HP OpenView Select Identity

Installation Guide
for the UNIX Connector
for Solaris 9 Systems, with SSH

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

This product includes software developed by the Apache Software Foundation (http://www.apache.org/). Portions Copyright (c) 1999-2003 The Apache Software Foundation. All rights reserved.

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.

Trademark Notices

HP OpenView Select Identity is a trademark of Hewlett-Packard Development Company, L.P.

Microsoft, Windows, the Windows logo, and SQL Server are trademarks or registered trademarks of Microsoft Corporation.

Sun™ workstation, Solaris Operating Environment™ software, SPARCstation™ 20 system, Java technology, and Sun RPC are registered trademarks or trademarks of Sun Microsystems, Inc. JavaScript is a trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

This product includes the Sun Java Runtime. This product includes code licensed from RSA Security, Inc. Some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/.

IBM, DB2 Universal Database, DB2, WebSphere, and the IBM logo are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

This product includes software provided by the World Wide Web Consortium. This software includes xml-apis. Copyright © 1994-2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institute National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/

Intel and Pentium are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

AMD and the AMD logo are trademarks of Advanced Micro Devices, Inc.

BEA and WebLogic are registered trademarks of BEA Systems, Inc.

VeriSign is a registered trademark of VeriSign, Inc. Copyright © 2001 VeriSign, Inc. All rights reserved.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

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

1

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 `unixschema.jar` and one containing the rest of the source code called `UnixConnector.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 9 with SSH. The application server used in this procedure is WebLogic 8.1.
- Perform this procedure after the Select Identity product installation.
- Make sure that SSH is installed on all UNIX systems to which you want Select Identity to connect and store information.

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 `UnixConnector.rar` file and extract the `unixschema.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. Start the web server.
4. 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.
5. Start the web server.
6. Log in to the WebLogic Administrator Console.
7. Navigate to ***My_domain* –> Deployments –> Connector Modules**.
8. Click **Deploy a New Connector Module**.
9. Locate and select the `UnixConnector.rar` file from the list. It is stored in the `Select_Identity` directory.
10. Click **Target Module**.
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.
15. Restart the application server.

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

3

Installed Scripts

The UNIX Connector uses a java based scripting engine called Bean Shell (www.beanshell.org). The Bean Shell scripts control the interactions between the unix machine and the Select Identity server. Copy the scripts from the Select Identity Connector CD (`Solaris-bsh-scripts.zip`), and install them on the UNIX system.

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

Special Variables

Variable Name	Type	Description
ExpectVerbose	Boolean	If true then server responses are echoed in the output
Login	String	The username to use when performing a login
loginPassword	String	The password to use when performing a login

Variable Name	Type	Description
rootPassword	String	The password to use when gaining super user privileges
scriptDir	String	The path to the Select Identity Bean Shell scripts
unixType	String	The type of unix operating system that hostname is running.
Prompt	String	A regular expression to match the unix host prompt.
Args	String	A name value list of extra arguments used by the script.
scriptResponse	String	The output from the last executed script.
Session	Expect Session	This is the class that provides communication to/from the unix host

Special Methods and Functions

These methods and functions enable Select Identity to send and receive data with UNIX systems.

Method Name	Parameters	Description
Session.sendExpect	Pattern errorPattern timeout	Success case text to expect Error case text to expect Timeout in seconds to wait for Pattern
loginExternal	None	Establishes super user privileges

Method Name	Parameters	Description
logoutExternal	None	Revokes super user privileges
setPassword	UserId OldPassword NewPassword	User to set password for Old password for UserId New password for UserId

Supplied Scripts

The following scripts are provided during the connector installation.

`common.bsh`

This provides common commands for several scripts.

`adduser.bsh`

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

`listusers.bsh`

This script lists all users configured on the UNIX system.

`changepasswd.bsh`

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

`changestatus.bsh`

This script changes a users status on the UNIX system.

`deleteuser.bsh`

This script deletes a user from the UNIX system.

`dotest.bsh`

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

`finduser.bsh`

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

`genericcmd.bsh`

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

`modifyuser.bsh`

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

Example adduser.bsh Script

The following is a sample `adduser.bsh` script.

```
1  /**
2  #
3  #  Usage:  to add a new user on solaris
4  #  Required vars: rootPassword, username, password, args
5  #
6  */
7  source( scriptdir + "/common.bsh");
8
9  // User's initial password will be this
10  // will be changed to the given password subsequently
11  tempPasswd="temp012345";
12
13  optionalArgs = prepareArgs(args);
14
15  loginExternal();
16  addusercmd = _addusercmd + " " + optionalArgs + " " + username;
17
18  try {
19      session.sendExpect( addusercmd, prompt, _addusercmd +
":.*$|" + username + " exists$", 10 );
20      session.sendExpect( "passwd " + username, ".*assword:",
"BAD.*", 10 );
21      session.sendExpect( tempPasswd, ".*assword:", 10 );
22      session.sendExpect( tempPasswd, prompt, 10 );
23
24      // Change the temp password to the actual one
25      setPassword( username, tempPasswd, password );
26
27  }
28  finally {
```

```
29      logoutExternal();
30  }
```

Lines 1-6 are comments.

Line 7 calls source to make available all of the commands in the `common.bsh` script in this script.

Line 11 defines a variable to hold the value of the temporary password assigned for the user.

Line 13 uses of one of the functions defined in `common.bsh` to parse any additional arguments passed to this script.

Line 15 obtains super user privileges.

Line 16 builds the command that will be executed for adding the new user. `_addusercmd` is defined in `common.bsh` based on the current `unixType`. `optionalArgs` was initialized on line 13 and `username` was initialized by Select Identity.

Lines 18-30 is a try block. Inside the try block are the actual commands to create the user.

Line 19 we send the `adduser` command via the session objects `sendExpect` method. The expected response is the UNIX prompt. The script uses a regular expression (`_addusercmd + ":.*$|" + username + " exists$"`) to detect any errors from the command. A timeout of 10 seconds is specified.

Line 20 executes the `passwd` command for `username`.

Line 21, if line 20 is successful, the script sends the `tempPasswd`, looking for a password: response.

Line 22 sends the `tempPasswd` again and looks for the UNIX prompt to indicate success.

Line 25 calls the `setPassword` function defined in `common.bsh` to set the users password.

Line 29 in the final block we revoke the super user privileges. It is highly recommended that you always place the `logoutExternal` call in a final block to insure execution, even if an exception occurs.

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**.