— true — the operation is supported by the connector

— false — the operation is not supported by the connector and will throw a permission exception

— bypass — the operation is not supported by the connector but will not throw any exception; the operation is simply bypassed

Here is an example:

```
- <objectClassDefinition name="User" description="Unix
User">
  - <properties>
    - <attr name="CREATE">
        <value>true</value>
      </attr>
    - <attr name="READ">
        <value>true</value>
      </attr>
```

- **<memberAttributes>**
  Defines the attribute mappings. This element contains <attributeDefinitionReference> elements that describe the mapping for each attribute. Each <attributeDefinitionReference> must be followed by an <attributeDefinition> element that specifies details such as minimum length, maximum length, and so on.

  Each <attributeDefinitionReference> element contains the following attributes:

  — Name — the name of the reference.

  — Required— if this attribute is required in the provisioning (set to true or false).

  — Concero:tafield — the name of the Select Identity resource attribute.

  — Concero:resfield — the name of the physical resource attribute from the resource schema. If the resource does not support an explicit schema (such as UNIX), this can be a tag field that indicates a resource attribute mapping.

— Concero:isKey — An optional attribute that, when set to true, specifies that this is the key field to identify the object on the resource. Only one <attributeDefinitionReference> can be specified where isKey="true". This key field does not need to be the same as the key field of the identity object in Select Identity.

— Concero:init — An optional attribute that identifies that the attribute is initialized with the value of the attribute passed in from Select Identity.

Here is an example:

```
- <memberAttributes>
  <attributeDefinitionReference name="GroupName"
required="true" concero:tafield="GroupName"
concero:resfield="gname" concero:isKey="true" />
  </memberAttributes>
  </objectClassDefinition>
- <attributeDefinition name="GroupName"
description="GroupName" type="xsd:string">
   - <properties>
    - <attr name="minLength">
       <value>1</value>
     </attr>
    - <attr name="maxLength">
       <value>8</value>
     </attr>
    - <attr name="pattern">
      - <value>
        - <![CDATA[ [a-zA-Z0-9@]+ ]]>
        </value>
      </attr>
    </properties>
  </attributeDefinition>
```

The interpretation of the mapping between the connector field (as specified by the Concero:tafield attribute) and the resource field (as specified by the Concero:resfield attribute) is determined by the connector. The UNIX connector has code to interpret the mappings in one way, as follows:

— The connector attribute names are specified in square braces, like this: [xyz]. The value of attribute xyz is taken from the UserModel during provisioning.

— Composite attributes can be specified in the UNIX connector mapping file. To do this, specify [attr1] xxxx [attr2] as the connector attribute. This specifies that the value of the attr1 and attr2 attributes should be combined with the string xxxx to form a mapping for the specified resource field. UNIX connector has code to handle these composite mappings.

- **<attributeDefinition>**

  Defines the properties of each object's attribute. For example, the attribute definition for the Directory attribute defines that it must be between one and 50 characters in length and can contain the following letters, numbers, and characters: a-z, A-Z, 0-9, @, +, and a space.

  Here is an excerpt from the `UnixConnector.xml` file:

```
- <attributeDefinition name="Directory" description="Directory"
type="xsd:string">
   - <properties>
    - <attr name="minLength">
       <value>1</value>
     </attr>
    - <attr name="maxLength">
       <value>50</value>
     </attr>
    - <attr name="pattern">
      - <value>
       - <![CDATA[ [a-zA-Z0-9/]+ ]]>
      </value>
     </attr>
   </properties>
  </attributeDefinition>
```

- **<concero:entitlementMappingDefinition>**

  Defines how entitlements are mapped to users.

- **<concero:objectStatus>**

  Defines how to assign status to a user.

- **<concero:relationshipDefinition>**

  Defines how to create relationships between users.

UNIX Information

The UNIX connector supports the following identify information to be provisioned on the UNIX system.

You can add, modify, delete attributes once you are familiar with the contents of this file. See the *HP OpenView Select Identity Connector Developer Guide* for more information about attributes and mapping information.

The Select Identity resource attributes are editable. They reflect the identity information as seen within the Select Identity system.

The logical UNIX attributes are attributes of user accounts on the UNIX system. These attributes cannot be changed.

SI Resource Attribute	Logical UNIX Attribute	Description
UserName	username	Unix login name
Password	password	Login password
FirstName	F	First Name
LastName	L	Last Name
FullName	comment	Comment section in /etc/passwd
Directory	directory	User's home directory
Shell	shell	Unix login shell
DefaultGroup	defaultgroup	Default Group membership

<div style="text-align: right;">**3**</div>

Installed Scripts

The UNIX connector carries out operations using a tool called Expect. This tool must be installed on the web server running Select Identity.

- Download expect at **http://expect.nist.gov/** and install it on the local disk. When deploying the UNIX system as a resource in the Select Identity interface, you will provide the location of the expect executable in the "access info" section.

- Copy Select Identity Expect scripts in `Solaris-expect-scripts.zip` from the Select Identity Connector CD. Select Identity distribution for UNIX connectivity includes scripts to perform add, modify, delete, and get operations on the UNIX system. Install the scripts in the local disk.

There are some special variables and classes available to the script when executing in the UNIX connector. Each is described in this chapter.

Supplied Scripts

The following scripts are provided during the connector installation.

`adduser.exp`

This is the script used to add a new user to the UNIX system.

listuser.exp

This script lists all users configured on the UNIX system.

changepassword.exp

This script changes the password of a user on the UNIX system.

changestatus.exp

This script changes a users status on the UNIX system.

deleteuser.exp

This script deletes a user from the UNIX system.

dotest.exp

This script is executed whenever a new connector is created. It is responsible for validating the connection.

finduser.exp

This script is used to search for a user on the UNIX system.

genericcmd.exp

This script is used to enable the execution of any command required by Select Identity.

modifyuser.exp

This script is used to modify a user on the UNIX system.

Example adduser.exp Script

The following is a sample adduser.exp script.

```
#!/usr/local/bin/expect -f
#
#  Usage:  to add a new user on solaris
#
#     expect -f adduser.exp <IPaddress> <userName> <passwd>
<rootPasswd>
#                   "[<arg1>=<value1>]" "[<arg2>=<value2>]" ...
#
```

```
set force_conservative 0  ;# set to 1 to force conservative mode
even if
         ;# script wasn't run conservatively originally
if {$force_conservative} {
  set send_slow {1 .1}
  proc send {ignore arg} {
    sleep .1
    exp_send -s -- $arg
  }
}

set unix_field(0) "username"
set unix_field(1) "password"
set unix_field(2) "comment"
set unix_field(3) "directory"
set unix_field(4) "shell"
set unix_field(5) "defaultgroup"
set unix_field(6) "expiredate"
set unix_field(7) "inactive"
set unix_field(8) "uid"
set unix_field(9) "auth"
set unix_field(10) "profile"
set unix_field(11) "role"
set unix_field(12) "gname"

set cmdarg_map(comment) "-c"
set cmdarg_map(directory) "-d"
set cmdarg_map(shell) "-s"
set cmdarg_map(defaultgroup) "-g"
set cmdarg_map(expiredate) "-e"
set cmdarg_map(inactive) "-f"
set cmdarg_map(uid) "-u"
set cmdarg_map(auth) "-A"
set cmdarg_map(profile) "-P"
set cmdarg_map(role) "-R"
set cmdarg_map(gname) "-G"

#This procedure extracts the arguments from attr_array and
#build a new array with name as index and value

proc extract_value {attr_values } {
#The array is of the for name=value
global arg_values
      foreach item $attr_values {
              set eqIdx [string first "=" $item]
#             puts "$item"
```

```
#              puts "$eqIdx"
              if { $eqIdx != -1 } {
                     set name [string range $item 0 [expr $eqIdx -
1]]
                     set value [string range $item [expr $eqIdx + 1]
[string length $item]]
#                     puts "Name=$name Value=$value len=[string
length $value]"
                     set arg_values([string tolower $name]) $value

              }
       }
}

set prompt "(%|#|\\$) $"
set server [lindex $argv 0]
set login [lindex $argv 1]
set loginpswd [lindex $argv 2]
set rootpswd [lindex $argv 3]

extract_value $argv
#      foreach item [array names arg_values] {
#              puts "$item $arg_values($item)"
#      }

# additional add arguments
# example: "-d /home/dir1 ..."

set errcode 0
set errcode [ catch {
       set username $arg_values(username)
       set password $arg_values(password)
}]

#puts "$errcode"

if {$errcode != 0} {
       send_user "ERROR: username/password not present"
       exit 1
}
```

4

Uninstalling the UNIX Connector

If you need to uninstall a connector from Select Identity, make sure that the following are performed:

- All resource dependencies have been removed.
- The connector has been deleted through the Select Identity client Connectors pages.

Perform the following to delete a connector:

1 Log in to the WebLogic Server Console.

2 Navigate to *My_Domain* –> **Deployments** –> **Connector Module**.

3 Click the delete icon next to the connector that you want to uninstall.

4 Click **Yes** to confirm the deletion.

5 Click **Continue**.