HP OpenView Select Identity

Software Version: 4.13

New Information

Document Release Date: May 2007
Software Release Date: May 2007

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.

HP OpenView Select Identity (OVSI) 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 HP OpenView 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
- BouncyCastle engine for keystore management, bouncycastle.org

Trademark Notices

Unix® is a registered trademark of The Open Group.

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.

Java™ is a US trademark of Sun Microsystems, Inc.

Support

Please visit the HP OpenView support web site at:

http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valuable support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

http://www.managementsoftware.hp.com/passport-registration.html

New Information

This addendum delivers information about Select Identity that became available after the product documentation was completed and immediately prior to actual release of the product. An addendum is provided as needed to accompany the documentation for a specific product release.

This addendum is specific to Select Identity version 4.13.

The content in this addendum is intended for more than one type of reader. Some of the sections provide a use-context overview, for instance, while others provide technical detail intended for developers.

In general, this information will be incorporated into the appropriate parts of the product documentation during the next available release cycle.

This addendum contains the following sections:

- Documentation Updates and Corrections
- Attribute Validation API Enhancement
- Request Dependency Enhancements
- Service Search in My Identity
- Linked Secondary User Registration
- Keystore Sample Commands
- Additional Migration Utility Information

Documentation Updates and Corrections

This section provides corrected and updated information discovered after the last release of the documentation item in which the information would normally appear.

Deleting or Modifying User Accounts on a Composite Service

This information will be added to a future revision of the *HP OpenView Select Identity Online Help for Administration*.

You cannot delete or modify user accounts on a composite service. This is because the composite service feature is a grouping mechanism for services, but does not directly take up the membership list for the grouped service; instead user accounts are added to each service as a result of being added to the composite.

To delete or modify user accounts on a composite service, you must perform these operations on member services individually.

Workflow External Calls Documentation Correction

This update will be added to the next available release of the *HP OpenView Select Identity External Calls Guide*.

Page eleven of this guide includes the following statement about the **WfExternalCall** interface:

> *External calls invoked by workflow instances must implement this interface. It provides the contract between Select Identity and an external stage in a workflow. Select Identity can invoke classes that implement this interface to perform actions in a workflow. This class returns a collection of approvers.*

The last sentence in this statement is incorrect. Callouts to the **WfExternalCall** interface do *not* return a collection of approvers.

A class that implements the **WfExternalCall** interface has no method to return a collection of approvers. It only returns the following values:

```
'java.util.Map params':

WF_COMMON_ATTRS

WF_PARAM_ADMINUSERID

WF_PARAM_REQUESTID

WF_PARAM_SERVICENAME

WF_PARAM_WORKFLOWINSTID
```

You can use the values above in the method `WfExternalCallStatus()`.

To return a collection of approvers, use the **WfSelectApproverIntf** interface. This is reflected in JavaDoc, as follows:

```
java.util.Collection getApprover(RequestTarget reqTarget,
java.util.HashMap attrs)
```

Select Identity calls this function to get an approver.

Attribute Validation API Enhancement

This section describes an enhancement to the attribute value validation functionality at the API level and is therefore supplemental to the developer-level information provided in the *HP OpenView Select Identity External Calls Guide*.

Custom Value Validation Logic

This section provides a high-level use-context summary of the attribute validation enhancement.

Select Identity supports the `TAValueValidationIntf` interface as a mechanism for custom value validation logic. This is accomplished by implementing this interface and registering the implementation using the external call user interface as documented in the *Select Identity Online Help for Administration* and then attaching it to an attribute.

Select Identity calls this implementation during the attribute value validation process. The system sends only the current value or values of an attribute with basic information such as `UserName`, `BizKey`, and `GUID`.

The enhancement in Release 4.13 enables the request broker to pass the `requestTarget` to the external system.

When adding a new user, `requestTarget` retains all attribute values. However, for other request types, `requestTarget` retains only modified values. The external call may therefore be required to query the Select Identity database for any other value if needed.

The validation call can retrieve `requestTarget` from the `maps` argument by using the `HPSIRequestTarget` constant.

```
RequestTarget  reqTarget = (RequestTarget  )
args.get("HPSIRequestTarget");
```

Any validation error from the external call will be displayed in the drop-down list on the top of the page.

SampleEmailValidation External Call

This sample external call validates the value of the `Email` attribute against the following format:

```
<FirstName>.<LastName>@<company>.com
```

Purpose

The purpose of this external call is to demonstrate that more than one attribute on the form can be used to validate an attribute using the attribute value validation external call function.

The attribute value validation external call has access to a new `RequestTarget` object passed in as a parameter to the validation external call. The `RequestTarget` object can be accessed from the collection using the following key:

```
PARAM_REQUEST_TARGET
```

This external call expects the following attributes to be defined in the service. The `Email` attribute listed below is the system-defined attribute, and therefore its name is fixed:

- `FirstName`
- `LastName`
- `Email`
- `Company`

The external call performs the following validations on these attribute values. Validation exceptions are reported in the following format:

- `RequestTarget` contains the following attribute values:
  - `FirstName`
  - `LastName`
  - `Email`
  - `Company`
- `Email` attribute value adheres to the following format:

  `FirstName.LastName@company.com`
- Validate that the values of the `Firstname` and `Lastname` attributes from the request target are present in the email address (`Email`). Comparisons are not case-sensitive.

External Call Setup Procedure

To set up the `SampleEmailValidation` external call, perform the following steps:

1. Copy the external call `.jar` file to the Web application server directory where `.jar` files are stored for Select Identity use. (This should be a shared file system folder in the case of an installation on a cluster).
2. Register the external call using the following information:

   Classname: `com.hp.ovsi.extcall.validation.SampleEmailValidation`

   Call Type: `AttributeValueValidation`

   Class Path: Directory path to the `.jar` file that contains the external call

Request Dependency Enhancements

Request dependency processing or sterilization was initially introduced in Select Identity 4.x releases. The feature processes requests in order of arrival at the parent request level, unlike in previous releases where requests for a given user were processed in random order.

Request dependency was implemented to avoid conflicts of values between requests submitted for the same user.

Disabling and Re-Enabling Request Dependency

By default request dependency is enabled unless the following property is set to `true` in the `TruAccess.properties` file:

```
hp.si.delegated.request.nodependency
```

Enhancements in Release 4.13

The following enhancements are implemented in release 4.13:

- Service requests are not dependent on approvals for other service requests. This functionality is now the same as was found in Version 3.3.1.
- Serial processing of multiple requests must wait on pending requests before the next request starts executing.
- User account termination requests do not wait behind previous requests for the same user account The system processes them immediately upon arrival.
- Password reset or change requests are processed immediately.
- If an **AddNewUser** request is pending, the system rejects subsequent attempts to add the user to the same service.

Service Search in My Identity

This section provides an end-user summary of changes to the Select Identity My Identity self-service features. Additional technical detail is provided in the second part of the section.

Self-service users can request additional service subscriptions via the Select Identity **My Identity** tab. In earlier releases, all unsubscribed services are available for self-service subscription requests.

Release 4.12 enhances the self-service subscription request feature by providing a list of available services that has been filtered to eliminate unavailable or unnecessary entries from the service search results listed. Self-service requests for service membership only offer selections that match the following criteria:

- The user is not subscribed to the service at the time of the request submittal.
- The service is configured to enable self-service subscription requests.

Technical Detail

The enhancement uses request event setup in the root service role in the case of the My Identity **Subscribe to Service** action. This action considers the **Self-AddNewUser** event.

If the service role on the root service (at a minimum) is configured to enable the **Self-AddNewUser** event and the user is not subscribed to the service, then the service is displayed during service search in the **Subscribe to Service** action on the **My Identity** tab.

Linked Secondary User Registration

Select Identity provides the add account capability on the **Service Subscription** page to add a new secondary user account to an existing normal or primary account.

This section summarizes a customized implementation of this capability that provides linked secondary user service registration. The summary is followed by additional technical detail about the custom external call and workflow template on which the enhancement is based.

AddSecondaryUser ExternalCall

The `AddSecondaryUser` external call is included in release 4.13 to support linked secondary user registration.

This external call provisions a new secondary account when subscribing an existing user to a new service using delegated or self-service **AddNewUser** events. This external call does not add the current user account to the service; it creates a new secondary user account and designates the existing account as the primary user.

Linked secondary user registration is a custom external call derived from the **AddSecondaryUser** extension. Linked secondary user registration creates a secondary account for an existing user, and assigns the secondary account to any service, without restriction to those listed in the Service Subscription page.

The purpose of this external call is to intercept each subscribe-to-service request and convert it to a new secondary user account creation request combined with an **AddNewUser** event.

The Linked secondary user registration external call spawns a sub-workflow that provisions the new secondary user account with its service membership set to the current service in the subscribe-to-service event.

The user account on which the request was originated is converted to a primary account (if it is a normal account) and the new secondary account is linked with the converted primary account.

Limitations

This section briefly describes the functional limitations of this implementation of linked secondary user registration.

Limited Service Selection From Service Subscription Page

A new secondary account can only be assigned to the services displayed on the **Service Subscription** page for the current user. Consequently, the new secondary account cannot be assigned to a service that is not already listed on the **Service Subscription** page for the current user.

This limitation exists only when adding a new secondary user account (an **AddAccount** request) from the browser interface. **AddAccount** requests originated via Web services or during bulk add or reconciliation operations, are not affected by this limitation.

Multiple Service Registrations by a Single User

This external call does not prevent a single user from registering for the same service multiple times. In this case multiple secondary account will be created.

Primary User Service Assignment

Do not use this external call to assign services to a primary user, because it converts the subscribe-to-service request into a secondary account registration request, and therefore creates an unwanted secondary account for the primary user.

Sample Workflow

A sample workflow template, named **AddSecUserWorkFlow** is provided in Select Identity release 4.13 with the Linked secondary user registration external call. This template is available in Select Identity along with other default workflow templates.

The workflow template is included in the following `.jar` file:

```
Workflow+Template_AddSecUserWorkFlow_20070226233751.xml
```

This sample workflow is associated with the **AddSecondaryUser** external call. It does not define provisioning or post-provisioning blocks, because it leaves the existing user account unchanged and provisions a new secondary account onto the target service.

Use the **AddSecondaryUser** external call in the sample workflow to intercept delegated or self-add-service events and convert them to linked secondary user registration requests.

Linked Secondary User Account Creation Workflow Process

Use this external call only when a new secondary account needs to be created for each new service subscription request.

To create a linked secondary user registration account using the **AddSecondaryUser** external call, perform the following steps:

1. Register the **AddSecondaryUser** external call in the Select Identity browser interface. The external call does not require any parameters.
2. Select a service for which linked secondary account registration is required.
3. Define delegated add to service and/or self add to service events for the selected service by mapping these events to the sample workflow.
4. Associate the **User Generation** function with the **UserName** attribute.
5. Log on to Select Identity as a delegated administrator or end user.
6. In the case of a delegated administrator, subscribe any existing user account to the service you selected in step 2.
7. In the case of an end user, subscribe to the service identified in step 2 using the My Identity self-registration page.
8. Select Identity creates a new subscribe-to-service request. Make a note of the request ID.

   This workflow instance invokes the **AddSecondaryUser** external call to create a secondary account on the service identified in step 2 by spawning a new workflow.
9. Check the status for the request ID that you noted in step 8.

Documentation References

Use following documentation references for appropriate technical and administrative user context:

- For technical information about external calls, refer to the *HP OpenView Select Identity External Calls Guide.*
- For administrator-oriented information about external calls and workflows, refer to the *HP OpenView Select Identity Online Help for Administration.*
- For technical information about workflows and workflow templates, refer to the *HP OpenView Select Identity Online Help for Workflow Studio.*

Existing Secondary Account Validation

The `AddSecondaryUser` external call does not check or validate how many secondary accounts exist for the given user on the subscribed service. However, the external call can be modified to perform this validation.

Preventing Superfluous Account Creation in Reconciliation Workflows

Do not use any service that uses this external call in any workflow **Add-to-Service** block .

If you attach a service associated with this external call to an **Add-to-Service** block in a reconciliation workflow, an additional secondary account will be created.

To prevent additional secondary account creation in reconciliation workflows, perform the following steps:

1. Modify the external call code to reuse the `GUID` value from the original `requestTarget`.

2 Change the request event type of the secondary user creation request to `ADD-SERVICE`.

3 Pass the `UserName` attribute as a parameter to the `AddSecondaryUser` external call.

Add SecondaryUser External Call Setup Procedure

To set up the `AddSecondaryUser` external call, perform the following steps:

1 Copy the external call `jar` file to the appropriate location under the Select Identity installation directory on the Application Server (this will be a shared file system folder in the case of a cluster).

2 Register the external call using the following information:

Classname: `com.hp.ovsi.extcall.wfexternalcall.AddSecondaryUser`

Classpath: Directory path to the `.jar` file containing this external call

Call Type: `WorkflowExternalCall`

For more information about how to set up external calls, refer to the *HP OpenView Select Identity External Calls Guide.*

Keystore Sample Commands

This section provides command line output and information for:

- Creating the bootstrap keystore for the Select Identity 4.1X environment or for a migrated environment that did not previously use a custom keystore.
- Upgrading the bootstrap keystore from earlier versions (pre-4.1X) to be compatible with Select Identity 4.1X.
- Creating object migration keystores / truststores.

Creating a Bootstrap Keystore

To create a bootstrap keystore for use in Select Identity 4.1X, perform the following steps:

1 Prepare for the keystore configuration.

a Set up the Java™ is environment variables to point to a proper JDK. It is recommended that you use the same JDK that will be used by the application server. You can set up the Java environment variables manually or use the command line setup utilities from the application server running Select Identity.

For example:

In WebSphere, enter:

```
cd<WAS_HOME>/bin
. ./setupCmdLine.sh
```

In WebLogic, enter:

```
cd<WL_HOME>/weblogic81/server/bin
. ./setWLSEnv.sh
```

b Run the “Java -version” command to ensure that you are using the appropriate JDK.

c Ensure that the OVSIKeyStoreUtility files are copied to the server so they can be accessed.

2 Create the database encryption key in the keystore.

a Execute: `./genkey.sh`

You will see this output message:

This utility creates one AES secret key in a JCEKS keystore.

b Enter the full path of the store, including the store file name:

`/opt/SIInstallation/OVSIKeyStoreUtility/mykey`

You will see these output messages:

File does not exist at the specified path.

KeyStore will be created.

c Enter the store password: ********

d Enter the key alias: **myDBKey**

e Enter the key password or press **Enter** to use the store password.

f Enter the key password or press **Enter** to use the store password again.

g Select a key size from the list:
1:128
2:192
3:256

Select an option: **3**

You will see these output messages:

Engine provider: SunJCE

Starting to verify the generated key.

Verified the generated key.

Finished!

3 Create the security framework key in keystore without a key password.

a Execute: `./genkey.sh`

You will see this output message:

This utility creates one AES secret key in a JCEKS keystore.

b Enter the full path of the store, including the store file name:

`/opt/SIInstallation/OVSIKeyStoreUtility/mykey`

c Enter the store password: ********

d Enter the key alias: **mySFKey**

e Enter the key password or press **Enter** to use the store password.

f Enter the key password or press **Enter** to use the store password again.

g Select a key size from the list:
1:128
2:192

3:256

Select an option: **3**

You will see these output messages:

Engine provider: SunJCE

Starting to verify the generated key

Verified the generated key

Finished!

4 Create the bootstrap properties file.

a Execute: `./genprop.sh`

You will see this output message:

This utility creates a OVSI property file for key and trust stores.

b Specify the file type to generate:
1:OVSI bootstrap keystore
2:OVSI secure object migration keystore
3:OVSI truststore

Select an option: **1**

c Enter the full path for the property file to be saved, including the file name. If the path doesn't include the file name, the default name (**keystore.properties**) will be used:

`/opt/SIInstallation/OVSIKeyStoreUtility/mykey.properties`

You will see this output message:

The information will be stored in: ***/opt/SIInstallation/OVSIKeyStoreUtility/mykey.properties***

d Enter the full path of the store, including the store file name:

`/opt/SIInstallation/OVSIKeyStoreUtility/mykey`

e Enter the store password: ********

f Enter the store password again: ********

g Enter the store type:
1:JCEKS
2:JKS
3:nCipher.sworld

Select an option: **1**

h Enter the database encryption key alias: **myDBKey**

i Enter the key password or press **Enter** to use the store password.

j Enter the key password or press **Enter** to use the store password again.

k Enter the security encryption key alias: **mySFKey**

You will see these output messages:

Verifying the database encryption key Key verified.

Verifying the security framework encryption key Key verified.

Finished!

Upgrading the Bootstrap Keystore from Earlier Versions (pre-4.1x)

The following example assumes the old keystore was generated using the ks_gen.sh script provided in prior versions of Select Identity. You will need the old keystore password in order to add a new key.

Here is the content of the old keystore.properties file:

```
Select Identity Keystore Parameters
Fri Nov 03 16:03:59 CST 2006
si.keystore.filepath=/opt/si4.0/weblogic/keystore/40ksklp
si.keystore.storepass=WoYknWKtXCHDyzf3l4xjh7qw2lZjaZ+i64LPCAAhxjp9rjX0ArNLd
Gv0qKR6PHrYPAsMp9Z6YUjhfUvSYyyk9/A9r80qhfjiZ9XCF/
GcJ7cPFr9Gtoz6bVdcIXMxg2zLZiaRw43GFUAKlqv13bfeXA6H88W5GWzsM0kIzZDFEc
k\=
si.keystore.keypass=OoOZwWSbM9PX/
wPGmGvlyIWAVvqjibW6WK+STCZmM5ddAXsqQcZHbwGCSeUD9g5opzjq2mTXoawu/
SgIimQMRDtGr1fZaWJ42ZkZR86KkHRF8YNxLcLvaE/NXIKknonu5f/
npw8KSK25WB5qu2y6RGqwrG1WavnsEL2rmViO0gk\=
si.keystore.alias=40key
```

1. Prepare for the keystore configuration.
   a. Set up the Java environment variables to point to a proper JDK. It is recommended that you use the same JDK that will be used by the application server. You can set up the Java environment variables manually or use the command line setup utilities from the application server running Select Identity.

      For example:

      In WebSphere, enter:
      ```
      cd <WAS_HOME>/bin
      . ./setupCmdLine.sh
      ```

      In Weblogic, enter:
      ```
      cd <WL_HOME>/weblogic81/server/bin
      . ./setWLSEnv.sh
      ```

   b. Run the "Java -version" command to ensure that you are using the appropriate JDK.

   c. Ensure that the OVSIKeyStoreUtility files are copied to the server so they can be accessed.

2. Create the security framework key in keystore without a key password.
   a. Execute: `./genkey.sh`

      You will see this output message:

      This utility creates one AES secret key in a JCEKS keystore.

   b. Enter the full path of the store, including the store file name:

      `/opt/si4.0/weblogic/keystore/40ksklp`

   c. Enter the old keystore password: ********

   d. Enter the key alias: **410sfkey**

   e. Press **Enter**.

f Press **Enter** again.

g Select a key size from the list:
1:128
2:192
3:256

Select an option: **3**

You will see these output messages:

Engine provider: SunJCE

Starting to verify the generated key.

Verified the generated key.

Finished!

3 Update the old bootstrap properties file by adding these four new lines bolded at the bottom of this file. The items in red will vary depending on your key alias names used in your bootstrap keystore.

si.keystore.filepath=/opt/si4.0/weblogic/keystore/40ksklp
si.keystore.storepass=WoYknWKtXCHDyzf3l4xjh7qw2lZjaZ+i64LPCAAhxjp9rjX0ArNLdGv0qKR6PHrYPAsMp9Z6YUjhfUvSYyyk9/A9r80qhfjiZ9XCF/GcJ7cPFr9Gtoz6bVdcIXMxg2zLZiaRw43GFUAKlqv13bfeXA6H88W5GWzsM0kIzZDFEck\=
si.keystore.keypass=OoOZwWSbM9PX/wPGmGvlyIWAVvqjibW6WK+STCZmM5ddAXsqQcZHbwGCSeUD9g5opzjq2mTXoawu/SgIimQMRDtGr1fZaWJ42ZkZR86KkHRF8YNxLcLvaE/NXIKknonu5f/npw8KSK25WB5qu2y6RGqwrG1WavnsEL2rmViO0gk\=
si.keystore.alias=**40key**
si.keystore.40key.keyalg=PBEWithMD5AndTripleDES
si.keystore.storetype=JCEKS
si.keystore.keypass.alias=410sfkey
si.keystore.410sfkey.keyalg=AES/ECB/PKCS5Padding

Additional Migration Utility Information

This section contains additional information about running the migration utility for environments that are running in an Oracle RAC configuration. For the upgrade to complete successfully, the migration utility used to migrate from 4.0/4.01 to 4.1X needs to connect to one specific node in the Oracle RAC. On the machine that will run the migration utility, ensure that there is an entry in the tnsnames.ora file that has the Oracle SID of the Oracle node you want to connect directly to as the identifier for the tnsnames.ora entry.

You need to use a tnsnames.ora entry that connects to just one node in the RAC.

For example, if a node in the RAC has an Oracle SID of 'RACNODE1' and an IP of '192.168.100.123', you could use an entry like this:

RACNODE1 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.100.123)(PORT = 1521))

(CONNECT_DATA =

(SERVER = DEDICATED)

(SID = RACNODE1)

)

)

In the setUserEnv.sh script, set the ORACLE_SID=RACNODE1 and ensure the DBSERVER=192.168.100.123 and DBPORT=1521. The ORACLE_SID value must be the actual Oracle SID for the Oracle node you are connecting to and also the identifier for the tnsnames.ora file entry connecting to the node in the Oracle RAC.

To find the entry to connect with in tnsnames.ora, the portion of the upgrade using SQLPlus will connect using the userid/password and the ORACLE_SID value. For example, a SQLPlus connection is built like this: sqlplus $DB_USER/$DB_PASS@$ORACLE_SID

The portion of the upgrade that connects using JDBC will construct the URL to connect to the database. For example, the JDBC URL is constructed like this:

-Djdbc.driverClassName=oracle.jdbc.OracleDriver -Ddatabase.url=jdbc:oracle:thin:@$ DBSERVER:$DB_PORT:$ORACLE_SID -Ddatabase.user.name=$DB_USER -Ddatabase.user.password=$DB_PASS