Oracle® Fusion Middleware

Integration Guide for Oracle Enterprise Repository

11*g* Release 1 (11.1.1.4.0)

E15754-07

March 2011

ORACLE®

Oracle Fusion Middleware Integration Guide for Oracle Enterprise Repository, 11*g* Release 1 (11.1.1.4.0)

E15754-07

Primary Author: Vimmika Dinesh

Contributing Author: Aditi Dalia, Atturu Chandra Prasad Reddy, David DiFranco, Dennis Chin, Jeff Cunnigham, John Yoder, Mike Wallace, Scott Spieker, Sharon Fay, Viswanatha Basavalingappa, Manoramya Chandramohan

Contents

1 Configuring an Artifact Store

1.1 Overview 1-1
1.2 Creating and Configuring an Artifact Store 1-1
1.3 Selecting a New Artifact Store 1-3

2 ClearCase Integration

2.1 ClearCase Web Interface 2-1
2.1.1 Overview 2-1
2.1.2 Prerequisites 2-2
2.1.3 Creating and Configuring Repository and Assets 2-3
2.2 File Stores 2-6
2.2.1 Overview 2-6
2.2.2 Adding the File Stores Feature to Oracle Enterprise Repository 2-6
2.2.3 Creating a File Store 2-7
2.2.4 Configuring an Artifact Store For a File Store 2-8
2.2.5 Adding a File to an Asset Using the File Store's Artifact Store 2-8
2.2.6 Using File Stores 2-9
2.3 ClearQuest Integration 2-11
2.3.1 Adding ClearQuest 2-11
2.3.2 Configuring a ClearQuest Artifact Store 2-11
2.3.3 Adding a File to an Asset Using the ClearQuest Artifact Store 2-12

3 Harvest-HTTP Repository Host Integration

3.1 Overview 3-1
3.2 Installation 3-1
3.3 Configure an Artifact Store 3-2
3.4 Add the Harvest Artifact Store to an Asset 3-2

4 Serena ChangeMan Integration

4.1 Adding Serena Changeman Plug-ins 4-1
4.2 Configuring an Artifact Store 4-1
4.3 Configuring a PVCS Artifact Store 4-2
4.4 Add a File to an Asset Using a PVCS Repository 4-2
4.5 Security Considerations 4-3

5 Enterprise Manager Integration Utility

5.1 Overview 5-1
5.1.1 Prerequisites 5-1
5.1.2 Obtaining the Enterprise Manager Integration Utility 5-2
5.1.3 High Level Use Cases 5-2
5.2 Using the Enterprise Manager Integration Utility 5-2
5.2.1 Running from Command Line 5-3
5.2.2 Scheduling from Enterprise Manager 5-4
5.2.3 Metric Publishing 5-4
5.2.3.1 Oracle Service Bus 5-4
5.2.3.2 BPEL PM 5-5
5.2.3.3 Web Services 5-5
5.2.3.4 Artifacts Creation 5-5
5.2.3.5 Endpoint Creation 5-5
5.2.3.6 Metrics to Update 5-5
5.3 Configuring the Enterprise Manager Integration Utility 5-6
5.3.1 Setting the Repository and Enterprise Manager Connection Information for the Command-line Utility 5-6
5.3.2 Advanced Configuration 5-8
5.3.2.1 Metric Mappings 5-8
5.3.2.2 Target Finders 5-9
5.3.2.3 Logging 5-9
5.4 Encrypting the Configuration File Passwords 5-9
5.5 Known Issues 5-10
5.5.1 Using Incorrect Encrypted Password 5-10

6 Integration with Amberpoint

7 Integration with Development Environments

7.1 Overview 7-1
7.2 Best Practices 7-1
7.2.1 Asset Production Process 7-1
7.2.1.1 Policies 7-2
7.2.1.2 Propose/Submit Assets 7-3
7.2.2 Asset Consumption Process 7-3
7.2.2.1 Prescription Reuse 7-3
7.2.2.2 Developer-Driven Discovery 7-5
7.2.2.3 Automated Usage Detection 7-5
7.3 High Level Use Cases 7-5
7.3.1 Submit Files 7-6
7.3.2 Harvest Files 7-6
7.3.3 Search Oracle Enterprise Repository 7-6
7.3.4 View Asset Details 7-6
7.3.5 Download Artifacts 7-6
7.3.6 Prescriptive Reuse 7-6
7.3.7 Automatic Usage Detection 7-6

8 Configuring Oracle Enterprise Repository to Support Integration with Your IDE

8.1 Install the Harvester 8-1
8.2 Assign IDE Users to Oracle Enterprise Repository Projects 8-1
8.3 Establish Compliance Templates 8-1
8.4 Set up Automatic Usage Detection 8-2

9 Configuring Your IDE to Support Integration with Oracle Enterprise Repository

9.1 Configuring Oracle JDeveloper 9-1
9.1.1 Integrating with Oracle JDeveloper 11*g* R1 Patchset Releases 9-1
9.1.2 Integrating with Oracle JDeveloper 11*g* R1 9-7
9.1.3 Integrating with Oracle JDeveloper 10*g* 9-7
9.2 Configuring Eclipse 9-8
9.2.1 Enable Harvesting in Eclipse 9-8
9.2.1.1 Setting up Eclipse Environment to use Harvester as an "External Program" 9-8
9.2.1.2 Harvesting in Eclipse Environment using "External Program" 9-10
9.2.1.3 Setting up Eclipse Environment to use Harvester via ANT 9-11
9.2.1.4 Harvesting in Eclipse Environment using ANT 9-12
9.2.2 Configure the Oracle Enterprise Repository Plug-ins 9-12
9.2.2.1 Configuring the Oracle Enterprise Repository Plug-ins for Repository Access 9-13
9.2.2.2 Prerequisites for Using the Oracle Enterprise Repository Plug-ins for Eclipse 9-13
9.2.2.2.1 Assign Users to an Oracle Enterprise Repository Project 9-13
9.2.2.2.2 Enabling the Assets-in-Progress Properties 9-14
9.2.2.2.3 SiteMinder 9-14
9.2.2.2.4 Java JDK 9-14
9.2.2.2.5 XML Parsing 9-14
9.2.3 Configure the Oracle Enterprise Repository Preferences 9-15
9.2.4 Enable Automatic Usage Detection 9-16
9.3 Configuring VS .NET 9-18
9.3.1 Enable Harvesting in VS .NET 9-18
9.3.2 Configure the Oracle Enterprise Repository Plug-ins 9-19
9.3.2.1 Prerequisities 9-20
9.3.2.2 Installation 9-20
9.3.3 Configure the Connection to Oracle Enterprise Repository 9-21
9.3.4 Assign an Oracle Enterprise Repository Project to a .NET Solution 9-22
9.3.5 Enable Automatic Usage Detection 9-23
9.3.5.1 Overview of SFID 9-23
9.3.5.2 Configuring Automatic Usage Detection 9-23

10 Using the IDE to Interact with Oracle Enterprise Repository

10.1 Using Oracle JDeveloper 10-1
10.1.1 Harvest Artifacts 10-2
10.1.2 Search Oracle Enterprise Repository 10-2

10.1.3 View Asset Details 10-2
10.1.4 Download Artifacts 10-3
10.1.4.1 Associating JDeveloper Application with Oracle Enterprise Repository 10-3
10.1.4.2 Consuming WSDL/Service from Oracle Enterprise Repository 10-4
10.1.5 Prescriptive Reuse 10-6
10.2 Using Eclipse 10-6
10.2.1 Submit Files 10-6
10.2.2 Harvest Artifacts 10-7
10.2.3 Search Oracle Enterprise Repository 10-7
10.2.4 View Asset Details 10-7
10.2.5 Download Artifacts 10-9
10.2.6 Prescriptive Reuse 10-10
10.2.7 Automatic Usage Detection 10-10
10.3 Using VS .NET 10-11
10.3.1 Submit Files 10-12
10.3.2 Harvest Artifacts 10-12
10.3.3 Search Oracle Enterprise Repository 10-12
10.3.4 View Asset Details 10-13
10.3.4.1 Accessing the Repository Assets Pane 10-14
10.3.4.1.1 About Oracle Enterprise Repository Projects 10-14
10.3.4.1.2 Accessing the Repository Assets View 10-14
10.3.4.2 Accessing the Oracle Enterprise Repository Log 10-14
10.3.5 Download Artifacts 10-15
10.3.6 Automatic Usage Detection 10-16

11 Integration with SAP

11.1 Overview 11-1
11.2 Prerequisites 11-2
11.3 Configuring Integration with SAP 11-2
11.3.1 Aligning the Taxonomy Between SAP and Oracle Enterprise Repository Assets. 11-3
11.3.2 Exporting the Selected SAP Service Assets from SAP Enterprise Service Registry to OER 11-3
11.3.3 Receiving New Services from SAP 11-4
11.3.4 Configuring Ongoing Updates from SAP 11-4
11.3.5 End User Visibility of Available SAP Services 11-4

12 Repository Extensibility Framework

12.1 Introduction to REX 12-1
12.2 REX Architecture 12-2
12.2.1 Subsystems Overview 12-3
12.2.2 CRUD-Q Naming Convention 12-3
12.2.2.1 Atomicity of Method Calls 12-4
12.2.2.2 No Inter-call Transaction Support 12-4
12.2.3 Fundamental WSDL Data Types 12-4
12.2.4 Versioning Considerations for the Oracle Enterprise Repository REX 12-5
12.3 Basic Concepts 12-6
12.3.1 Getting Started - Enabling the OpenAPI within the Oracle Enterprise Repository 12-6

12.3.2 Getting Started - Consuming the WSDL 12-6

13 ArtifactStore API

13.1 Overview 13-1
13.2 Use Cases 13-1
13.2.1 Use Case: Create missing ArtifactStore 13-1

14 AcceptableValueLists API

14.1 Overview 14-1
14.2 Use Cases 14-1
14.2.1 Use Case: Create and Edit an Acceptable Value List 14-1
14.2.2 Use Case: Find an Acceptable Value List and use it in an asset 14-3

15 Asset API

15.1 Overview 15-1
15.1.1 Definitions 15-3
15.1.2 Sample Code 15-3
15.1.3 Related Subsystems 15-6
15.2 Use Cases 15-7
15.2.1 Use Case: Creating a new asset 15-8
15.2.2 Use Case: Creating a new asset from XML 15-9
15.2.3 Use Case: Modifying an asset 15-11
15.2.4 Use Case: Assign users to an asset 15-13
15.2.5 Use Case: Building an asset search 15-15
15.2.6 Use Case: Upgrading asset status 15-18
15.2.7 Use Case: Downgrading asset status 15-20
15.2.8 Use Case: Apply and remove Compliance Templates from a project 15-21
15.2.9 Use Case: Creating the new version of an asset and retiring the old version 15-23
15.2.10 Use Case: "Housekeeping" 15-25
15.2.11 Use Case: Finding assets and updating custom-data 15-28
15.2.12 Use Case: Reading an Asset's Tabs 15-29
15.2.13 Use Case: Retrieve An Asset's Tab Based on TabType 15-30
15.2.14 Use Case: Approving and Unapproving a tab 15-31

16 AssetType API

16.1 Overview 16-1
16.2 Use Cases 16-2
16.2.1 Use Case: Create and edit a new Type 16-2
16.2.2 Use Case: Create a Compliance Template Type 16-3
16.2.3 Use Case: Find Types 16-4
16.2.4 Use Case: Read tab types 16-6
16.2.5 Use Case: Retrieve all Asset Type tabs 16-7

17 Categorization Types and Categorizations API

17.1 Overview 17-1

17.2 Use Cases ... 17-2
17.2.1 Use Case: Create a Categorization Type ... 17-2
17.2.2 Use Case: Manipulate Categorization Types ... 17-4
17.2.3 Use Case: Manipulate Categorizations ... 17-7

18 CMF Entry Type API

18.1 Overview ... 18-1
18.2 Use Cases ... 18-1
18.2.1 Use Case: Manipulating CMF Entry Types ... 18-1

19 Custom Access Settings API

19.1 Overview ... 19-1
19.2 Use Cases ... 19-2
19.2.1 Use Case: Retrieve a List of Custom Access Setting Types ... 19-2
19.2.2 Use Case: Get Default Custom Access Setting Names ... 19-3

20 Department API

20.1 Overview ... 20-1
20.2 Use Cases ... 20-1
20.2.1 Use Case: Manipulate Departments ... 20-1

21 Extraction API

21.1 Overview ... 21-1
21.2 Use Cases ... 21-2
21.2.1 Use Case: Extract an Asset ... 21-2
21.2.2 Use Case: Read an Extraction ... 21-5
21.2.3 Use Case: Update an Extraction ... 21-6

22 Localization of REX Clients

22.1 Overview ... 22-1
22.2 Use Cases ... 22-2
22.2.1 Use Case: Creating localized messages from REX Exceptions ... 22-2
22.2.2 Use Case: Creating localized messages from REX Audit Messages ... 22-3

23 Notification API

23.1 Overview ... 23-1
23.2 Use Cases ... 23-1
23.2.1 Use Case: Read Notification Substitution List ... 23-1
23.2.2 Use Case: Create a Notification ... 23-1

24 Policy API

24.1 Overview ... 24-1
24.2 Use Cases ... 24-2
24.2.1 Use Case: Create a Policy ... 24-2

24.2.2 Use Case: Get All Policies 24-4
24.2.3 Use Case: Get/Set Policy Assertions 24-5
24.2.4 Use Case: Get Policies That Have Been Applied To An Asset 24-7
24.2.5 Use Case: Set Which Policies Are Applied To An Asset 24-8
24.2.6 Use Case: Evaluate Asset Compliance 24-9

25 Projects API

25.1 Overview 25-1
25.2 Use Cases 25-1
25.2.1 Use Case: Create a New Project 25-2
25.2.2 Use Case: Read a Project 25-3
25.2.3 Use Case: Validate a Project 25-4
25.2.4 Use Case: Update a Project 25-6
25.2.5 Use Case: Update a Project's Produced Assets 25-8
25.2.6 Use Case: Remove Produced Assets from a Project 25-9
25.2.7 Use Case: Update a Project's Asset Usage 25-11
25.2.8 Use Case: Closing a Project with Hidden Assets 25-13
25.2.9 Use Case: Add Users and Related Projects to a Project 25-15
25.2.10 Use Case: Remove Related Projects and Users from a Project 25-19
25.2.11 Use Case: Update a Project's Extractions - Reassign Extractions to a Different User on the Same or a Different Project 25-23
25.2.12 Use Case: Update a Project's User - Reassign User and His/Her Extractions to Another Project 25-25
25.2.13 Use Case: Update a Project's User - Reassign User Only (Not the User's Extractions) to Another Project 25-27
25.2.14 Use Case: Read the Value-Provided for a Project and Asset 25-31
25.2.15 Use Case: Update the Value Provided for a Project and Asset - Use Predicted Value 25-33
25.2.16 Use Case: Update the Value Provided for a Project and Asset - Use Consumer Value 25-36
25.2.17 Use Case: Update the Value-Provided for a Project and Asset - Use Project Lead Value 25-39

26 Relationship Types API

26.1 Overview 26-1
26.2 Use Cases 26-1
26.2.1 Use Case: Create a new relationship type 26-1
26.2.2 Use Case: Modify related assets 26-3
26.2.3 Use Case: Query related assets 26-4

27 Role API

27.1 Overview 27-1
27.2 Use Cases 27-1
27.2.1 Use Case: Manipulate Roles 27-1

28 Subscriptions API

28.1 Overview 28-1
28.2 Use Cases 28-1
28.2.1 Use Case: Create Subscription to Assets 28-1
28.2.2 Use Case: Delete Subscription to Assets 28-3
28.2.3 Use Case: Read Subscriptions for Assets 28-4
28.2.4 Use Case: Read Users Subscribed to an Asset 28-5

29 System Settings API

29.1 Overview 29-1
29.2 Use Cases 29-1
29.2.1 Use Case: Query for System Settings 29-1

30 User API

30.1 Overview 30-1
30.2 Use Cases 30-1
30.2.1 Use Case: Manipulating Users 30-1

31 Vendor API

31.1 Overview 31-1
31.2 Use Cases 31-1
31.2.1 Use Case: Manipulating Vendors 31-1

Index

Preface

Oracle Fusion Middleware Integration Guide for Oracle Enterprise Repository describes how to configure Enterprise Manager, Oracle JDeveloper, VS .NET with Oracle Enterprise Repository. This guide also describes the Oracle Enterprise Repository connectors.

Audience

This document is intended for all Oracle Enterprise Repository users who want configure the development environments to easily produce or consume files from Oracle Enterprise Repository. This document is also intended for all Oracle Enterprise Repository users who want to use REX and the REX APIs.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Enterprise Repository 11*g* Release 1 (11.1.1.4.0) documentation set:

- Oracle Enterprise Repository on OTN - The home page for Oracle Enterprise Repository on Oracle Technology Network (OTN) is:

  `http://www.oracle.com/technologies/soa/enterprise-repository.html`

- Architect Center: SOA Governance: Essential to Your Business - Learn how effective SOA governance is an essential element in any enterprise transformation strategy by reading the Architect Center: SOA Governance: Essential to Your Business documents at:

  `http://www.oracle.com/technology/architect/soa/soagov/index.html`

- SOA Blog - Keep on top of the latest SOA blogs at:

  `http://blogs.oracle.com/governance`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Oracle Enterprise Repository Connectors

This part describes how to get started with Oracle Enterprise Repository connectors.

This part contains the following chapters:

- Chapter 2, "ClearCase Integration"
- Chapter 3, "Harvest-HTTP Repository Host Integration"
- Chapter 4, "Serena ChangeMan Integration"

1 Configuring an Artifact Store

This chapter provides an overview of artifact stores and describes how to create and configure an artifact store.

This chapter contains the following sections:

- Section 1.1, "Overview"
- Section 1.2, "Creating and Configuring an Artifact Store"
- Section 1.3, "Selecting a New Artifact Store"

1.1 Overview

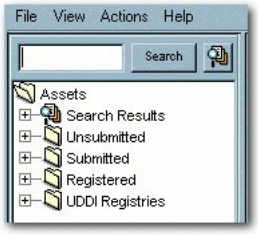
An Artifact Store is where the files relevant to assets in Oracle Enterprise Repository are stored. Launch the Asset Editor window from the Oracle Enterprise Repository home page.

1.2 Creating and Configuring an Artifact Store

This section describes the the steps to create and configure an artifact store.

1. Click **Edit/Manage Assets**. The Asset Editor is displayed, as shown in Figure 1–1.

Figure 1–1 Asset Editor

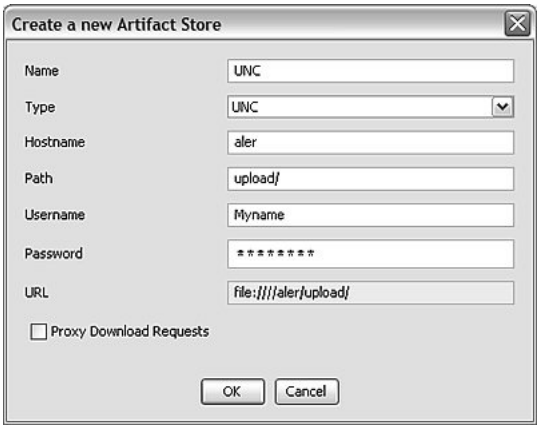


2. Select **Configure Artifact Stores** in the Actions menu. The Configure Artifact Stores dialog is displayed, as shown in Figure 1–2.

Figure 1–2 Configure Artifact Stores Dialog

3. Click **Add**. The Create a New Artifact Store dialog is displayed, as shown in Figure 1–3.

Figure 1–3 Create a New Artifact Store Dialog



4. Enter a unique name for the artifact store in the Name text box.
5. Select the type of artifact store from the Type list (this example uses UNC).
6. Enter the host name of the application server in the Hostname box.
7. Enter the rest of the path in the Path text box.

   A file link, appended with the host name and path, appears in the URL text box. This link can be cut/pasted into a file explorer/browser window in order to view the file.
8. If necessary, enter the appropriate information in the Username and Password text boxes.
9. Click **OK**.
10. Enable the artifact store using Submission Upload Artifact Store system setting, as described in Section 1.3, "Selecting a New Artifact Store".

1.3 Selecting a New Artifact Store

After configuring an artifact store as described in Creating and Configuring an Artifact Store, you must select it using the Submission Upload Artifact Store system setting on the Oracle Enterprise Repository Admin page.

1. Open the Oracle Enterprise Repository Admin page.
2. In the left panel, click **System Settings**.
3. Locate the Upload Area section in the Server Settings group of system settings, as shown in Figure 1–3.
4. Use the Submission Upload Artifact Store list to select the newly created artifact store.
5. Click **Save**.

2

ClearCase Integration

This chapter describes how to set up a ClearCase repository and integrate it with Oracle Enterprise Repository.

This chapter contains the following sections:

- Section 2.1, "ClearCase Web Interface"
- Section 2.2, "File Stores"
- Section 2.3, "ClearQuest Integration"

2.1 ClearCase Web Interface

This section explains the procedure to follow to setup a ClearCase repository, which allows ClearCase files to be linked to assets for future use/download. This section contains the following topics:

- Section 2.1.1, "Overview"
- Section 2.1.2, "Prerequisites"
- Section 2.1.3, "Creating and Configuring Repository and Assets"

2.1.1 Overview

This section contains the following topics:

- "Webshpere 5.x Apache Plug-In"
- "WebLogic 8.1 Tuxedo Plug-In"

Webshpere 5.x Apache Plug-In

When using an HTTP server (such as Apache, IIS, IBM HTTP Server) to connect to a Websphere 5.x server using the `mod_was_ap20_http.so` or `mod_was_ap20_http.dll` plug-in, a configuration change must be applied to the `plugin-cfg.xml` document used with this connector:

Each time the `crtplugininst` application is run to regenerate the `plugin-cfg.xml` document for use on the HTTP server(s) the Config element contains a value of AcceptAllContent=`false`, by default. This parameter must be changed to true to allow deltaV requests to be passed between the HTTP Server and the Websphere application server hosting the Oracle Enterprise Repository. This restriction only applies to an HTTP server using the Websphere plug-in to connect the two servers.

WebLogic 8.1 Tuxedo Plug-In

When using the Weblogic Tuxedo Plugin, there is a requirement of 8.1 SP3 being applied to both the application server as well as the Tuxedo Plug-In on the Apache server.

2.1.2 Prerequisites

Before using ClearCase, you must perform the following prerequisites:

- The application server must support the UTF-8 character set to allow ClearCase and Oracle Enterprise Repository to function properly together.
- Ensure the application server has access to the ClearCase server.
- CCWeb and/or ClearTool must be installed and enabled on the application server computer. For more information, see the ClearCase documentation.

Enabling UTF-8 Support

Enabling the UTF-8 character set is accomplished in the following manners based on the server employed.

- **Weblogic 7.x/8.x**

  You may specify the character set for all deployed Weblogic Web applications deployed on a Weblogic Server instance by setting the system properties client.encoding.override and file.encoding equal to the name of the character set. Set this system property in the environment variable called `JAVA_OPTIONS`, for example, `JAVA_OPTIONS=-Dclient.encoding.override=UTF-8 -Dfile.encoding=UTF-8`.

  These values can also be supplied as part of the startup options for the domain.

- **Websphere 5.x**

  Change the Generic JVM Arguments for the server to include the following parameter: `-Dclient.encoding.override=UTF-8`

- **Tomcat 5.0.25**

  Change the `URIEncoding` value in the Connector element within the `server.xml` file in the *CATALINA_HOME/conf* directory:

```
<Connector port="8080" URIEncoding="UTF-8" ...
```

Note: The ability to browse into the ClearCase server and view files/directory structure from within the Oracle Enterprise Repository application is provided through Files Stores integration.

Important Notes

1. Construct a view in CCWeb.
2. Create the link based on that view constructed in CCWeb.
3. Add the link within the File Information section of the asset within the Asset Editor.
4. Select the **Test** button to verify that the link is valid.

5. It should also be possible to access the link by pasting the URL into a browser address window. If this is not possible the link itself may be in error, or there may be a problem with the network connection to the ClearCase server.

2.1.3 Creating and Configuring Repository and Assets

You can create and configure repository and assets in Oracle Enterprise Repository. This section contains the following topics:

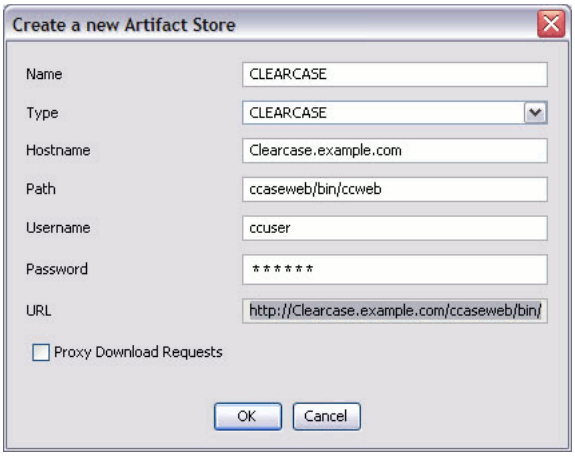
- "Configure an Artifact Store"
- "Set the artifact store from which to extract files"
- "Create an asset for the ClearCase file(s)"
- "Link the ClearCase file to the asset"
- "Extract the asset and the ClearCase file(s)"

Configure an Artifact Store

This procedure is performed in the Oracle Enterprise Repository Asset Editor screen.

1. Open the **Actions** menu.
2. Click **Configure Artifact Store**.
3. Click **Add**. The Create a New Artifact Store dialog is displayed, as shown in Figure 2–1.
4. Fill in the appropriate information, as shown in Figure 2–1.

Figure 2–1 Create a New Artifact Store Dialog



5. Click **OK**.

Set the artifact store from which to extract files

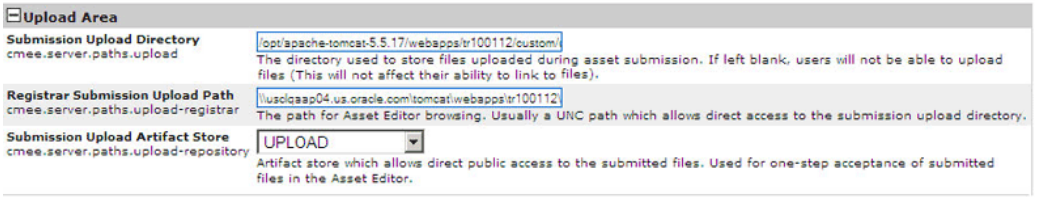
This procedure is performed on the Oracle Enterprise Repository Admin screen.

1. Click **System Settings**.
2. Enter `cmee.server.paths.upload-repository` in the Search box, as shown in Figure 2–2.

Figure 2–2 Search Box

The Upload Area section within the Server Settings section is displayed, as shown in Figure 2–3.

Figure 2–3 Upload Area Section



3. Use the Submission Upload Artifact Store list to select the ClearCase repository.
4. Click **Save** when finished.

Create an asset for the ClearCase file(s)

1. Click **Submit an Asset** in the Oracle Enterprise Repository Assets screen.

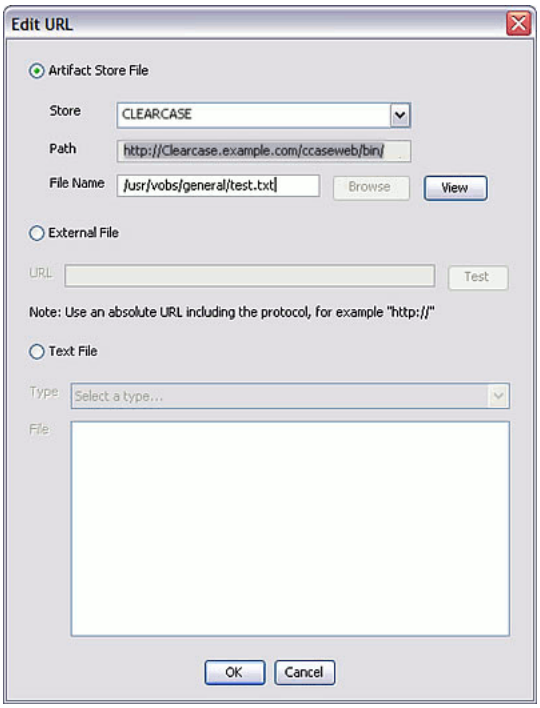
   New assets may also be created through the File menu in the Asset Editor.
2. Select an Asset Type from the drop-down menu.
3. Enter a name for the new asset in the Name text box.
4. Enter a brief description in the Description field.

> **Note:** The asset detail does not appear on Oracle Enterprise Repository Assets screen until the registration process is completed.

Link the ClearCase file to the asset

1. Open the asset in the Asset Editor.
2. Locate the File Information section (typically on the General tab).
3. Click **Add**.
4. Create a name and/or description.
5. Click **Edit**.
6. Select the Repository File radio button.

Figure 2–4 Edit URL Dialog

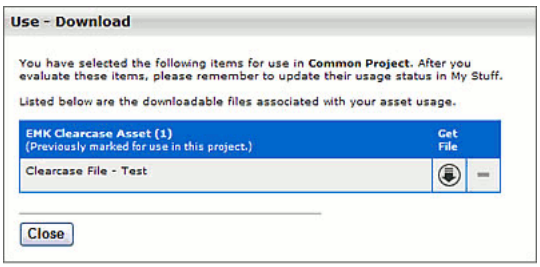


7. Select **ClearCase** in the Host list.
   - The path should populate with information configured in artifact store section:
     - *http://clearcase.example.com/ccaseweb/bin/ccweb/test.txt?dir=//usr/vobs/geneva&elem=test.txt&cmd=view&user=ccuser&password=<password>*
8. Populate the file name field with the path of the view from CCWeb and the file name.
9. Click **View** to test the file.

Extract the asset and the ClearCase file(s)

1. In the Oracle Enterprise Repository Assets screen, use **Search** to locate the newly created asset.
2. Click the asset to open its Asset Detail Display.
3. Click **Use/Download**. The Use - Download page is displayed, as shown in Figure 2–5.

Figure 2–5 Use - Download Page



ClearCase files should be available for download along with the asset.

2.2 File Stores

File Stores allow Oracle Enterprise Repository to integrate with underlying proprietary repositories. File Stores allow integration with Rational ClearCase.

This section contains the following topics:

- Section 2.2.1, "Overview"
- Section 2.2.2, "Adding the File Stores Feature to Oracle Enterprise Repository"
- Section 2.2.3, "Creating a File Store"
- Section 2.2.4, "Configuring an Artifact Store For a File Store"
- Section 2.2.5, "Adding a File to an Asset Using the File Store's Artifact Store"
- Section 2.2.6, "Using File Stores"

2.2.1 Overview

The File Store integration with Rational ClearCase allows an asset registrar to browse the ClearCase Repository. The registrar can select a specific branch and version of a file to be used when the asset is extracted. In addition, the config spec for a file can be made available for use with WSAD or XDE.

The Rational ClearCase client must be installed and configured on the application server to use File Stores with Rational ClearCase. All connections to ClearCase use the ClearTool application in the ClearCase client and share a common set of ClearCase authentication credentials.

2.2.2 Adding the File Stores Feature to Oracle Enterprise Repository

To add the file stores feature to Oracle Enterprise Repository, perform the following steps:

1. Download the Oracle Enterprise Repository installation package from the Oracle download website.
2. Unzip the downloaded file to a temporary directory.
3. Download the SQL scripts from the following location to a temporary directory:

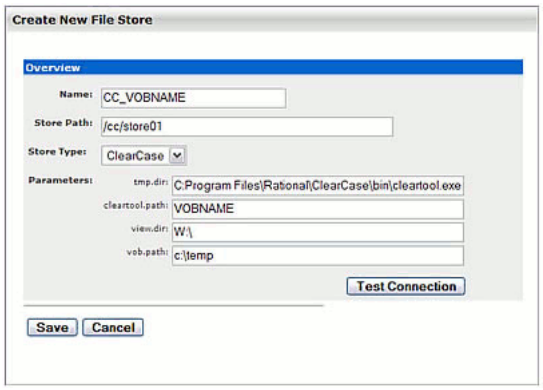
   https://support.oracle.com/oip/faces/secure/km/DownloadAttachment.jspx?attachid=825973.1:SCMIntegration
4. Using a SQL tool appropriate for your database, run the SQL script located in the temporary directory to add the File Store Artifact Store to your Oracle Enterprise Repository database.
5. Click the **Admin** link on the Oracle Enterprise Repository menu bar.
6. On the Admin screen, click **System Settings**.
7. Enable the property `registry.advanced.filestores.enabled`. A new section called File Store appears, to which the application automatically navigates.
8. Set **Advanced Access File Stores** to `true`.
9. Click **Save**.

10. Refresh the Admin screen to make the File Store section appear in the list on the left, before Basic Access Settings.

2.2.3 Creating a File Store

1. Install the Rational ClearCase client on the application server hosting Oracle Enterprise Repository.
2. Locate the **cleartool.exe** file on the application server. The entire file path to **cleartool.exe** is necessary. It is used with a File Store parameter called **cleartool.path**.
3. In the ClearCase client on the application server, create a view to be used by Oracle Enterprise Repository. A recommended practice is to include the word Flashline in the name of the view. The access path for the view is used with a File Store parameter called view.dir.
4. Mount all desired ClearCase VOBs to the created view. Each VOB requires the creation of a different File Store. The name of the VOB is the beginning of a File Store parameter called **vob.path**.
5. Locate a temporary directory on the application server. The full path to the temporary directory is used with a File Store parameter called `tmp.dir`.
6. Click the **Admin** link in the Oracle Enterprise Repository menu bar.
7. In the Admin screen, locate the **File Stores** section.
8. Click **Create New** to create a new File Store. The Create New File Store dialog is displayed, as shown in Figure 2–6.

Figure 2–6 Create New File Store Dialog



9. Populate the fields with the following parameters:
   - **Name**: specify a representative name for the ClearCase VOB referenced by the File Store. The CC prefix is recommended to indicate that the File Store is of type ClearCase. Recommended protocol for the File Store name is CC_ VOBNAME.
   - **Store Path**: enter /cc/store01 for the first ClearCase File Store, /cc/store02 for the second ClearCase File Store, and so on. The Store Path field is a symbolic path in Oracle Enterprise Repository. All File Stores use a common URL for file extraction. The Store Path appears in the URL, indicating which File Store hosts the integration to the actual content. The Store Path must be unique across all File Stores. The construction /cc/store01 is recommended.

- **Store Type**: select ClearCase from the list.
- **cleartool.path**: enter the entire file path to the cleartool.exe file on the application server.
- **vob.path**: enter the name of the ClearCase VOB mounted in the view on the application server.
- **view.dir**: enter the access path for the ClearCase view on the application server.
- **tmp.dir**: enter the temporary directory on the application server.

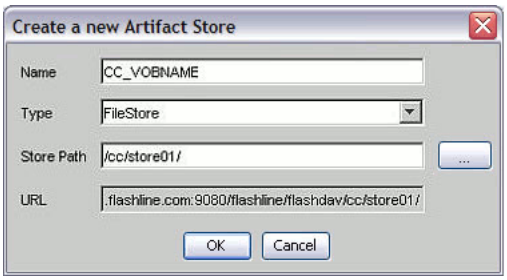
10. Click the **Test Connection** button to test the connection to the ClearCase client. If the connection is properly configured, then the message Test Succeeded appears.
11. Click the **Save** button to save the File Store for use with Artifact Stores.

2.2.4 Configuring an Artifact Store For a File Store

This procedure is performed in the Asset Editor screen.

1. Select **Configure Artifact Stores** on the Actions menu. The Configure Artifact Stores dialog is displayed.
2. Click **Add**. The Create a New Artifact Store dialog is displayed, as shown in Figure 2–7.

Figure 2–7 Create a New Artifact Store Dialog



3. Enter a name for the artifact store. `CC_VOBNAME` is the recommended protocol.
4. In the Type list, select **FileStore** as the Artifact Store type.
   - FileStore accesses the list of File Stores created on the Oracle Enterprise Repository Admin screen.
5. Enter a Store Path by clicking on the Elipses button (next to the Store Path field) and selecting the name of the File Store to be used by this Artifact Store.
6. When finished, click **OK**. A separate Artifact Store must be created for each File Store.

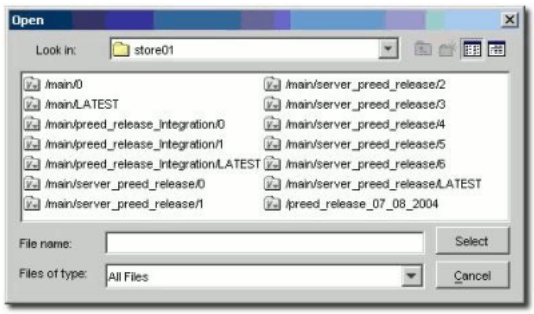
2.2.5 Adding a File to an Asset Using the File Store's Artifact Store

This procedure is performed in the Asset Editor.

1. Select the asset to which the file is to be added.
2. Click the **Add** button in the File Information section on the Overview tab.
3. In the dialog enter a name (and description, if necessary).
4. Click **Edit**. The Edit URL dialog is displayed.

5. Click the **Artifact Store File** option.
6. From the Store list, select the repository for the File Store (CC_VOBNAME).
7. Next to the File Name box, browse the ClearCase repository.
8. When browsing, the top-level branches in the VOB. Select a branch. The branchname format is Branchname/Version. The version LATEST refers to the information that is currently checked in. Generally the highest number before LATEST is the desired version. In the image below, */main/server_preed_release/6* is version six of the branch */main/server_preed_release*. The name in the folder area (store01) refers to the Store Path for the designated File Store.

Figure 2–8 Open Dialog



9. The second level displays the folders within the selected Branchname/Version pair.
10. The third level displays the versions of the selected folder.
11. Subsequently, every selected folder is followed by a desired version. The last two browsed levels are:
    - The selected filename
    - The version of that file
12. The Select button populates the File Name field on the Edit URL window.
13. Click **View** to test the URL.

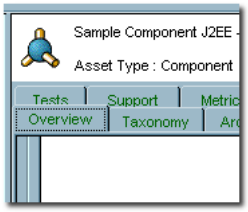
2.2.6 Using File Stores

A File Store allows the user to see all versions of all files contained in the store. At this time File Stores work only with ClearCase.

To view a file in a store, perform the following steps in the Asset Editor screen:

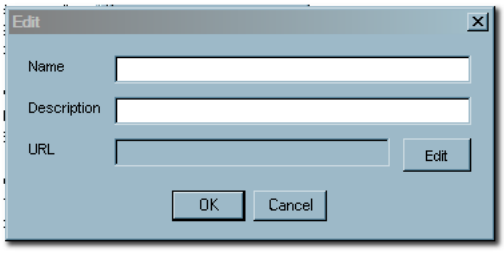
1. Select an asset.
2. Click the **Overview** tab.

Figure 2–9 Overview Tab



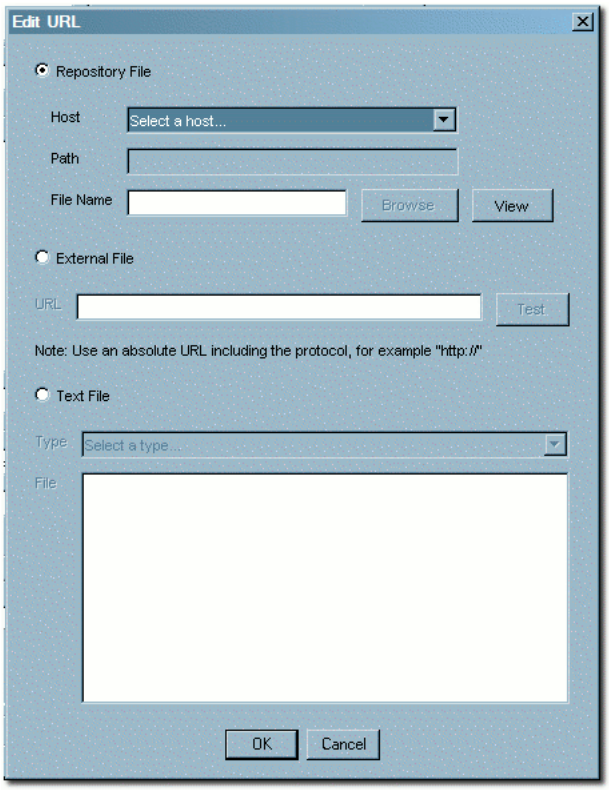
3. Click **Add** in the File Information section. The Edit dialog is displayed, as shown in Figure 2–10.

Figure 2–10 Edit Dialog



4. Click the **Edit** button. The Edit URL dialog is displayed, as shown in Figure 2–11.

Figure 2–11 Edit URL Dialog



5. In the Edit URL dialog, select **Repository File**.
6. Select a file store from the Host list.
7. Click **Browse** to locate a file, or enter the filename in the File Name text box.
8. Click **View** to view the file.
9. Click **OK** when finished.

2.3 ClearQuest Integration

Integrating ClearQuest with your system enables you to use a ClearQuest store in Oracle Enterprise Repository. Typically, an URL used to reach a file in ClearQuest resembles the following:

```
http://server.host.com:port/clearcasePath/fileName?dir=vobStructure&elem=fileName&
cmd=view&user=username&password=<password>
```

This section contains the following topics:

- Section 2.3.1, "Adding ClearQuest"
- Section 2.3.2, "Configuring a ClearQuest Artifact Store"
- Section 2.3.3, "Adding a File to an Asset Using the ClearQuest Artifact Store"

2.3.1 Adding ClearQuest

To add ClearQuest to your instance of Oracle Enterprise Repository:

1. Download the Oracle Enterprise Repository installation package (**clearquest.zip**) from the Oracle download site.
2. Unzip the download file to a temporary directory.
3. Using a SQL tool appropriate for your database, run the SQL script located in the temporary directory to add ClearQuest to your Oracle Enterprise Repository database.
4. Restart the application server.

2.3.2 Configuring a ClearQuest Artifact Store

To configure a ClearQuest artifact store in Oracle Enterprise Repository:

1. Click the **Assets** link in the Oracle Enterprise Repository menu bar. The Oracle Enterprise Repository Assets page is displayed.
2. Click **Edit/Manage Assets** to launch the Asset Editor.
3. Open the **Actions** menu.
4. Click **Configure Artifact Stores**. The Configure Artifact Stores dialog is displayed.
5. Click **Add**.
6. On the Create a new Artifact Store screen, enter a name for the artifact store file.
   - (Recommended: **ClearQuest**.)
7. In the Type list, select **ClearQuest**.
   - Accesses the ClearQuest web interface.
8. Enter a host name for the server.
9. For the Path, enter: `logon/url/default.asp`.
10. (Optional) Enter a username.
11. (Optional) Enter a password.
12. When finished, click **OK**.

2.3.3 Adding a File to an Asset Using the ClearQuest Artifact Store

To add a file to an asset using a ClearQuest artifact store:

1. Launch the Asset Editor.
2. Select the appropriate asset.
3. Navigate to the File Information section on the asset's Overview tab.
4. Click **Add**.
5. In the dialog, enter a name and description, if necessary.
6. Click **Edit**. The Edit URL dialog is displayed.
7. Select the **Artifact Store File** option.
8. From the Store list, select **ClearQuest Repository**.
9. Click the **Browse** button (next to the File Name box) to edit a ClearQuest shortcut.
10. Click the button to Launch ClearQuest Web interface and create a shortcut to a ClearQuest resource.
11. In the ClearQuest Web interface use the Operation -> Create a Shortcut link to run the wizard to generate a shortcut in ClearQuest.
12. Copy and paste everything following the question mark (?) in the resulting shortcut to the Launch ClearQuest Web shortcut screen.
13. Click **OK** to populate the File Name with the shortcut portion.
14. Click **View** to test the URL.
    - This should open the ClearQuest interface to the resource to which the shortcut was assigned.

3

Harvest-HTTP Repository Host Integration

This chapter describes how to set up a Harvest-HTTP repository and integrate it with Oracle Enterprise Repository.

This chapter contains the following sections:

- Section 3.1, "Overview"
- Section 3.2, "Installation"
- Section 3.3, "Configure an Artifact Store"
- Section 3.4, "Add the Harvest Artifact Store to an Asset"

3.1 Overview

The Harvest-HTTP Repository Integration is a single servlet Web application that allows the addition of files residing in a Harvest Repository to assets within Oracle Enterprise Repository. The application handles the physical retrieval of a file within a Harvest Repository based on an URL. The servlet name is `retrieve`.

3.2 Installation

1. Deploy the Harvest-HTTP Repository Host Integration on the application server.
2. Modify and edit the `<app_server>/harvest_rep/WEB-INF/classes/harvest.properties` file.
   - **enterprise.repositories.harvest.hco.path**
     - Local path to the Harvest **hco.exe** file (for example, `\C:\\Program Files\\CA\\AllFusion Harvest Change Manager\\hco.exe`)

     Note: Unix/Linux paths use forward slashes (/) and Windows paths use double backslashes (\\).

   - **enterprise.repositories.harvest.tmp.path**
     - Local path to where files are temporarily stored.
   - **enterprise.repositories.harvest.use-single-sign-on**=`false`

   Leave the rest of the properties as they are by default.
3. Save changes and close.
4. Restart application server.

3.3 Configure an Artifact Store

This procedure is performed in the Oracle Enterprise Repository Asset Editor.

1. Select **Configure Artifact Stores** from the Actions menu.
2. Click **Add** to open the Create a new Artifact Store dialog.
3. Create a new http repository using the following parameters:
   - **Hostname**: `<app server name with port>` where Harvest-HTTP Artifact Store Integration was installed.
   - **Path**: *harvest_rep/retrieve/<harvest_broker_name>*
   - Leave Username and Password blank.
4. Click **OK**.

3.4 Add the Harvest Artifact Store to an Asset

This procedure is performed in the Oracle Enterprise Repository Asset Editor.

1. Use Search or other means to locate the asset to which the files are to be added.
2. Open the asset.
3. Add the file:
   - Click **Repository File Selection**.
   - Select the Artifact Store corresponding to Harvest.
   - For File Name specify the Harvest path with the following format:
     - `<package_name>/<project_name>/<state_name>/<repository_path>/<filename>`
4. Click **OK**.

4

Serena ChangeMan Integration

This chapter describes how to integrate Serena PVCS Dimensions with Oracle Enterprise Repository, which enables the use of a webdav-enabled PVCS repository within Oracle Enterprise Repository.

This chapter contains the following sections:

- Section 4.1, "Adding Serena Changeman Plug-ins"
- Section 4.2, "Configuring an Artifact Store"
- Section 4.3, "Configuring a PVCS Artifact Store"
- Section 4.4, "Add a File to an Asset Using a PVCS Repository"
- Section 4.5, "Security Considerations"

4.1 Adding Serena Changeman Plug-ins

To add Serena Changeman Plug-ins:

1. Navigate to this `Oracle_HOME\repositoryXXX\core\tools\solutions` directory on your computer. A list of all the solution packs for Oracle Enterprise Repository is displayed.
2. Unzip the **OER30-RC-Enhanced-SCM-Integrations.zip** file to a temporary directory.
3. Using an SQL tool appropriate for your database, run the SQL script located in the temporary directory to add Serena Changeman to your Oracle Enterprise Repository database.

4.2 Configuring an Artifact Store

This procedure is performed in the Oracle Enterprise Repository Asset Editor.

1. Select **Configure Artifact Stores** from the Actions menu.
2. Click **Add** to open the Create a new Artifact Store dialog.
3. Create a new SCM repository using the following parameters:
   - **Hostname**: `<app server name with port>` where Harvest-HTTP Artifact Store Integration was installed.
   - **Path**: *harvest_rep/retrieve/<harvest_broker_name>*
   - Leave Username and Password blank.
4. Click **OK**.

4.3 Configuring a PVCS Artifact Store

To configure a PVCS artifact store:

1. Login to Oracle Enterprise Repository as a user. (Requires assignment to a role that includes the Edit Artifact Stores permission.)
2. Click the **Edit/Manage Assets** link in the sidebar on the Assets screen. The Asset Editor launches.
3. Select **Configure Artifact Stores** in the Actions menu. The Configure Artifact Stores dialog is displayed.
4. Click the **Add** button. The Create a new Artifact Store dialog is displayed.
5. Enter a descriptive name for the repository.
6. In the Type list, select the PVCS repository type for this repository definition.
7. Enter a host name and path for the server.
8. If necessary, enter a username and password.

Note:

- This is an option; entering a username and password allows automatic login. To add a database password, add the "+" character and the database password. An example PVCS URL used to reach a specific file in PVCS through webdav resembles the following: *http://repositoryName+databaseName:password@host.server.com:port/webdavPath/additionalPathStructure/filename*
- If a username and/or a password is used to access this repository, check the option to Proxy Download Requests.

9. When finished, click **Save**.

4.4 Add a File to an Asset Using a PVCS Repository

To add a file to an asset using a PVCS repository:

1. Launch the Asset Editor.
2. Select the appropriate asset.
3. Navigate to the File Information section on the asset's Overview tab.
4. Click **Add**.
5. In the dialog, enter a name and description, if necessary.
6. Click **Edit**. The Edit URL dialog is displayed.
7. Select the **Repository File** option.
8. From the Host drop-down list, select **PVCS Repository**.
9. In the File Name box, enter the file name or browse to the file in the PVCS repository and select it.
10. Select **View** to test the URL.

4.5 Security Considerations

- Depending on the specific configuration, user names and passwords may be visible in URLs. Forcing individual user authentication at the file level solves the issue, but disables the Serena PVCS Dimensions browser in the Oracle Enterprise Repository Asset Editor. Using the Proxy Download Requests option in the artifact store definiton eliminates the possibility of the user gaining access to these details.
- Users may be able to exploit certain Serena PVCS Dimensions filepath characteristics to gain unauthorized access to restricted files when the Proxy Download Requests option is not selected in the artifact store definition.

Part II

Oracle Enterprise Repository Integration with Runtime Monitoring Tools

This part describes how to use the Enterprise Manager Integration Utility to integrate Enterprise Manager with Oracle Enterprise Repository. The Enterprise Manager Integration Utility is used from the command line.

This part contains the following chapters:

- Chapter 5, "Enterprise Manager Integration Utility"
- Chapter 6, "Integration with Amberpoint"

5 Enterprise Manager Integration Utility

This chapter describes how to get started with the Enterprise Manager Integration Utility and use it to integrate Enterprise Manager with Oracle Enterprise Repository.

This chapter contains the following sections:

- Section 5.1, "Overview"
- Section 5.2, "Using the Enterprise Manager Integration Utility"
- Section 5.3, "Configuring the Enterprise Manager Integration Utility"
- Section 5.4, "Encrypting the Configuration File Passwords"
- Section 5.5, "Known Issues"

5.1 Overview

Enterprise Manager (EM) is a comprehensive monitoring tool. It includes support for monitoring Oracle's SOA Products, as well as J2EE Servers, Databases, JVMs, and Operating Systems.

The EM Integration utility closes the loop between runtime and design-time by providing a summary of runtime performance metrics from EM into Oracle Enterprise Repository. The EM Integration utility pulls from EM and pushes to Oracle Enterprise Repository. The EM Integration tool currently supports and works with EM Grid Control 10g R3.

This section contains the following topics:

- Section 5.1.1, "Prerequisites"
- Section 5.1.2, "Obtaining the Enterprise Manager Integration Utility"
- Section 5.1.3, "High Level Use Cases"

5.1.1 Prerequisites

Before using the EM Integration utility, you must perform the following prerequisites:

- The EM Integration utility requires the OER-Harvester datapack to be installed in Oracle Enterprise Repository.
- The assets that receive metrics from EM must already be in Oracle Enterprise Repository. They should have been published to Oracle Enterprise Repository from SOA Suite/BPEL-PM or Oracle Service Bus using the Oracle Enterprise Repository Harvester. The EM Integration utility requires the OER-Harvester datapack to be installed in Oracle Enterprise Repository.

For more information about importing, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

- The EM Integration utility needs read access to the Enterprise Manager shared database views.

  For more information, see *http://download.oracle.com/docs/cd/B16240_01/doc/em.102/b40007/views.htm#BACCEIBI*.

- The EM Integration utility requires Java SDK version 6 or higher.
- Install 10*g* or 11*g* EM Grid Control to enable EM Integration tool to harvest into Oracle Enterprise Repository. The Oracle Enterprise Repository EM Integration tool, currently, integrates with EM-Grid control and works with both 10*g* and 11*g* version of SOA.

5.1.2 Obtaining the Enterprise Manager Integration Utility

To obtain the Enterprise Manager Integration utility:

1. Navigate to this `Oracle_HOME\repositoryXXX\core\tools\solutions` directory on your computer. A list of all the solution packs for Oracle Enterprise Repository is displayed.
2. Unzip the **11.1.1.x.x-EM-Integration.zip** file to a temporary directory.
3. Import the **11.1.1.x.x-OER-Harvester-Solution-Pack.zip** file into Oracle Enterprise Repository using the Import/Export Tool. For more information, see *Oracle Fusion Middleware User Guide for Oracle Enterprise Repository*.

5.1.3 High Level Use Cases

You can use the EM Integration utility to:

- Publish metrics from Oracle Service Bus Services in EM to existing Oracle Enterprise Repository assets.
- Publish metrics from BPEL-PM Processes in EM to existing Oracle Enterprise Repository assets.
- Publish metrics from Web Services in EM to existing Oracle Enterprise Repository assets.
- Publish Endpoint and Deployment : BPEL assets in Oracle Enterprise Repository from deployment information in EM.
- Support links back to the detailed metrics in EM, from the Oracle Enterprise Repository Web UI.
- Support configurability and extensibility, so that new targets in EM can be mapped to Oracle Enterprise Repository assets.

5.2 Using the Enterprise Manager Integration Utility

This section describes how to use the Enterprise Manager Integration Utility.

This section contains the following topics:

- Section 5.2.1, "Running from Command Line"
- Section 5.2.2, "Scheduling from Enterprise Manager"
- Section 5.2.3, "Metric Publishing"

5.2.1 Running from Command Line

The EM Integration utility can be run in the command line using the `em-integration.bat` utility (for Windows) or `em-integration.sh` (for Linux and Unix).

Before running `em-integration.bat` or `em-integration.sh`, ensure that the environment variables mentioned in Table 5–1 are set. In Windows, from a command window, you can type "set X" to see the value of the variable X, and "set X=abc" to set the value of FOO to "abc".

Table 5–1 Command Line Script

Environment Variable	Description
`JAVA_HOME`	Ensure that the `JAVA_HOME` environment variable points to an installed java runtime (JRE) or SDK. This must be Java version 5 or higher.
`JAVA_OPTS`	Ensure that you set your `JAVA_OPTS` parameter as follows: `set JAVA_OPTS=-Dhttp.proxyPort=80 -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.nonProxyHosts= "*.oracle.com|localhost"` `JAVA_OPTS` refers to the extra options to the java executable. In normal cases, there is no must set this variable. However, a common exception when you must set this variable is when your computer is inside a firewall, and you must use an HTTP proxy to access external servers. See Also: *http://java.sun.com/javase/6/docs/technotes/guides/net/proxies.html*

Table 5–2 shows the options that can be specified using the EM Integration command line utility:

Table 5–2 Command Line Options for the EM Integration

EM Integration Options	Description
`-settings <file>`	Refers to the configuration settings specified in the XML file.
`-er_url <URL>`	Specifies the URL for the Oracle Enterprise Repository instance.
`-er_user <User Name>`	Specifies the name of the Oracle Enterprise Repository user.
`-er_password <Password>`	Specifies the password of the Oracle Enterprise Repository user. To ensure security, the password must be encrypted. The Oracle Enterprise Repository Web console has a tool to encrypt passwords: *http://<host>:<port>/<domain>/diag/encryptstrings.jsp*
`-em_url <URL>`	Specifies the JDBC URL for the EM database.
`-em_user <User Name>`	Specifies the EM database user name.
`-em_password <Password>`	Specifies the EM database password. To ensure security, the password must be encrypted. The Oracle Enterprise Repository Web console has a tool to encrypt passwords: *http://<host>:<port>/<domain>/diag/encryptstrings.jsp* For more information about encrypting passwords from command-line, see Section 5.4, "Encrypting the Configuration File Passwords".

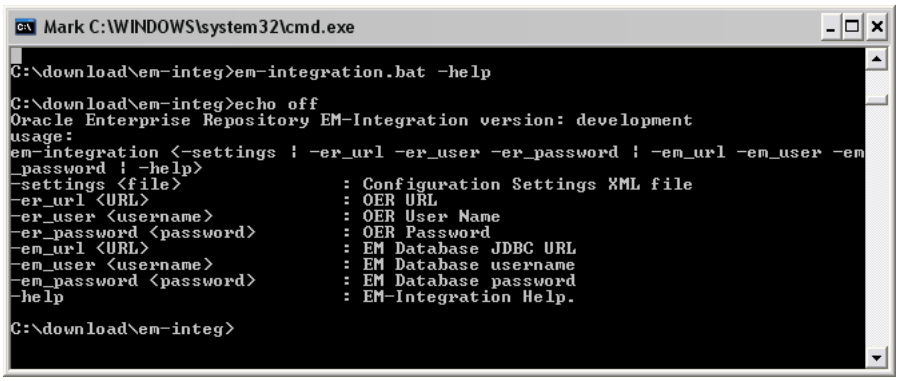
Table 5–2 (Cont.) Command Line Options for the EM Integration

EM Integration Options	Description
`-help`	Displays the online help for the EM Integration utility.

Note: Note none of the above options are mandatory and these can be omitted. If the options are omitted, then the utility makes use of the information in the `em-integration-settings.xml` file in the current directory where `em-integration.bat` resides. If the command line parameters are provided, then these override the information in `em-integration-settings.xml`.

Figure 5–1 shows the command line utility options and online help displayed by the `em-integration.bat -help` command.

Figure 5–1 Enterprise Manager Integration Command-Line Utility Options



5.2.2 Scheduling from Enterprise Manager

Enterprise Manager has the ability to schedule command-line applications to run periodically. The EM Integration utility can be scheduled to run this way.

For more information, see "Job System" in *Oracle Enterprise Manager Administrator's Guide*.

It is recommended that you schedule the EM Integration to run once a day, in the evening or when there is low activity on the system.

5.2.3 Metric Publishing

The section describes the various metric publishing options:

5.2.3.1 Oracle Service Bus

EM Integration updates existing assets in Oracle Enterprise Repository of type "`Endpoint`", which have an endpointURI attribute that matches the endpointURI metric in EM, for Oracle Service Bus Proxy Services and External Services.

Note: An External Service is a service, which is invoked by an Oracle Service Bus Business Service.

If no "Endpoint" asset is found with the matching endpointURI, then the tool attempts to find a matching Service asset, by name and Oracle Service Bus service type (BusinessService or ProxyService). If it finds one, then it creates a new "Endpoint" asset and relates it to the service. In the case of Business Services, the endpoint is related to the External Service invoked by the Business Service.

5.2.3.2 BPEL PM

EM Integration updates existing assets in Oracle Enterprise Repository of type "Deployment:BPEL", which has the "EntryPoint" relationship to an "Endpoint" asset that has an endpointURI attribute that matches the endpointURI metric in EM. EM stores the WSDL location of the endpoint.

If no "Endpoint" asset is found with the matching endpointURI, then the tool attempts to find a matching Service asset, by qualified name from the WSDL. If it finds one, then it creates a new "Endpoint" asset and relates it to the service.

If no "Deployment:BPEL" asset is found, then the tool attempts to find a maching Business Process: BPEL asset, by process name. If it finds one, then it creates a new "Deployment:BPEL" asset and relates it to the service.

5.2.3.3 Web Services

EM Integration updates existing assets in Oracle Enterprise Repository of type "Endpoint"", which have an endpointURI attribute that matches the endpointURI metric in EM.

If no "Endpoint" asset is found with the matching endpointURI, then the tool attempts to find a matching Service asset, by qualified name from the WSDL. If it finds one, then it creates a new "Endpoint" asset and relates it to the service.

5.2.3.4 Artifacts Creation

For every "Endpoint" that is created, the EM Integration Utility creates a corresponding Artifact:WSDL asset, if it does not already exist in Oracle Enterprise Repository. The EM Integration Utility also creates the Artifact:WSDL and Artifact:XSD assets for any imported file. It follows the fingerprinting rules in Oracle Enterprise Repository Harvester to look for existing artifacts in Oracle Enterprise Repository.

5.2.3.5 Endpoint Creation

It is common for production endpoints to be monitored in EM, but not captured in Oracle Enterprise Repository. The EM Integration utility creates these endpoints, if it can find a matching service.

The EM Integration utility creates Endpoint assets, if it can find a matching Service and creates Deployment:BPEL assets, if it can find a matching Business Process:BPEL.

5.2.3.6 Metrics to Update

In its default configuration, EM Integration attempts to update the following metrics in Oracle Enterprise Repository. These are all visible in the Operational tab for the asset in the Oracle Enterprise Repository Web user interface.

Not all of these metrics are gathered in EM for all of the target types mentioned above, so some might be omitted. The em-integration-settings.xml file configures which metrics are gathered for which targets.

See Also: Section 5.3.2.1, "Metric Mappings"

- Daily Average Response Time
- Weekly Average Response Time
- Monthly Average Response Time
- Minimum Response Time
- Maximum Response Time
- Daily Requests
- Weekly Requests
- Monthly Requests
- Daily Faults
- Weekly Faults
- Monthly Faults
- Availability - Daily %
- Availability - Weekly %
- Availability - Monthly %
- Availability - Daily %
- Current Rated Maximum Requests per Second
- Start Date for Metrics Monitoring
- Last Updated
- Link back to detailed metrics in EM

5.3 Configuring the Enterprise Manager Integration Utility

This section describes how to configure the Enterprise Manager Integration Utility.

This section contains the following topics:

- Section 5.3.1, "Setting the Repository and Enterprise Manager Connection Information for the Command-line Utility"
- Section 5.3.2, "Advanced Configuration"

5.3.1 Setting the Repository and Enterprise Manager Connection Information for the Command-line Utility

Open the `em-integration-settings.xml` file, which is located at `<EM Integration Home>` and modify the following XML chunk to point to the Oracle Enterprise Repository instance, the EM database, and the web console with the right credentials.

The EM database is used to retrieve the metric information that is sent to Oracle Enterprise Repository. The EM console URL is used as the base URL for links back to EM, that are created within Oracle Enterprise Repository.

The `introspectionSettings` section is used to configure the properties of any Endpoint assets that the utility creates. The `createMissingEndpoints` attribute

indicates whether Endpoint and `Deployment:BPEL` assets are created. If false, then the rest of the section in the code below is ignored.

The `matchMultipleEndpoints` attribute indicates the required action to be taken if the utility finds multiple Endpoints or `Deployment:BPEL` assets in Oracle Enterprise Repository that match a target in EM. If true, then it updates all the endpoints with metrics (and log a warning). If false, then it skips the target (and log a warning).

The `matchMultipleServices` attribute indicates the required action to be taken if the utility finds multiple Services or `Business Process:BPEL` assets in Oracle Enterprise Repository that match a target in EM, during endpoint creation. If true, then it attaches new Endpoints / `Deployment:BPEL`s to each of the matching assets (and log a warning). If false, then it skips the target (and log a warning).

The introspectionDescription, introspectionVersion, and registrationStatus attributes indicate properties that are set on any assets created.

```
<repository>
  <uri>http://localhost:7131/oer30/services/FlashlineRegistry</uri>
  <credentials>
    <user>admin</user>
    <password>*****</password>
    <enableTransaction>false</enableTransaction>
    <triggerEvent>false</triggerEvent>
  </credentials>
  <timeout>120000</timeout>
</repository>
<introspectionSettings>
  <createMissingEndpoints>true</createMissingEndpoints>
   <introspectionDescription>Found in EM </introspectionDescription>
    <introspectionVersion>1.0</introspectionVersion>
    <namespace>mythings</namespace>
    <!-- The Valid Registration states are 1)Unsubmitted 2)Submitted - Pending
 Review 3)Submitted - Under Review 4) Registered -->
    <registrationStatus>Submitted</registrationStatus>
  </introspectionSettings>
<emDatabase>
  <driverClass>oracle.jdbc.driver.OracleDriver</driverClass>
  <url>jdbc:oracle:thin@remotehost.there.com:1501:mysid</url>
  <credentials>
    <user>sysman</user>
    <password>*****</password>
  </credentials>
</emDatabase>
```

> **Note:** It is recommended that you run the Enterprise Manager Integration Utility as a user with the Basic Access Settings for Assets - View, Edit, Accept, and Register.

Alternatively, the connection information can also be passed as parameters to the command line utility as follows:

```
C:\test\em-integration> em-integration - er_url
http://localhost:7101/oer/services/FlashlineRegistryTr -er_user
admin - er_password <password> - em_url
jdbc:oracle:thin:@remotehost.there.com:1501:mysid - em_user
sysman - em_password <empassword>
```

5.3.2 Advanced Configuration

This section describes the following advanced configuration options for the Enterprise Manager Integration utility:

- Section 5.3.2.1, "Metric Mappings"
- Section 5.3.2.2, "Target Finders"
- Section 5.3.2.3, "Logging"

5.3.2.1 Metric Mappings

The mapping between metrics in EM and fields in Oracle Enterprise Repository are configured in `em-integration-settings.xml`, as shown in Figure 5–2. To change these from the default settings, open this file and modify the following:

- `<metricMappingGroup>`: Mappings for a set of metrics with the same metric name in the EM database views. EM groups a set of related metrics under the same "metric name". Each individual metric has a different "metric column". This element has the following attribute:
  - `metricName`: The EM metric name from the EM database views.
- `<metricMapping>`: Mapping for a single metric. This element has the following attributes:
  - `metricColumn`: The EM metric column from the EM database views.
  - `rollupPeriod`: Indicates which EM database view to query against, and how to aggregate the data. Must be one of {DAILY, WEEKLY, MONTHLY, CURRENT}.
  - `viewColumn`: Indicates which column in the EM database view to pull the data from. Must be one of {VALUE, AVERAGE, SUM, MINIMUM, MAXIMUM, STANDARD_DEVIATION, SAMPLE_COUNT, COLLECTION_TIMESTAMP, ROLLUP_TIMESTAMP, NOW(), DETAILS_URL}.
    - The CURRENT rollup period does not support columns AVERAGE, SUM, MINIMUM, MAXIMUM, STANDARD_DEVIATION, SAMPLE_COUNT, or ROLLUP_TIMESTAMP.
    - The DAILY, WEEKLY and MONTHLY rollup periods doe not support columns VALUE or COLLECTION_TIMESTAMP.
    - NOW() is a "virtual column" that contains the time of the query against EM.
    - DETAILS_URL is a "virtual column" that contains a URL to the detailed information in EM. This is constructed by the EM Integration code - it's not in the EM database views.
  - `oerField`: The internal name of the custom data field in Oracle Enterprise Repository where this data is stored.
  - `overwrite`: Indicates whether to overwrite the custom data field in Oracle Enterprise Repository if it already has data. It represents either true or false, by default the value is true, if not specified.

Figure 5–2 Metric Mappings

```
<!--OSB Metrics-->
<metricMappingGroup metricName="OSBResourceURIMetrics">
   <metricMapping metricColumn="countStat" rollupPeriod="DAILY" viewColumn="SUM" oerField="daily-requests"/>
   <metricMapping metricColumn="countStat" rollupPeriod="WEEKLY" viewColumn="SUM" oerField="weekly-requests" />
   <metricMapping metricColumn="countStat" rollupPeriod="MONTHLY" viewColumn="SUM" oerField="monthly-requests" />
   <metricMapping metricColumn="errorStat" rollupPeriod="DAILY" viewColumn="SUM" oerField="daily-faults" />
   <metricMapping metricColumn="errorStat" rollupPeriod="WEEKLY" viewColumn="SUM" oerField="weekly-faults" />
   <metricMapping metricColumn="errorStat" rollupPeriod="MONTHLY" viewColumn="SUM" oerField="monthly-faults" />
   <metricMapping metricColumn="avgStat" rollupPeriod="DAILY" viewColumn="AVERAGE" oerField="average-response-time--seconds-"/>
   <metricMapping metricColumn="avgStat" rollupPeriod="WEEKLY" viewColumn="AVERAGE" oerField="weekly-average-response-time--milliseconds-" />
   <metricMapping metricColumn="avgStat" rollupPeriod="MONTHLY" viewColumn="AVERAGE" oerField="monthly-average-response-time--milliseconds-" />
   <metricMapping metricColumn="avgStat" rollupPeriod="MONTHLY" viewColumn="MAXIMUM" oerField="maximum-response-time--milliseconds-" />
   <metricMapping metricColumn="avgStat" rollupPeriod="MONTHLY" viewColumn="MINIMUM" oerField="minimum-response-time--seconds-" />

   <!--pick an arbitrary metric to pull the timestamps and details from-->
   <metricMapping metricColumn="avgStat" rollupPeriod="CURRENT" viewColumn="COLLECTION_TIMESTAMP" oerField="last-updated" />
   <metricMapping metricColumn="avgStat" rollupPeriod="CURRENT" viewColumn="NOW()" oerField="endpoint-publication-date" overwrite="false" />
   <metricMapping metricColumn="avgStat" rollupPeriod="CURRENT" viewColumn="DETAILS_URL" oerField="detailed-performance-information" />
</metricMappingGroup>
```

5.3.2.2 Target Finders

The logic for correlating targets in EM with assets in Oracle Enterprise Repository is different for each target type. EM Integration ships with target finders for the following products: Oracle Service Bus, BPEL-PM, WebLogic.

If you are an advanced user, then you can create your own target finder classes. These must be configured in `em-integration-settings.xml`, as shown in Figure 5–3.

- `<targetFinder>`: Mappings for a target type in EM to a target finder class. This element has the following attributes
  - `targetType`: The EM target type from the EM database views.
  - `targetFinderClass`: The fully-qualified classname of the finder class. This must implement com.oracle.oer.integration.em.TargetFinder, which is found in em-integration.jar.

Figure 5–3 Target Finders

```
<!-- target finders -->
<targetFinder targetType="bea_alsb" targetFinderClass="com.oracle.oer.integration.em.targets.OSBTargetFinder" />
<targetFinder targetType="oracle_integrationbpm" targetFinderClass="com.oracle.oer.integration.em.targets.BPELPMTargetFinder" />
<targetFinder targetType="oc4j" targetFinderClass="com.oracle.oer.integration.em.targets.OC4JTargetFinder" />
<targetFinder targetType="weblogic_j2eeserver" targetFinderClass="com.oracle.oer.integration.em.targets.WebLogicTargetFinder" />
<targetFinder targetType="websphere_j2eeserver" targetFinderClass="com.oracle.oer.integration.em.targets.WebSphereTargetFinder" />
```

5.3.2.3 Logging

EM Integration uses log4j for logging the detailed tasks performed and the log file is placed in the `<EM Integration Home>`. The logging options can be changed by updating the `log4fl.properties` file located in the `<EM Integration>` folder.

5.4 Encrypting the Configuration File Passwords

To ensure security, the passwords in the configuration file must be encrypted before running the Enterprise Manager Integration utility. You can encrypt the configuration file passwords in the EM using either of the following methods:

- "From the Command Prompt"
- "From the Oracle Enterprise Repository Web Console"

From the Command Prompt

1. Navigate to the `<em_integration_home>` directory.
2. From a command prompt, run the password encryption tool as follows:

```
> encrypt.bat em-integration-settings.xml
em-integration-settings.xml
```

The password encryption tool, (`encrypt.bat/encrypt.sh`), which is located in *<Oracle_home>/tools/solutions/11.1.1.x.x-OER-PasswordTools.zip*, allows you to encrypt the passwords that are stored in the configuration file, `em-integration-settings.xml`.

From the Oracle Enterprise Repository Web Console

1. Navigate to the Oracle Enterprise Repository Web console at

   http://<host>:<port>/<domain>/diag/encryptstrings.jsp (replace host, port, and domain to match to your computer credentials)

2. Scroll down to the Tools section and click the **Encrypt Strings for passwords** link to launch the Password encryption page.
3. Enter the clear text password into the String to Encrypt text box.
4. Click the **Submit Query** button.
5. Copy the resulting encrypted password string and paste it into the appropriate context or properties file(s).

5.5 Known Issues

This section describes the following known Enterprise Manager Integration Utility issues:

5.5.1 Using Incorrect Encrypted Password

If you configure the `em-integration-settings.xml` file with incorrect encrypted password, then a very long stack trace is displayed. An extract from the very long stack trace is as follows:

```
        at org.apache.xerces.impl.XMLNSDocumentScannerImpl.scanStartElement(Unk
own Source)
        at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl$FragmentConten
Dispatcher.dispatch(Unknown Source)
        at org.apache.xerces.impl.XMLDocumentFragmentScannerImpl.scanDocument(U
known Source)
        at org.apache.xerces.parsers.XML11Configuration.parse(Unknown Source)
        at org.apache.xerces.parsers.DTDConfiguration.parse(Unknown Source)
        at org.apache.xerces.parsers.XMLParser.parse(Unknown Source)
        at org.apache.xerces.parsers.AbstractSAXParser.parse(Unknown Source)
        at javax.xml.parsers.SAXParser.parse(SAXParser.java:395)
        at org.apache.axis.encoding.DeserializationContext.parse(Deserializatio
Context.java:227)
        at org.apache.axis.SOAPPart.getAsSOAPEnvelope(SOAPPart.java:696)
        at org.apache.axis.Message.getSOAPEnvelope(Message.java:424)
        at org.apache.axis.handlers.soap.MustUnderstandChecker.invoke(MustUnder
tandChecker.java:62)
        at org.apache.axis.client.AxisClient.invoke(AxisClient.java:206)
        at org.apache.axis.client.Call.invokeEngine(Call.java:2765)
        at org.apache.axis.client.Call.invoke(Call.java:2748)
        at org.apache.axis.client.Call.invoke(Call.java:2424)
        at org.apache.axis.client.Call.invoke(Call.java:2347)
        at org.apache.axis.client.Call.invoke(Call.java:1804)
        at com.flashline.registry.openapi.service.v300.FlashlineRegistryTrSoapB
ndingStub.authTokenCreateWithLicense(FlashlineRegistryTrSoapBindingStub.java:74
```

```
1)
        at com.oracle.oer.sync.plugin.writer.oer.ALERConnectionCache.getAuthTok
n(ALERConnectionCache.java:128)
        at com.oracle.oer.sync.plugin.writer.oer.ALERAssetQueries.getToken(ALER
ssetQueries.java:82)
        at com.oracle.oer.sync.plugin.writer.oer.ALERAssetQueries.assetTypeQuer
ByUUID(ALERAssetQueries.java:159)
        at com.oracle.oer.sync.framework.MetadataManager.putAssetType(MetadataM
nager.java:204)
        at com.oracle.oer.sync.framework.impl.DefaultPluginManager.processIntro
pector(DefaultPluginManager.java:104)
        ... 5 more
```

6

Integration with Amberpoint

Note: Integration of Oracle Enterprise Repository with Amberpoint, which is now rebranded to Oracle Business Transaction Manager(OBTM), is supported only to product versions prior to OBTM 11. This integration support with Oracle Enterprise Repository is not supported for OBTM 11.

This chapter describes how to get started with the integration of Oracle Enterprise Repository with Amberpoint.

Oracle Exchange Utility updates the endpoint asset with the performance metrics deposited by Amberpoint. The **UDDIMappings.xml** file contains the mapping between the performance metrics t-models and the keys to Oracle Enterprise Repository custom fields where these metrics are populated. You can also customize the UDDI Mapping file so that the metrics can appear on any tab mapped to any field in the tabs of any custom asset type. The following snippet describes the metrics part in the `UDDIMappings.xml` file:

```
<metrics>
        <keyedReference metricsName="DailyAvailability"
alerName="availability---year----"
tModelKey="uddi:amberpoint.com:management:metrics:availability" keyName="Last 24
hours - Availability (percentage value)"/>
        <keyedReference metricsName="WeeklyAvailability"
alerName="availability---week----"
ModelKey="uddi:amberpoint.com:management:metrics:availability" keyName="Last week
- Availability (percentage value)"/>
        <keyedReference metricsName="MonthlyAvailability"
alerName="availability---month----"
tModelKey="uddi:amberpoint.com:management:metrics:availability" keyName="Last
month - Availability (percentage value)"/>
        <keyedReference metricsName="DailyAvgResponseTime"
alerName="average-response-time--seconds-"
tModelKey="uddi:amberpoint.com:management:metrics:avgResponseTime" keyName="Last
24 hours - Response Time (average value in milliseconds)"/>
        <keyedReference metricsName="WeeklyAvgResponseTime"
alerName="weekly-average-response-time--milliseconds-"
ModelKey="uddi:amberpoint.com:management:metrics:avgResponseTime" keyName="Last
week - Response Time (average value in milliseconds)"/>
        <keyedReference metricsName="MonthlyAvgResponseTime"
alerName="monthly-average-response-time--milliseconds-"
tModelKey="uddi:amberpoint.com:management:metrics:avgResponseTime" keyName="Last
month - Response Time (average value in milliseconds)"/>
        <keyedReference metricsName="DailyFaults" alerName="daily-faults"
```

```
tModelKey="uddi:amberpoint.com:management:metrics:faults"
keyName="Last 24 hours - Faults (number)"/>
        <keyedReference metricsName="WeeklyFaults" alerName="weekly-faults"
tModelKey="uddi:amberpoint.com:management:metrics:faults"
keyName="Last week - Faults (number)"/>
        <keyedReference metricsName="MonthlyFaults" alerName="monthly-faults"
tModelKey="uddi:amberpoint.com:management:metrics:faults"
keyName="Last month - Faults (number)"/>
        <keyedReference metricsName="DailyRequests" alerName="daily-requests"
tModelKey="uddi:amberpoint.com:management:metrics:requests"
keyName="Last 24 hours - Requests (number)"/>
        <keyedReference metricsName="WeeklyRequests" alerName="weekly-requests"
tModelKey="uddi:amberpoint.com:management:metrics:requests"
keyName="Last week - Requests (number)"/>
        <keyedReference metricsName="MonthlyRequests" alerName="monthly-requests"
tModelKey="uddi:amberpoint.com:management:metrics:requests"
keyName="Last month - Requests (number)"/>
        <keyedReference metricsName="Timestamp" alerName="last-updated"
tModelKey="uddi:amberpoint.com:management:metrics:timeStamp"
keyName=""/>
        <keyedReference metricsName="RegisteredDate"
alerName="endpoint-publication-date"
tModelKey="uddi:amberpoint.com:management:registeredDate"
keyName=""/>
    </metrics>
```

Figure 6–1 illustrates how the performance metrics deposited by Amberpoint shows up in Oracle Enterprise Repository after the endpoint is synchronized by the Exchange Utility.

Figure 6–1 Oracle Enterprise Repository Operation Information

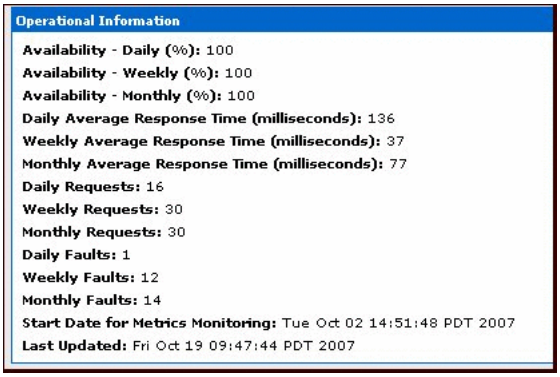
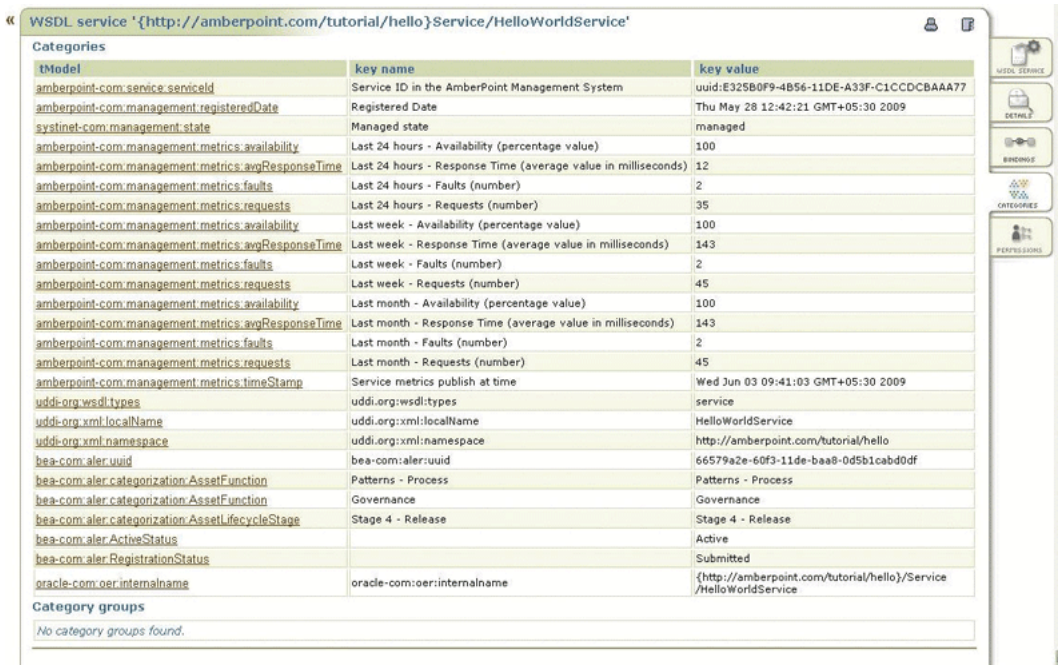


Figure 6–2 illustrates how the performance metrics deposited by Amberpoint appear in Oracle Service Registry.

Figure 6–2 Oracle Service Registry Performance Metrics for an Amberpoint WSDL Service



WSDL service '{http://amberpoint.com/tutorial/hello}Service/HelloWorldService'

Categories

tModel	key name	key value
amberpoint-com:service:serviceId	Service ID in the AmberPoint Management System	uuid:E325B0F9-4B56-11DE-A33F-C1CCDCBAAA77
amberpoint-com:management:registeredDate	Registered Date	Thu May 28 12:42:21 GMT+05:30 2009
systinet-com:management:state	Managed state	managed
amberpoint-com:management:metrics:availability	Last 24 hours - Availability (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Last 24 hours - Response Time (average value in milliseconds)	12
amberpoint-com:management:metrics:faults	Last 24 hours - Faults (number)	2
amberpoint-com:management:metrics:requests	Last 24 hours - Requests (number)	35
amberpoint-com:management:metrics:availability	Last week - Availability (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Last week - Response Time (average value in milliseconds)	143
amberpoint-com:management:metrics:faults	Last week - Faults (number)	2
amberpoint-com:management:metrics:requests	Last week - Requests (number)	45
amberpoint-com:management:metrics:availability	Last month - Availability (percentage value)	100
amberpoint-com:management:metrics:avgResponseTime	Last month - Response Time (average value in milliseconds)	143
amberpoint-com:management:metrics:faults	Last month - Faults (number)	2
amberpoint-com:management:metrics:requests	Last month - Requests (number)	45
amberpoint-com:management:metrics:timeStamp	Service metrics publish at time	Wed Jun 03 09:41:03 GMT+05:30 2009
uddi-org:wsdl:types	uddi.org:wsdl:types	service
uddi-org:xml:localName	uddi.org:xml:localName	HelloWorldService
uddi-org:xml:namespace	uddi.org:xml:namespace	http://amberpoint.com/tutorial/hello
bea-com:aler:uuid	bea-com:aler:uuid	66579a2e-60f3-11de-baa8-0d5b1cabd0df
bea-com:aler:categorization:AssetFunction	Patterns - Process	Patterns - Process
bea-com:aler:categorization:AssetFunction	Governance	Governance
bea-com:aler:categorization:AssetLifecycleStage	Stage 4 - Release	Stage 4 - Release
bea-com:aler:ActiveStatus		Active
bea-com:aler:RegistrationStatus		Submitted
oracle-com:oer:internalname	oracle-com:oer:internalname	{http://amberpoint.com/tutorial/hello}/Service /HelloWorldService

Category groups

No category groups found.

Part III

Oracle Enterprise Repository Integration with Development Environments

This part describes how to configure Oracle JDeveloper and other IDEs to easily consume files from Oracle Enterprise Repository.

This part contains the following chapters:

- Chapter 7, "Integration with Development Environments"
- Chapter 8, "Configuring Oracle Enterprise Repository to Support Integration with Your IDE"
- Chapter 9, "Configuring Your IDE to Support Integration with Oracle Enterprise Repository"
- Chapter 10, "Using the IDE to Interact with Oracle Enterprise Repository"

7

Integration with Development Environments

This chapter describes how to integrate Oracle Enterprise Repository with various development environments.

This chapter contains the following sections:

- Section 7.1, "Overview"
- Section 7.2, "Best Practices"
- Section 7.3, "High Level Use Cases"

7.1 Overview

The Oracle Enterprise Repository provides integration within development environments so developers can easily search for and use assets from the repository without leaving their developent environment. Assets and any associated artifacts are downloaded directly to the developer's workspace. Integration also provides a convenient way to submit or harvest assets from the development environment into the Oracle Enterprise Repository for use throughout the enterprise.

Repository Access within the developer's workspace also provides a view into Oracle Enterprise Repository that enables developers to, download artifacts and assets from the repository, query the repository, and view the contents of the repository.

The goal of this integration is to ensure that SOA Governance becomes part of the fabric of every day development.

This section contains the following topics:

- Section 7.2, "Best Practices"
- Section 7.3, "High Level Use Cases"

7.2 Best Practices

This section describes the following best practice processes:

- Section 7.2.1, "Asset Production Process"
- Section 7.2.2, "Asset Consumption Process"

7.2.1 Asset Production Process

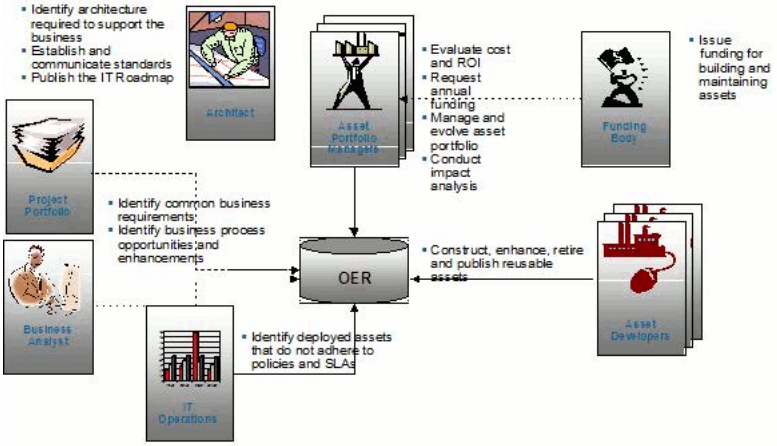
Oracle Enterprise Repository is used to track new assets and asset enhancements from the time they are proposed through the retirement of the assets, which also includes the time when the assets are being developed and when the assets are completed.

Through Oracle Enterprise Repository, assets are used to produce and consume projects, to provide traceability and support impact analysis, and to point out project-level impacts of changes to the asset release plans.

Initially, a business analyst provides a description of the functionality that must be produced. If determined that the proposed functionality does not already exist, then an architect provides the functional designs and non-functional requirements. The development team then produces, harvests, or enhances an asset that meets the functional and non-functional requirements.

Figure 7–1 describes a sample asset production process.

Figure 7–1 The Asset Production Process



This means that the development team needs visibility into the assets that they are to produce such as the requirements, use cases and so on. Oracle Enterprise Repository provides that visibility by allowing developers to view an asset details of an asset directly from the development environment. The development team builds the specified asset and then harvests the asset into the Oracle Enterprise Repository, where it goes through a review and approval process.

This section contains the following topics:

- Section 7.2.1.1, "Policies"
- Section 7.2.1.2, "Propose/Submit Assets"

7.2.1.1 Policies

The production and enhancement of assets can be governed through design-time policies. Policies are applied to assets to communicate asset requirements that must be considered during design and development, and to provide administrators with the means to enforce and monitor asset compliance with governance, architecture, and other organizational standards. For example, a policy might articulate corporate quality standards, identifying the platforms that an asset should run on and identifying acceptable defect density rates.

- A policy can be applied to multiple assets.
- Multiple policies can be applied to any asset.
- Each policy consists of at least one assertion statement. Each assertion has a name and description and includes a technical definition. The technical definition accommodates additional metadata that may be required to automatically validate the assertion using third-party testing and validation tools. This metadata may be

Web service-specific policy information, XML, or any other format that can be read by an external system. For example, an assertion statement for defect density might state that defect density must be less than .1% .

Policy information in Oracle Enterprise Repository can be accessed through the development environment. Once the development team harvests an asset into Oracle Enterprise Repository, then the policies can be automatically validated through tooling or manually validated by subject matter experts.

7.2.1.2 Propose/Submit Assets

Assets may be proposed or submitted to the Oracle Enterprise Repository in multiple ways, depending on the role and situation, some of the methods are as follows:

- The general Oracle Enterprise Repository user community can submit asset requests or completed assets from the console.
- Business analysts and architects can create assets to be built using the Asset Editor.
- Development teams can submit or harvest assets from their development environments.
- Existing assets stored in files or directories can be harvested by Competency Centers or Portfolio Managers.
- QA or IT Operations can harvest assets from the build environment or from the run time environment.

7.2.2 Asset Consumption Process

There are two primary design-time consumption models within a standard Software Development Lifecycle (SDLC) process:

Model 1: Assets are identified early in the lifecycle (also known as Prescriptive Reuse)

- Business analysts and/or project architects identify assets that fulfill the functional and non-functional requirements of the project.
- Development teams then receive a "kit" of relevant assets.

Model 2: Development-driven discovery

- Developers identify assets that might fulfill the functional and non-functional requirements of the project.

Model 1 is the most-preferred design-time consumption models because it results in a higher level of reuse, but requires the organization to have a mature SDLC process and well-defined roles.

This section contains the following topics:

- Section 7.2.2.1, "Prescription Reuse"
- Section 7.2.2.2, "Developer-Driven Discovery"
- Section 7.2.2.3, "Automated Usage Detection"

7.2.2.1 Prescription Reuse

Within the Oracle Enterprise Repository Web console, business analysts and/or project architects identify assets that fulfill the functional and non-functional requirements of the project. They package these assets into a "kit" called a Compliance Template. A Compliance Template communicates asset requirements or asset solution

sets to internal or outsourced project teams. Two types of Compliance Templates included in Oracle Enterprise Repository are:

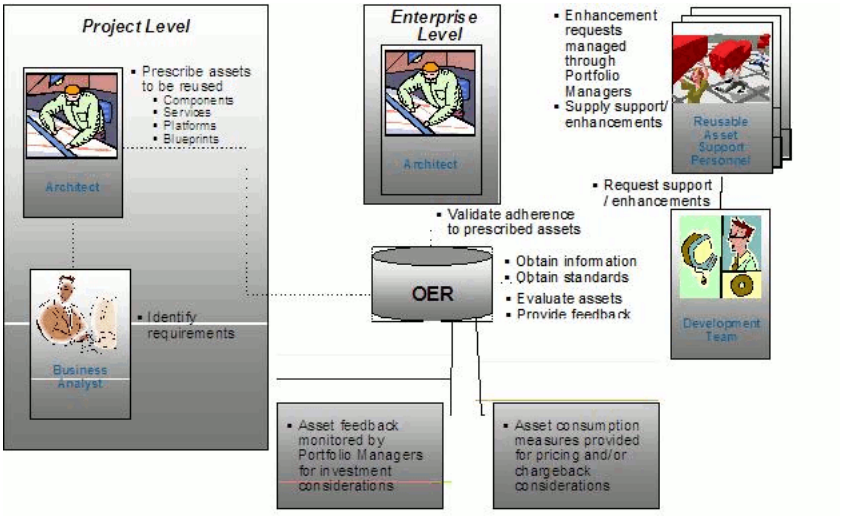
- Project Profiles
- Architecture Blueprints

Project profiles are usually created for individual projects, whereas architecture blueprints are reusable solution sets that can be leveraged by multiple projects. Some of the common use cases include:

- Project planners generate a project profile for each project in the portfolio, identifying the reusable assets that factored into the project's planning and estimating assumptions.
- Business analysts generate a project profile to identify assets that fulfill a project's business requirements.
- Project architects generate a project profile to identify assets that fulfill a project's technical requirements.
- Enterprise architects generate an architecture blueprint that specifies the standard frameworks and assets that are to be used to fulfill project-level security requirements.
- Those responsible for Service-Oriented Architecture(s) (SOA) generate an architecture blueprint to identify the services that orchestrate a particular business function.
- Product line architects generate an architecture blueprint that specifies the assets that are to be used to build a specific product line (similar to a Bill of Material).

Figure 7–2 describes a sample prescriptive reuse process.

Figure 7–2 Prescriptive Reuse



Compliance Templates are applied to projects through the Oracle Enterprise Repository Web console and the assets identified in the compliance template are automatically displayed in the project's development environment. In this way, development teams get a jump-start on their development efforts.

For more information about Compliance Templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

7.2.2.2 Developer-Driven Discovery

There are two ways that developers can get assets from the Oracle Enterprise Repository:

- Through the Oracle Enterprise Repository Web Console
- Through their IDE (Note that this functionality is currently available for Eclipse and VS .NET. Developers are able to download assets from JDeveloper in a future release.)

Developers can come to the Oracle Enterprise Repository to obtain the assets that they would like to use on their projects. The Use-Download function in the Oracle Enterprise Repository provides access to the asset artifacts. In addition, developers can select and download all assets related to the primary asset, ensuring that they have all necessary dependencies. The repository tracks usage and generates usage-based reports.

Developers can also access Oracle Enterprise Repository from their development environment. This means that developers never have to leave their IDE's to get access to the assets that they need.

7.2.2.3 Automated Usage Detection

Oracle Enterprise Repository tracks and reports on the design-time use of assets. The automated usage detection is tracked through two methods:

- Through the manual asset Use - Download process within Oracle Enterprise Repository or through the development environment
- Automatic usage detection leveraging Software File Identification (SFID)

Software File Identification (SFID) provides the ability to determine asset usage independent of the manual asset Use - Download process. The SFID process tags selected files and asset artifacts with a unique SFID fingerprint. This tag is then used to detect when and where an asset is used, even if the asset was acquired through means other than the Use - Download process. An instance of usage is recorded by Oracle Enterprise Repository when tagged files within the asset are opened in a developer's IDE. For more information about SFID, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

7.3 High Level Use Cases

The Section 7.2, "Best Practices" provides an overview of the production and consumption processes, and encompasses the development environment use cases. The individual use cases are described below. Each development environment includes a subset of these use cases.

- Section 7.3.1, "Submit Files"
- Section 7.3.2, "Harvest Files"
- Section 7.3.3, "Search Oracle Enterprise Repository"
- Section 7.3.4, "View Asset Details"
- Section 7.3.5, "Download Artifacts"
- Section 7.3.6, "Prescriptive Reuse"
- Section 7.3.7, "Automatic Usage Detection"

7.3.1 Submit Files

Through the development environment, developers can select files to submit to the Enterprise Repository. The files are bundled into a .zip format for submission. The developer can submit single and/or compound-payload assets to Oracle Enterprise Repository.

7.3.2 Harvest Files

Oracle Enterprise Repository can harvest from Oracle products and standards based files. This includes Oracle SOA Suite, Oracle Service Bus, and standard BPEL, WSDL, XSD and XSLT files and file directories. When harvested, Oracle Enterprise Repository automatically creates assets, populates asset metadata, and generates relationship links based on the information in the artifact files. The harvesting function is available from the command line, and can be integrated into the IDE, or into the build process.

For more information, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

7.3.3 Search Oracle Enterprise Repository

Access to assets and artifacts in the Enterprise Repository is available through the development environment. Through the IDE, developers can search for assets matching various criteria or view assets that may be of interest to a development project project.

7.3.4 View Asset Details

For selected assets, developer can view asset details, such as description, usage history, expected savings, relationships, etc. Within the asset metadata, links to supporting documentation, user guides, test cases, etc., are provided to better enable developers to reuse existing functionality.

7.3.5 Download Artifacts

Developers can download an asset's artifacts (i.e., payload) into their project. Typically an asset payload is the functionality that a developer must use a service (such as a WSDL file) or incorporate into their code base (such as a binary or a BPEL file).

7.3.6 Prescriptive Reuse

Through the Oracle Enterprise Repository, analysts, architects, technical leads, and others who are involved in the design stages of a project can create a list of assets that fulfills a project's requirements. The list of assets are captured in compliance templates in the repository and the compliance templates are associated with an Oracle Enterprise Repository project.

From within the IDE, you can view a list of assets appearing in all of the Compliance Templates assigned to your project. You can see which of the assets have been used and/or other project members. For more information about compliance templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

7.3.7 Automatic Usage Detection

Oracle Enterprise Repository can automatically detect asset reuse within the development environment. This allows development teams to ensure that they get asset reuse credit, regardless of whether the assets have been downloaded through

Oracle Enterprise Repository or pulled from another source, such as the developer's desktop.

For more information about each of these use cases for various IDEs, see Chapter 10, "Using the IDE to Interact with Oracle Enterprise Repository".

Table 7–1 describes the specific use cases supported for each development environment.

Table 7–1 Use Cases and Supported Development Environments

Use Case	JDeveloper	Eclipse	VS .NET
Submit Files		Yes	
Harvest (BPEL, WSDL, XSD, XSLT)	Yes	Yes	Yes
Harvest (SCA)	11g		
Search Oracle Enterprise Repository	11g	Yes	Yes
View Asset Details	11g	Yes	Yes
Download Artifacts	11g	Yes	Yes
Prescriptive Reuse		Yes	Yes
Automatic Usage Detection		Yes	Yes

Table 7–2 describes the supported versions and target audience for each IDE.

Table 7–2 Supported Versions for the IDEs

IDE	Target Audience	Supported Versions
JDeveloper	Oracle Customers	10gR3, 11gR1
Eclipse	Java Developers	3.4
		3.3 with Sun JDK 5.0
		2.0.3 with Sun JDK 5.0
VS .NET	.Net Developers	2008
		2005

8

Configuring Oracle Enterprise Repository to Support Integration with Your IDE

This chapter describes how to configure Oracle Enterprise Repository to support Integration with your IDE.

You must perform some steps in the Oracle Enterprise Repository before configuring the IDE. The steps are as mentioned below:

- Section 8.1, "Install the Harvester"
- Section 8.2, "Assign IDE Users to Oracle Enterprise Repository Projects"
- Section 8.3, "Establish Compliance Templates"
- Section 8.4, "Set up Automatic Usage Detection"

8.1 Install the Harvester

The Harvester can harvest standards-based artifacts generated from any IDE such as Oracle JDeveloper, Eclipse, and VS .NET. The Harvester can be integrated with any of these IDEs and then run with the respective IDE client.

For more information about the Harvester, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

8.2 Assign IDE Users to Oracle Enterprise Repository Projects

Oracle Enterprise Repository tracks assets produced by projects in any IDE such as Oracle JDeveloper, Eclipse, and VS .NET, as well as assets consumed by the projects created in these IDEs.

For more information about adding a new project and assigning users to the project, see *Oracle Fusion Middleware User Guide for Oracle Enterprise Repository*.

8.3 Establish Compliance Templates

Compliance Templates describe a particular family of Oracle Enterprise Repository artifacts and are available only for some product configurations such as Eclipse and VS .NET. Compliance templates are required to support the Prescriptive Reuse use cases.

For more information about how to establish compliance templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

8.4 Set up Automatic Usage Detection

Software File Identification (SFID) is a process of determining asset usage in Oracle Enterprise Repository. You can use SFID to work with your development environment such as Eclipse and VS .NET. Depending on your IDE, SFID requires the installation of the Oracle Enterprise Repository Plug-in for Workspace Studio, which is an Eclipse-based IDE, or the Oracle Enterprise Repository Plug-in for Visual Studio .NET.

For more information about setting up automatic usage detection using SFID, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

9

Configuring Your IDE to Support Integration with Oracle Enterprise Repository

This chapter describes how to get started with Harvester and its use in various high level use cases.

- Section 9.1, "Configuring Oracle JDeveloper"
- Section 9.2, "Configuring Eclipse"
- Section 9.3, "Configuring VS .NET"

9.1 Configuring Oracle JDeveloper

This section contains the following topics:

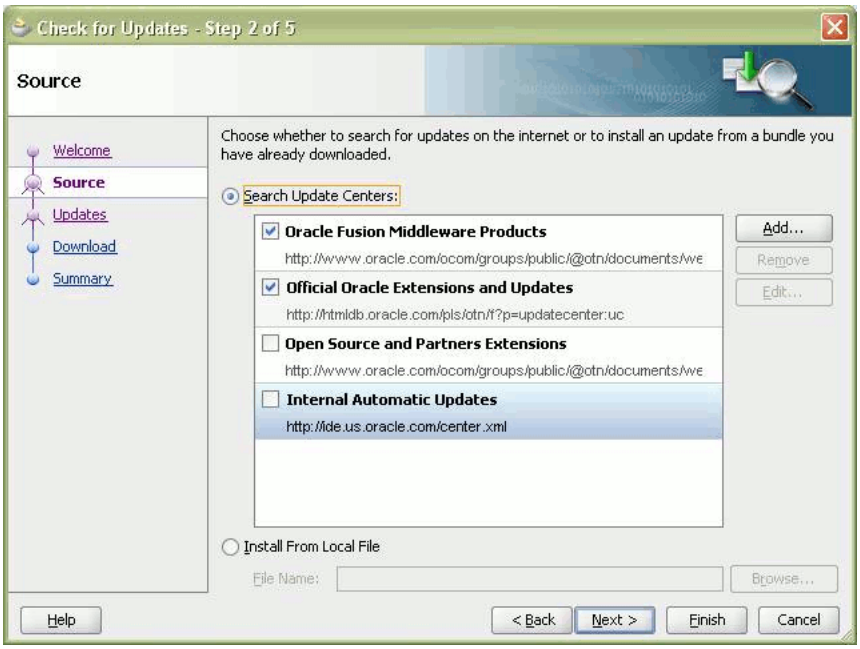
- Section 9.1.1, "Integrating with Oracle JDeveloper 11g R1 Patchset Releases"
- Section 9.1.2, "Integrating with Oracle JDeveloper 11g R1"
- Section 9.1.3, "Integrating with Oracle JDeveloper 10g"

9.1.1 Integrating with Oracle JDeveloper 11*g* R1 Patchset Releases

To create a connection between the Oracle Enterprise Repository and Oracle JDeveloper 11*g* Release 1 PatchSets:

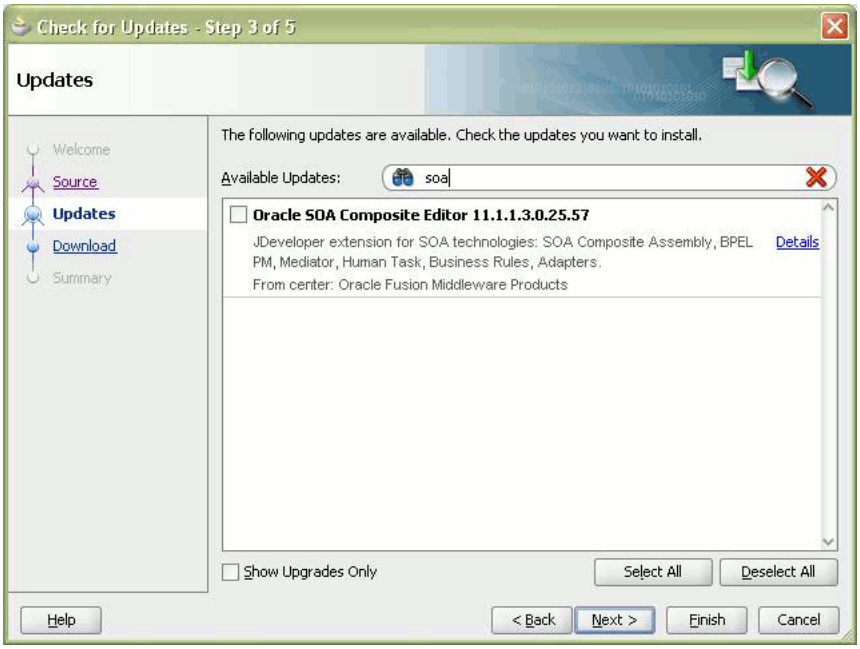
1. Install Oracle JDeveloper on your local computer.
2. Open Oracle JDeveloper and click **Help**, **Check for Updates**. The Check for Updates dialog is displayed.
3. Click **Next**. The Source page is displayed, as shown in Figure 9–1.

Figure 9–1 Check for Updates - Source Page



4. Click **Next**. The Updates page is displayed.
5. Enter `soa` in the Available Updates box, as shown in Figure 9–2, and search for the update.

Figure 9–2 Check for Updates - Updates Page

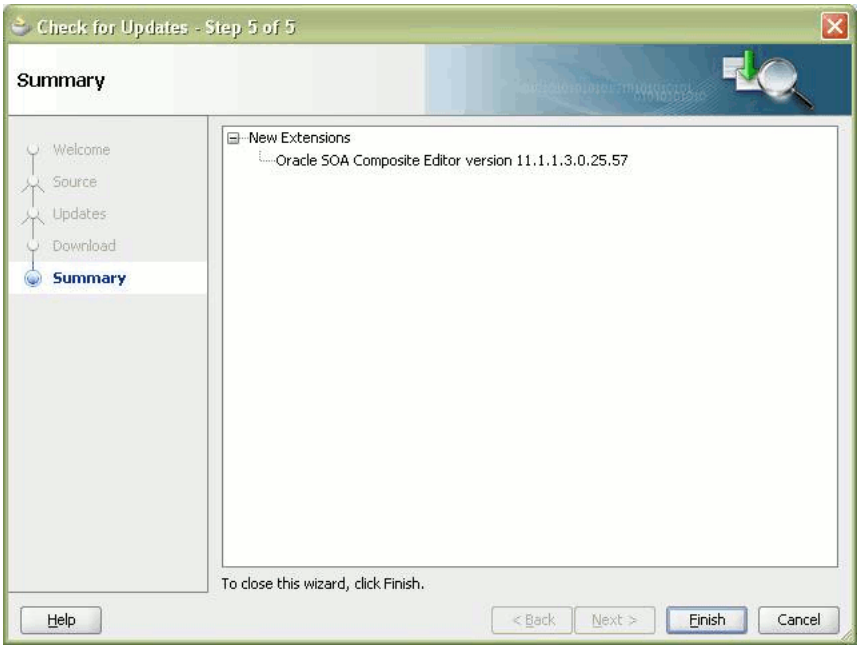


6. Select the SOA JDeveloper Extension update and click **Next**. The Download page is displayed.

Note: The version of Composite editor may change based on the JDeveloper version you are using.

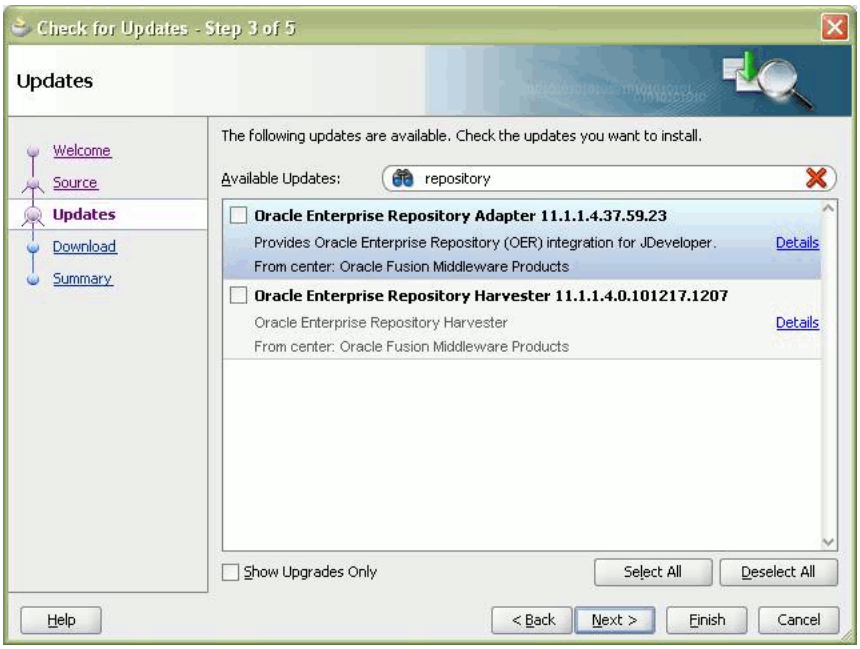
7. Click **Next**. The Summary page is displayed, as shown in Figure 9–3.

Figure 9–3 Check for Updates - Summary Page



8. Click **Finish**.
9. Restart Oracle JDeveloper. The Oracle JDeveloper window is displayed.
10. Repeat Steps 2 to 4. The Updates page is displayed.
11. Enter `repository` in the Available Updates box, as shown in Figure 9–4, and search for the update.

Figure 9–4 Check for Updates - Updates Page

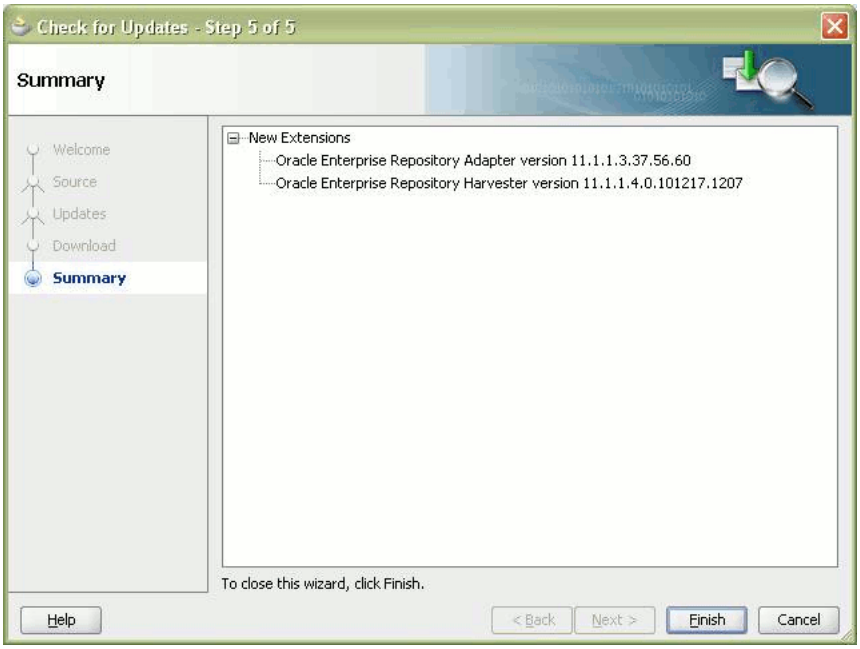


12. Select both the Extensions, as shown in Figure 9–4, and click **Next**. The Download page is displayed.

> **Note:** The versions of Oracle Enterprise Repository Adapter and Oracle Enterprise Repository Harvester Extensions may vary based on the JDeveloper version you are using.

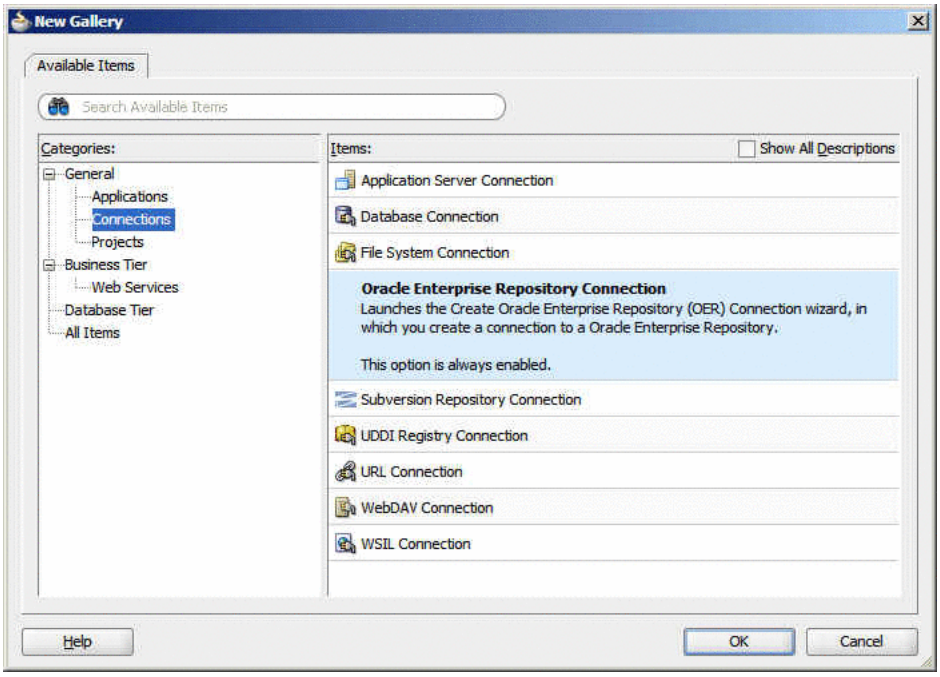
13. Click **Next**. The Summary page is displayed, as shown in Figure 9–5.

Figure 9–5 Check for Updates - Summary Page



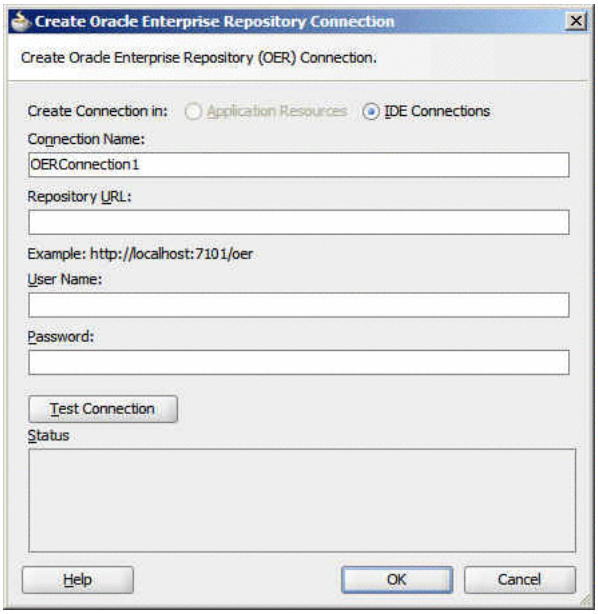
14. Click **Finish**.
15. Restart Oracle JDeveloper. The Oracle JDeveloper window is displayed.
16. Click **File**, **New**. The New Gallery dialog is displayed.
17. Select **General**, **Connections**, and then select **Oracle Enterprise Repository Connection**, as shown in Figure 9–6.

Figure 9–6 New Gallery Dialog



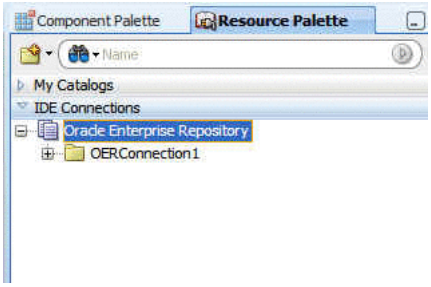
18. Click **OK**. The Create Oracle Enterprise Repository Connection dialog is displayed, as shown in Figure 9–7.

Figure 9–7 Create Oracle Enterprise Repository Connection Dialog



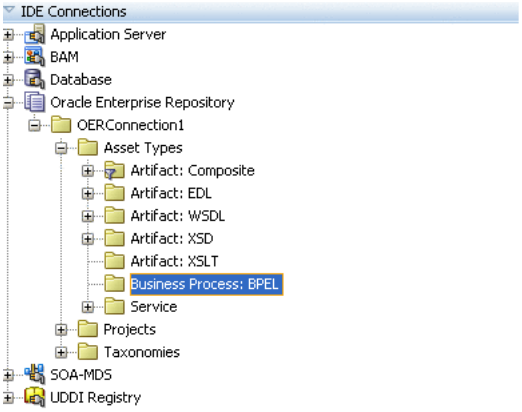
19. Enter the following information:
    - Repository URL: The URL from where a running instance of Oracle Enterprise Repository can be accessed.
    - User Name: The user name for the Oracle Enterprise Repository.
    - Password: The password for the Oracle Enterprise Repository.
20. Click **Test Connection**. A success message is displayed in the Status pane.
21. Click **OK**.
22. In the Resource Palette, under IDE Connections, expand Oracle Enterprise Repository to see the application server connection that you created, as shown in Figure 9–8.

Figure 9–8 Resource Palette



23. Enter a search criteria to search for assets in the Search field. A list of all the assets is displayed, as shown in Figure 9–9.

Figure 9–9 Search Results



Note:

- You have to install SOA solution pack into Oracle Enterprise Repository to enable harvesting from JDeveloper. This is applicable for both JDeveloper 10*g* and 11*g*.
- Steps 16 - 23 that describe the procedure for creating and testing the Oracle Enterprise Repository connection is same for all the patchset releases, i.e. for PS2 (11.1.1.3.0) and PS3 (11.1.1.4.0).

9.1.2 Integrating with Oracle JDeveloper 11*g* R1

To install the harvester Ant tasks in JDeveloper 11g or in SOA Suite 11g, perform the following steps:

1. Double-click **jdeveloper.exe** in the jdev installation directory to open Oracle JDeveloper.

   Note: You should start and close the JDeveloper application at least once during this step.

2. Unzip the harvester zip file into your JDeveloper installation. The contents of the zip file are extracted to the *<jdeveloper_home>/harvester* directory.
3. Edit the `tools11g.xml` file to match your JDeveloper installation.
4. Merge the contents of `tools11g.xml` into your JDeveloper's product preferences XML file located in the *<jdeveloper_home>\system11.1.1.x.xx.xx.xx\o.jdeveloper\product-preferences.xml* directory.

   If there is an existing entry "*<hash n = "oracle.ideimpl.externaltools.ExternalToolList">*, then replace with the contents in the *tools11g.xml* file. If not, add it right after the initial *<ide:preferences>* element.

9.1.3 Integrating with Oracle JDeveloper 10*g*

To configure Oracle JDeveloper to support the integration with Oracle Enterprise Repository, perform the following steps:

1. Navigate to the *Oracle_HOME\repositoryXXX\core\tools\solutions* directory and unzip the `11.1.1.x.x-OER-Harvester.zip` file to the Oracle JDeveloper directory. For example, if the jdeveloper.exe file is located in *C:\oracle\soa*, ensure that the introspector directory is unzipped into that directory.
2. Navigate to the *<jdeveloper_home>\harvester* directory and right-click the **tools.xml** file to open in a text editor.
3. Copy all the elements between the `<tools>` and `</tools>` elements and paste the copied elements into the tools.xml file in the `<jdeveloper_home>\jdev\system\oracle.jdeveloper.10.1.xxxxx` directory.
4. Save the `tools.xml` file in the `<jdeveloper_home>\jdev\system\oracle.jdeveloper.10.1.xxxxx` directory.
5. Start Oracle JDeveloper. In JDeveloper window, select **Tools**, **External Tools**. The following two options are displayed:
   - Submit this File into OER
   - Submit this Project into OER
6. Select **Submit this File into OER** and click **Edit**. The Edit External Tool dialog is displayed.
7. Click the **Properties** tab, and configure the missing properties to point to your Oracle Enterprise Repository server.
8. To point to an external HarvesterSettings.xml file, add a property called settings.file, and set the value to the URL of the settings file, for example, *c:\temp\MyHarvesterSettings.xml*.
9. Repeat Steps 6 to 8 for the Submit this Project into OER option.

10. In the Applications Navigator, select a file, and right-click and then select **Submit this File into OER** or **Submit this Project into OER**.

> **Note:** Oracle JDeveloper 10*g* does not support multi-select correctly for external tools. Even if you multi-select, only one file is harvested.

11. In the Edit External Tool dialog, click the **Process** tab.
12. Click **Change** to change the JDK version to 1.6, and then click **OK**.

> **Note:** Oracle Enterprise Repository browsing is not supported from Oracle JDeveloper 10*g*. To make 10g assets as consumable:
>
> - Harvest assets from JDeveloper 10*g*.
> - Reharvest from SOA runtime for just WSDLs.
>
> Also, the assets that are harvested from JDeveloper 10*g* or 11*g* cannot be used for consumption.

9.2 Configuring Eclipse

The Harvester integrates the Oracle SOA Suite artifacts to Oracle Enterprise Repository to support the visibility, impact analysis, and reusability use cases. This section describes the various steps involved in configuring Eclipse to support integration with Oracle Enterprise Repository:

- Section 9.2.1, "Enable Harvesting in Eclipse"
- Section 9.2.2, "Configure the Oracle Enterprise Repository Plug-ins"
- Section 9.2.3, "Configure the Oracle Enterprise Repository Preferences"
- Section 9.2.4, "Enable Automatic Usage Detection"

9.2.1 Enable Harvesting in Eclipse

This section describes how to harvest sample artifacts into Oracle Enterprise Repository using Eclipse:

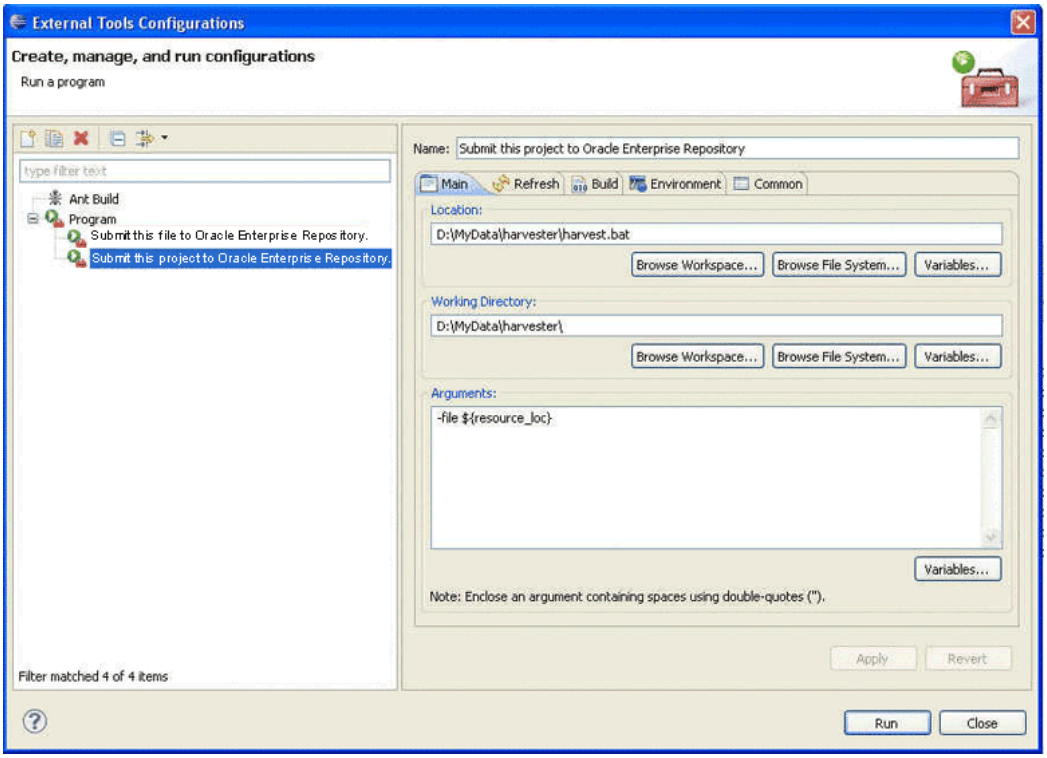
- Section 9.2.1.1, "Setting up Eclipse Environment to use Harvester as an "External Program""
- Section 9.2.1.2, "Harvesting in Eclipse Environment using "External Program""
- Section 9.2.1.3, "Setting up Eclipse Environment to use Harvester via ANT"
- Section 9.2.1.4, "Harvesting in Eclipse Environment using ANT"

9.2.1.1 Setting up Eclipse Environment to use Harvester as an "External Program"

1. In Eclipse, click **Run**, **External Tools**. The External Tools dialog is displayed.
2. Right-click **Program**, and then select **New**.
3. Enter the following details in the External Tools dialog, as shown in Figure 9–10.
   - In the Name field, type `Submit this project to Oracle Enterprise Repository`.

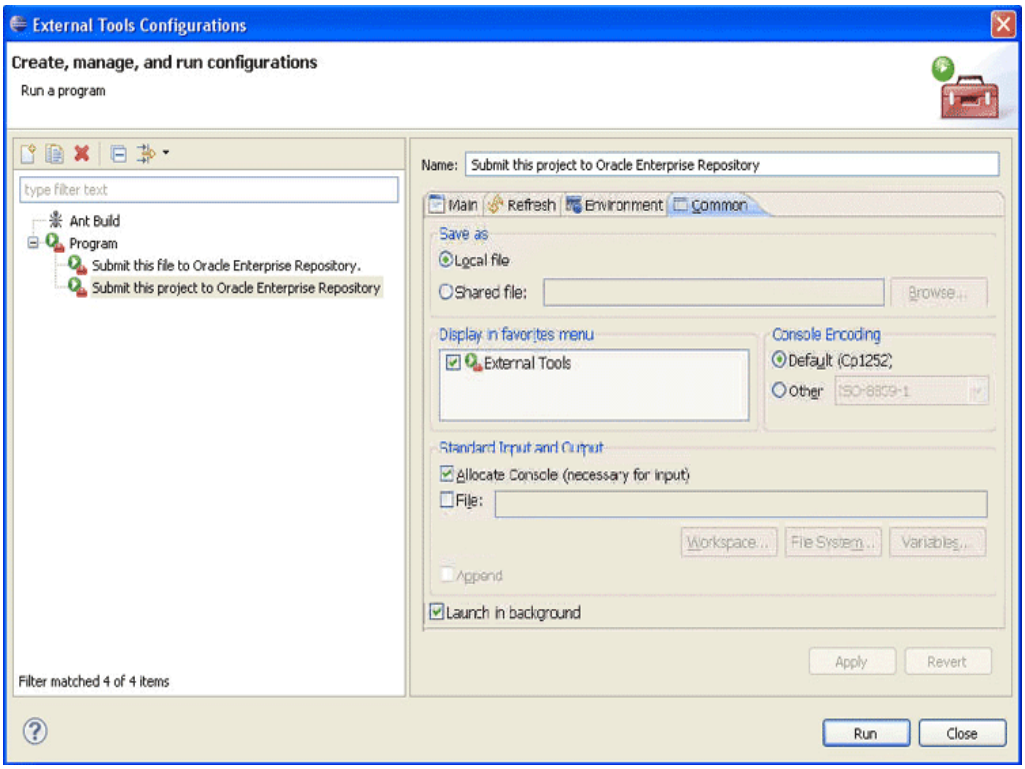
- In the Location field, type `<Harvester Home>\harvest.bat`. You can also browse the `harvest.bat` file using the **Browse** button.
- In the Working Directory field, type `<Harvester Home>`. You can also browse the working directory using the **Browse** button.
- In the **Arguments** field, type `-file ${project_loc}`.

Figure 9–10 External Tools Dialog



4. Click the **Common** tab.
5. In the Display in favorites menu pane, enable **External Tools**, as shown in Figure 9–11.

Figure 9–11 External Tools Dialog - Common Tab

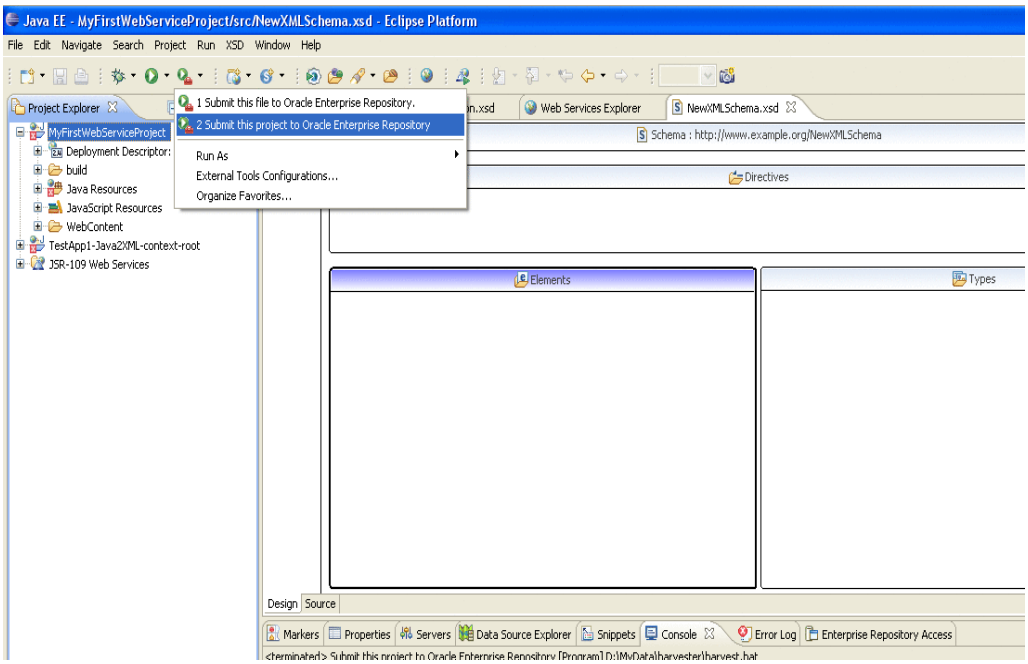


Note: You must install Harvester in the same computer where you have installed eclipse.

9.2.1.2 Harvesting in Eclipse Environment using "External Program"

1. In Eclipse, click **New**, **Project**, **General**, **Project** to create a new eclipse project.
2. Browse for any WSDL file in the file system, copy it and paste into the just created project.
3. Select the project and then click **Submit this project to Oracle Enterprise Repository**. This invokes the Harvester and submits all the artifacts in the project, as shown in Figure 9–12.

Figure 9–12 Java EE - Eclipse Platform



4. Repeat the above steps to create another program called Submit this file to Oracle Enterprise Repository by using `-file ${resource_loc}` instead of the ${project_loc} variable and to submit the individual files.

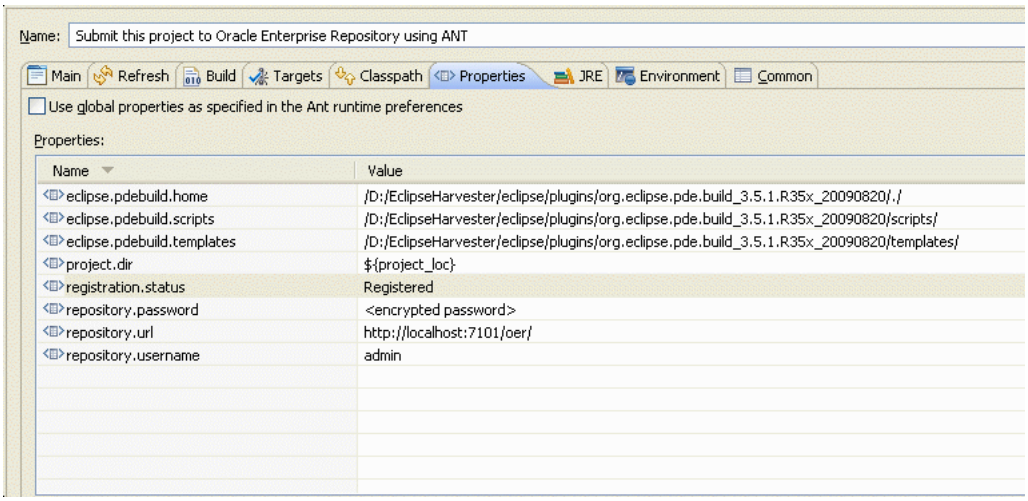
9.2.1.3 Setting up Eclipse Environment to use Harvester via ANT

1. In the Eclipse Workspace, select **Run**, **External Tools**, **External Tools Configurations**.
2. Create a new ANT build.
3. In the Main tab, enter the following:
   - **Buildfile**: `<Harvester_Home>\harvester-ant.xml`
   - **Base Directory**: `<Harvester_Home>`
4. In the Targets tab, check the `introspect-project` option.
5. In the Classpath tab, add the following external jars:
   - `<Harvester_Home>\lib\mail-1.4.jar`
   - `<Harvester_Home>\lib\activation-1.0.2.jar`
6. In the Properties tab, add the following properties/values
   - `project.dir/${project_loc}`
   - `registration.status/<"Unsubmitted", "Submitted - Pending Review", "Submitted - Under Review", or "Registered">`
   - `repository.url/http://localhost:7101/oer/`
   - `repository.username/admin`
   - `repository.password/<encrypted password>`
7. In the Common tab, enable the `External Tools` option.

8. Click **Apply**, and then click **Close**.

After the configuration of the Properties tab, the External Tools dialog is displayed, as shown in Figure 9–13.

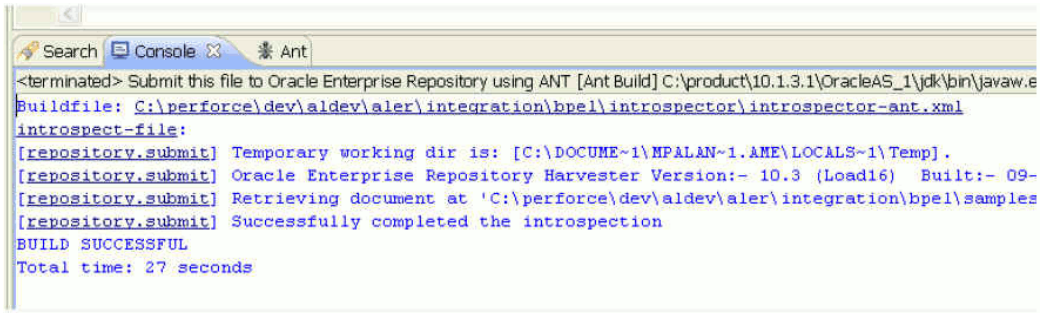
Figure 9–13 External Tools Dialog - Properties Tab



9.2.1.4 Harvesting in Eclipse Environment using ANT

1. In Eclipse, click **New**, **Project**, **General**, **Project** to create a new eclipse project.
2. Browse for any WSDL file in the file system, copy and paste it into the project that you just created.
3. Select the project and then click **Submit this file to Oracle Enterprise Repository using ANT**. This invokes Harvester and submits all the artifacts in the project, as shown in Figure 9–14.

Figure 9–14 The Console Window



9.2.2 Configure the Oracle Enterprise Repository Plug-ins

This section describes the steps to configure the Oracle Enterprise Repository Plug-ins for repository access and the prerequisities to enable this configuration. This section contains the following topics:

- Section 9.2.2.1, "Configuring the Oracle Enterprise Repository Plug-ins for Repository Access"
- Section 9.2.2.2, "Prerequisites for Using the Oracle Enterprise Repository Plug-ins for Eclipse"

9.2.2.1 Configuring the Oracle Enterprise Repository Plug-ins for Repository Access

For instructions on installing the Oracle Enterprise Repository plug-ins for repository access within the Eclipse-based Enterprise Repository Plug-in for Eclipse, see *Oracle Fusion Middleware Installation Guide for Oracle Enterprise Repository*.

Prerequisites for Using the Oracle Enterprise Repository Plug-in for Eclipse

Perform the following steps to install the Oracle Enterprise Repository plug-in for repository access:

1. Open Eclipse. The Eclipse window is displayed.
2. Click **Help**, **Install New Software**. The Install dialog is displayed.
3. Click the **Add** button that is found at the end of the Work With field. The Add Site dialog is displayed.
4. Enter Oracle Enterprise Repository in the Name field and *http(s)://<hostname>:<port>/<context>-web/eclipse* in the Location field.
5. Click **OK**.
6. Select **Oracle Enterprise Repository** and Click **Next**.
7. Review the items that must be installed.
8. Click **Next** and accept the agreement.
9. Click **Finish**.
10. In the Confirmation dialog, click **Yes** to restart Eclipse.

    You can verify if the Oracle Enterprise Repository plug-in was installed by clicking Windows, Preferences.

Uninstalling the Oracle Enterprise Repository Plug-ins

The Oracle Enterprise Repository plug-in for Eclipse can be uninstalled the same as any other Eclipse plug-in through the Eclipse software update menu.

Installing Products After Installing Oracle Enterprise Repository

If Oracle Service Bus applications are installed after the Oracle Enterprise Repository plug-in is installed, then Eclipse must be launched using the `-clean` flag.

9.2.2.2 Prerequisites for Using the Oracle Enterprise Repository Plug-ins for Eclipse

You should complete the prerequisites described in this section before using the Oracle Enterprise Repository plug-ins for Eclipse:

- Section 9.2.2.2.1, "Assign Users to an Oracle Enterprise Repository Project"
- Section 9.2.2.2.2, "Enabling the Assets-in-Progress Properties"
- Section 9.2.2.2.3, "SiteMinder"
- Section 9.2.2.2.4, "Java JDK"
- Section 9.2.2.2.5, "XML Parsing"

9.2.2.2.1 Assign Users to an Oracle Enterprise Repository Project

To download assets from the repository, users must be assigned to at least one Oracle Enterprise Repository project. An Oracle Enterprise Repository project administrator can assign users to projects using the Oracle Enterprise Repository Projects page.

Obtain the Eclipse integration path from the Oracle Enterprise Repository administrator. (For example, *http://appserver.example.com/oer-web/eclipse*).

9.2.2.2.2 Enabling the Assets-in-Progress Properties

Two system settings must be enabled to activate Assets-in-Progress when using the Oracle Enterprise Repository Plug-in for Eclipse.

This procedure is performed on the Oracle Enterprise Repository Admin screen.

1. Click **System Settings**.
2. Click **General Settings** in the System Settings section.
3. Enter the `cmee.asset.in-progress` property in the **Enable New System Setting** box and click **Enable** to reveal this hidden property.
4. Ensure the Asset in Progress property is set to `True`.
5. Click **Save**.
6. Enter the `cmee.asset.in-progress.visible` property in the **Enable New System Setting** box and click **Enable** to reveal this hidden property.
7. Ensure the Asset in Progress property is set to `True`.
8. Click **Save**.

The Registration Status list now appears in the Search section on the Oracle Enterprise Repository Assets screen. For more information about Assets-in-Progress, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

9.2.2.2.3 SiteMinder

If Oracle Enterprise Repository is or will be configured to be secured by Siteminder, the policy server must be configured to ignore (or unprotect) the following URL:

```
http://appserver.example.com:8080/oer-web/eclipse/
```

9.2.2.2.4 Java JDK

The Java Cryptography Extension (JCE) is required. It is provided in JDK v1.4, and is available as an optional package in JDK 1.2.x through 1.5.x. Note that Oracle Enterprise Repository plug-ins for use with Eclipse 3.x require JDK v 1.5.x or later.

9.2.2.2.5 XML Parsing

Since Editor and Viewer metadata is represented as CDATA-escaped XML, some XML parsers may exceed their entity expansion limit when communicating with Oracle Enterprise Repository. For example, if you have defined a large number of Asset Types in Oracle Enterprise Repository, then you may must increase the Entity Expansion Limit of your XML parser.

On some popular parsers, the default entity expansion limit is set to 64,000. This limit can be increased on JAXP-compliant processors by passing a command-line parameter called entityExpansionLimit. The entityExpansionLimit can be increased by passing a VM argument on the Eclipse command-line (modify the Eclipse desktop shortcut). For example, set the target of the shortcut to the following:

```
c:\eclipse\eclipse.exe -debug -consolelog -vmargs -DentityExpansionLimit=1048576
```

9.2.3 Configure the Oracle Enterprise Repository Preferences

This section describes the steps to configure the Oracle Enterprise Repository connection.

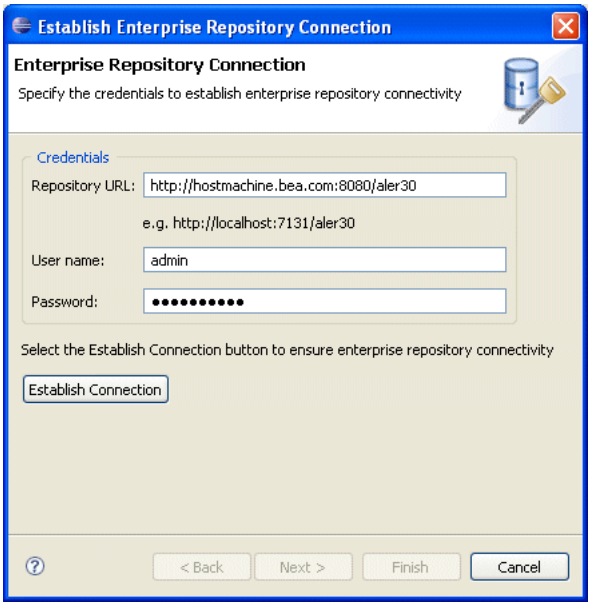
When you invoke an action on a repository, such as querying or publishing assets, before repository connectivity has been established, then the Connect to Enterprise Repository wizard is either automatically displayed (in the case of querying the repository), or is launched by an explicit user gesture.

Note: If credential information had been specified in a previous session, the wizard is displays this persisted information when it is launched.

---

1. In the Credentials area, as shown in Figure 9–15, enter the server location and login credentials, as follows:
   - **Repository URL**: the URL of the repository server. The URL must include the host, port, and Oracle Enterprise Repository server name, for example, *http://localhost:7001/oer*.
   - **User Name**: user name to gain access to the repository.
   - **Password**: password to gain access to the repository.

Figure 9–15 Establish Enterprise Repository Connection



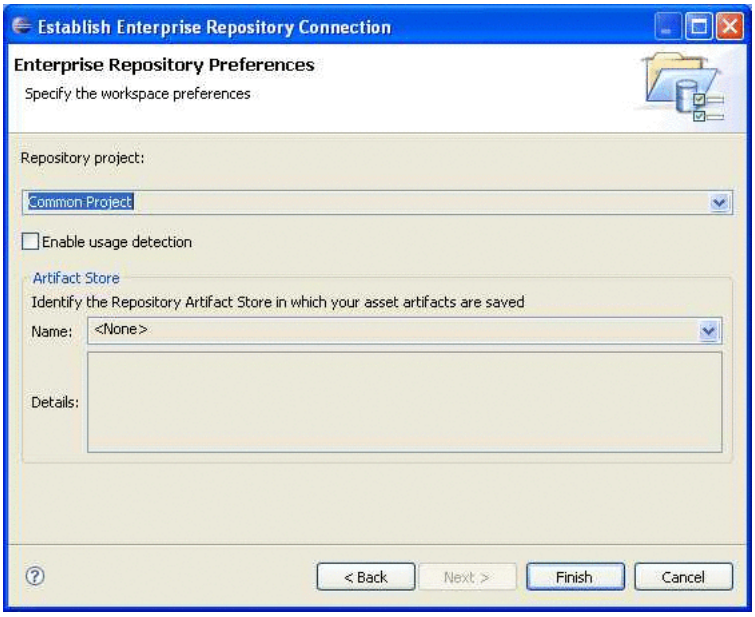
2. Click the **Establish Connection** button to ensure enterprise repository connectivity.

   If a connection cannot be established, then an appropriate error message is displayed.

3. In Figure 9–16, once connectivity is established, you can either:

- Click **Finish** to exit.
- Click **Next** to select your workspace preferences (skip to Step 4).

Figure 9–16 Specify Workspace Preferences



4. Once connectivity is established, you can specify your workspace preferences:
   - Select a Repository project in Oracle Enterprise Repository that the submitted model is associated with. Asset usage is tracked in the repository and attributed to repository projects, which typically represent software development programs, business initiatives, etc.
   - Enable usage detection: If you selected an Oracle Enterprise Repository project as the workspace default, usage detection is enabled for the default Oracle Enterprise Repository project. For more information about workspace preferences, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.
5. Click **Finish** to exit.

9.2.4 Enable Automatic Usage Detection

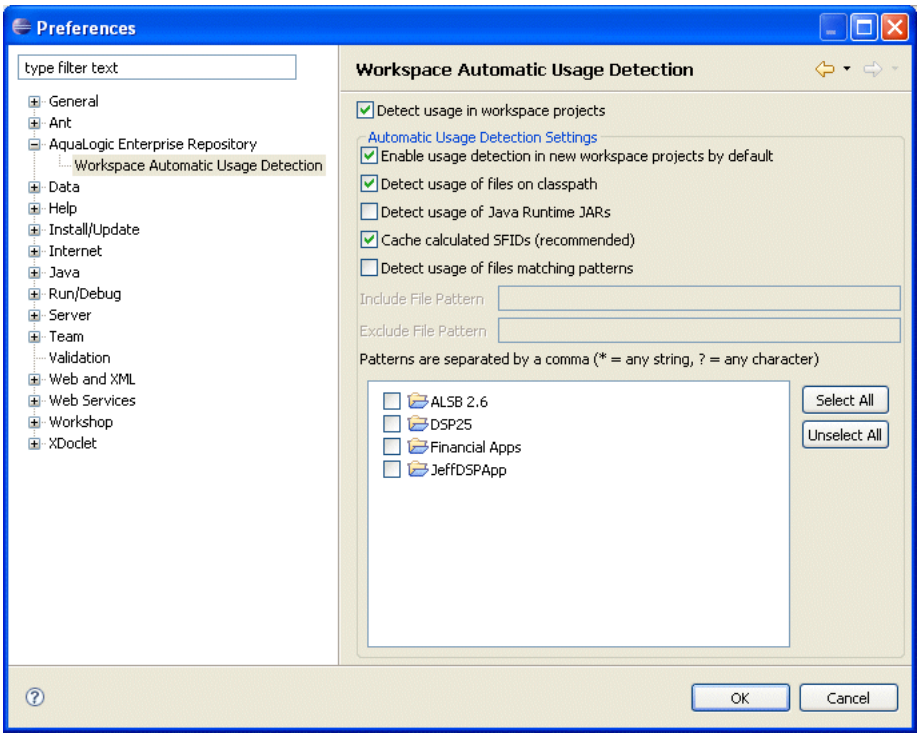
Oracle Enterprise Repository can automatically detect asset reuse within the development environment. This allows development teams to ensure that they get asset reuse credit, regardless of whether the assets have been downloaded through Oracle Enterprise Repository or pulled from another source, such as the developer's desktop. Automated Usage Detection relies on a fingerprinting process, called Software File Identification (SFID), which tags selected files within an asset with a unique ID. This SFID is then used to detect when and where an asset is used, even if the asset was acquired through means other than the Oracle Enterprise Repository Use - Download process. An instance of usage is recorded by Oracle Enterprise Repository when tagged files within the asset are brought into the developer's IDE, and a new build or build clean occurs.

For more information, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

Note: Automated Usage Detection requires the installation of the Oracle Enterprise Repository Plug-in for Eclipse, and is currently compatible only with Eclipse and Eclipse-based IDEs.

1. On the Window menu, click **Preferences**.
2. Select **Oracle Enterprise Repository**.
3. Select **Workspace Automatic Usage Detection**. The Workspace Automatic Usage Detection screen is displayed, as shown in Figure 9–17.

Figure 9–17 Preferences - Workspace Automatic Usage Detection



4. Click the **Detect usage in workspace projects** option, and then activate the desired usage detection features, as appropriate:
   - **Enable usage detection in new workspace projects by default** - monitors new projects
   - **Detect usage of files on classpath** - monitors files on classpath.
   - **Detect usage of Java Runtime JARs** - monitors Java Runtime JARs
   - **Cache calculated SFIDs (recommended)** - caches calculated SFIDs (enhances performance)
   - **Detect usage of files matching pattern** - monitors files matching specified patterns
5. Enter the appropriate information in the File Pattern text boxes:
   - **Include File Pattern** - Includes indicated file pattern
   - **Exclude File Pattern** - Excludes the indicated file pattern

6. Specify which project directories are targets for automatic usage detection by using the individual check boxes or by using the Select All and/or Unselect All buttons.
7. Click **OK** when finished.

9.3 Configuring VS .NET

Oracle Enterprise Repository integration with Visual Studio .NET provides users with the ability to easily search for and use assets from the repository without leaving the VS .NET IDE environment. Assets and any associated artifacts are downloaded directly to your VS .NET solution. Repository Access within the VS .NET solution also provides a view into Oracle Enterprise Repository that enables you to download artifacts and assets from the repository, query the repository, and view the contents of the repository.

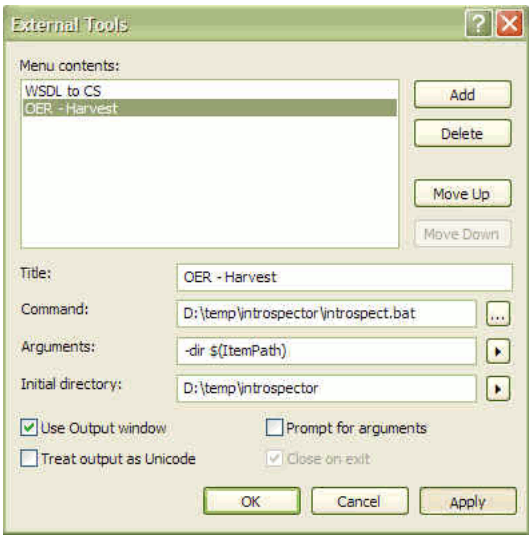
This section contains the following topics:

- Section 9.3.1, "Enable Harvesting in VS .NET"
- Section 9.3.2, "Configure the Oracle Enterprise Repository Plug-ins"
- Section 9.3.3, "Configure the Connection to Oracle Enterprise Repository"
- Section 9.3.4, "Assign an Oracle Enterprise Repository Project to a .NET Solution"
- Section 9.3.5, "Enable Automatic Usage Detection"

9.3.1 Enable Harvesting in VS .NET

1. In Microsoft Visual Studio, click **Tools**, **External Tools**. The External Tools dialog is displayed.
2. Click **Add**. A entry is added to the Menu Contents pane.
3. Enter the following details in the External Tools dialog, as shown in Figure 9–18:
   - In the Title field, type `OER - Harvest`.
   - In the Comman field, click the **Browse** button at the end of the field and select the harvest.bat file in the harvester directory.
   - In the Arguments field, type the `-dir` parameter. Click the right-arrow at the end of this field and select ItemPath from the menu.
   - In the Initial Directory field, type the location of the harvester directory.
   - Select the **Use Output Window** option. This option enables you to monitor progress.

Figure 9–18 External Tools Dialog



4. Click **OK**.
5. Select the WSDL file in the Microsoft Visual Studio and click **Tools**, **OER - Harvest**. The Output window is displayed with the Shutdown and Clean up messages indicating that the introspection is complete.
6. Open the Oracle Enterprise Repository home page with your username/password credentials.
7. In Assets pane, enter the name of the WSDL as the search criteria in the Enter Search String field, and then click **Search**. The search results are displayed in the right pane.
8. Select the service in the search results section, the details of the service are displayed in the bottom pane.
9. Click the **Navigator** button to view the relationships.
10. In the Oracle Enterprise Repository main page, click **Admin**, and then **System Settings**. The System Settings page is displayed.
11. Enter **Show** in the Search field, and set the Show System-Supplied Relationships option to `True`.
12. Click **Save** at the bottom of the page.
13. In the Oracle Enterprise Repository main page, click **Assets** and repeat the same search that you performed in step 7. The automatic relationships that were not imported earlier are now imported.

9.3.2 Configure the Oracle Enterprise Repository Plug-ins

Oracle Enterprise Repository can automatically detect asset reuse within the development environment. This allows development teams to ensure that they get asset reuse credit, regardless of whether the assets have been downloaded through Oracle Enterprise Repository. For more information, see Section 9.3.5, "Enable Automatic Usage Detection".

9.3.2.1 Prerequisities

To configure the Oracle Enterprise Repository plug-ins, you need the following prerequisites:

- Microsoft Visual Studio 2008.
- Microsoft Visual J# 2005 runtime. (If J# is not installed on your computer, the installer prompts you to download the correct version from Microsoft.)
- The VS .NET Always show solution option should be selected (**Tools** -> **Options** -> **Projects and Solutions** -> **General**).
- Users must be assigned to at least one Oracle Enterprise Repository project. A Project Administrator can assign users to projects using the Oracle Enterprise Repository Projects page.
- If your Oracle Enterprise Repository is or will be secured by Siteminder, then you must configure the policy server to ignore (or unprotect) the following URL to allow the OpenAPI integration to function properly:
  - *http://appserver.example.com:8080/OER/services/*

9.3.2.2 Installation

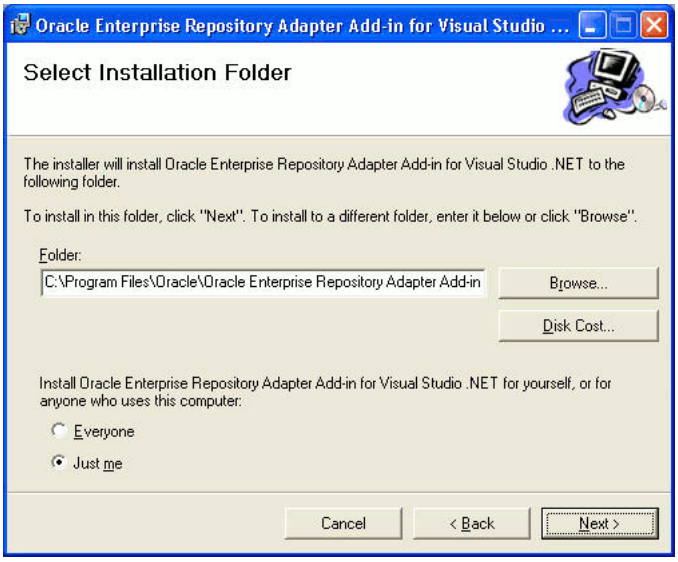
To install VS .NET plug-in, perform the following steps:

1. Download the VS .NET plug-in Zip file from your Oracle Enterprise Repository instance at the following URL:

   http://appserver.example.com/oer-web/integration/dotnet/OER103-VisualStudioAddin.zip

2. Unzip the **OER103-VisualStudioAddin.zip** file.
3. Locate and run the **setup.exe** program.
4. Follow the prompts, as shown in Figure 9–19, to select installation parameters.

Figure 9–19 Oracle Enterprise Repository Adapter Add-in for Visual Studio



5. Click **Finish** to complete the installation.

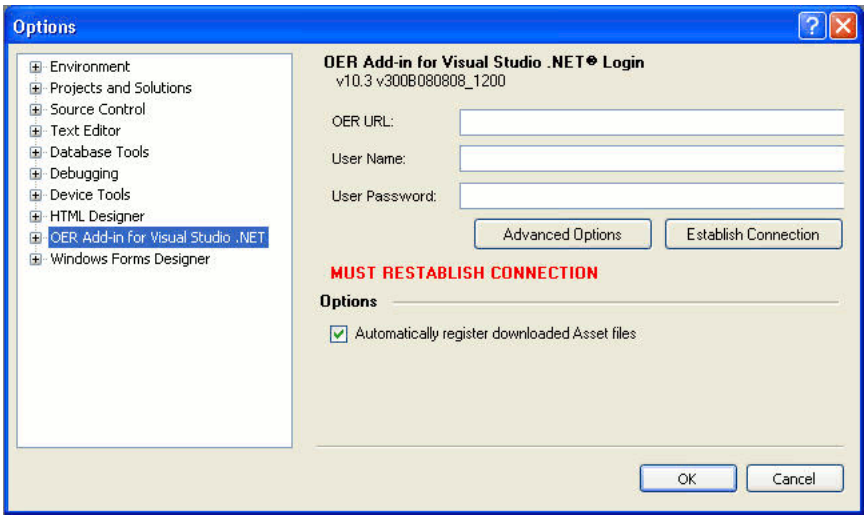
6. Follow the instructions in Section 9.3.3, "Configure the Connection to Oracle Enterprise Repository" to configure and establish a connection to an Oracle Enterprise Repository instance from VS. NET.

9.3.3 Configure the Connection to Oracle Enterprise Repository

Follow these steps to configure and establish a connection to an Oracle Enterprise Repository instance from VS. NET.

1. Launch Visual Studio .NET.
2. Open the Tools menu and click **Options**.
3. On the list of options, click the **OER Add-in for Visual Studio .NET** option, as shown in Figure 9–20, and provide the required login information.

Figure 9–20 The Options Dialog



- **OER URL**: The URL of the Oracle Enterprise Repository instance, for example, *http://appserver.example.com/OER*.

  Note: Do not include `index.jsp` used in the default home page as part of the URL.

- **User Name**: The user name to connect as.
- **User Password**: The password to connect with. Passwords are case-sensitive.
- **Establish Connection**: Click to verify a valid connection.
- **Automatically register downloaded Asset files**: If selected, downloaded asset files are registered with the Windows Registry, as appropriate. This may be overridden on an case-by-case basis for each asset download.

4. Click the **Establish Connection** button to connect to the Oracle Enterprise Repository instance you specified.
5. Optionally, click the **Advanced** button to enable additional Oracle Enterprise Repository options:
   - Usage detection for VS .NET Solution Projects

- Automated usage detection of referenced DLLs, WSDLs, and allow local caching of SFIDs (if SFID is enabled at your installation)
- File name patterns to include and exclude

6. Click **OK** when finished.

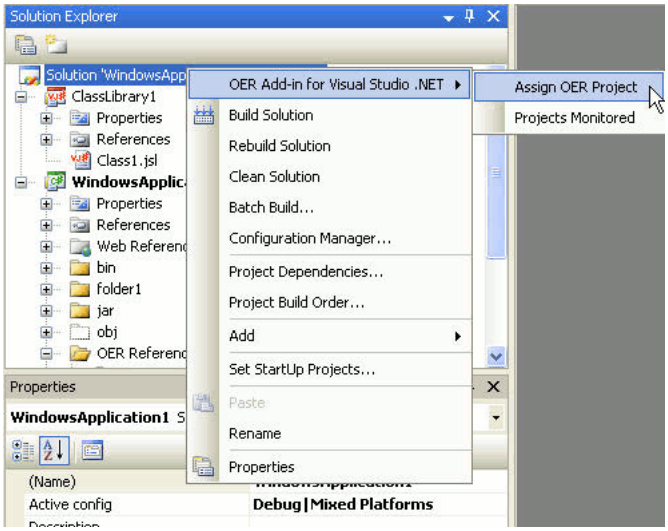
9.3.4 Assign an Oracle Enterprise Repository Project to a .NET Solution

To track the usage of downloaded assets, an Oracle Enterprise Repository project must be assigned to a .NET solution.

> **Note:** Before using this feature, you must be assigned to at least one Oracle Enterprise Repository Project by a Project Administrator.

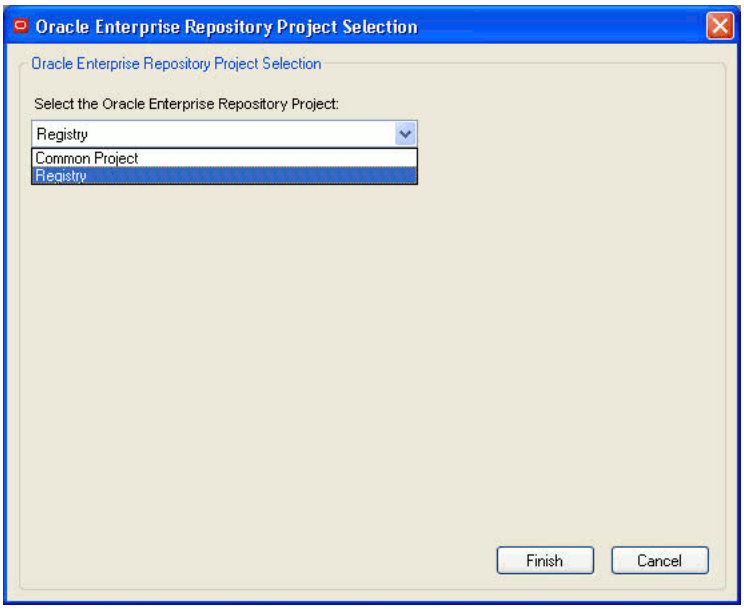
1. Open the .NET Solution Explorer.
2. Right-click a solution in the file tree and select the **Oracle Enterprise Repository Add-in for Visual Studio .NET** option from the context menu.
3. Click **Assign OER Project** from the submenu, as shown in Figure 9–21.

Figure 9–21 The Solution Explorer Window



4. In the Project Selection window, use the Select the Oracle Enterprise Repository Project list to view the Oracle Enterprise Repository projects that you are assigned to, as shown in Figure 9–22.

Figure 9–22 Oracle Enterprise Repository Project Selection Dialog



Note: If the list is empty, you have not been assigned to any projects and the procedure must be canceled.

5. Select an Oracle Enterprise Repository project from the list.
6. Click **Finish** to save your changes.

9.3.5 Enable Automatic Usage Detection

Follow these steps to enable advanced configuration options, such as enabling automatic usage detection of DLLs, WSDLs, local caching of SFIDs, and file pattern detection.

9.3.5.1 Overview of SFID

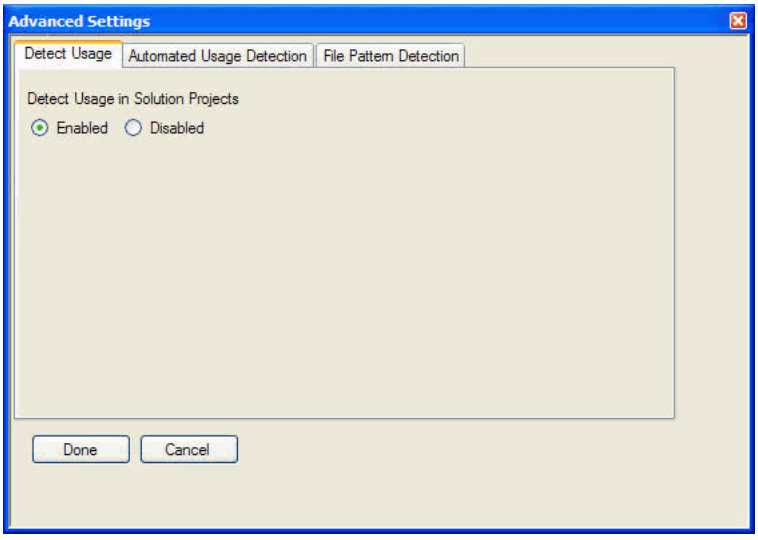
If SFID is enabled at your installation, Oracle Enterprise Repository can automatically detect asset reuse within the development environment. This allows development teams to ensure that they get asset reuse credit, regardless of whether the assets have been downloaded through Oracle Enterprise Repository. Automated Usage Detection relies on a fingerprinting process, called Software File Identification (SFID), which tags selected files within an asset with a unique ID. This SFID is then used to detect when and where an asset is used, even if the asset was acquired through means other than the Oracle Enterprise Repository Use - Download process. An instance of usage is recorded by Oracle Enterprise Repository when tagged files within the asset are brought into the developer's IDE, and a new build or build clean occurs.

9.3.5.2 Configuring Automatic Usage Detection

1. Launch Visual Studio .NET.
2. Open the **Tools** menu and click **Options**.
3. In the list of options, click **Oracle Enterprise Repository Add-in for Visual Studio .NET** to reopen the Login window.

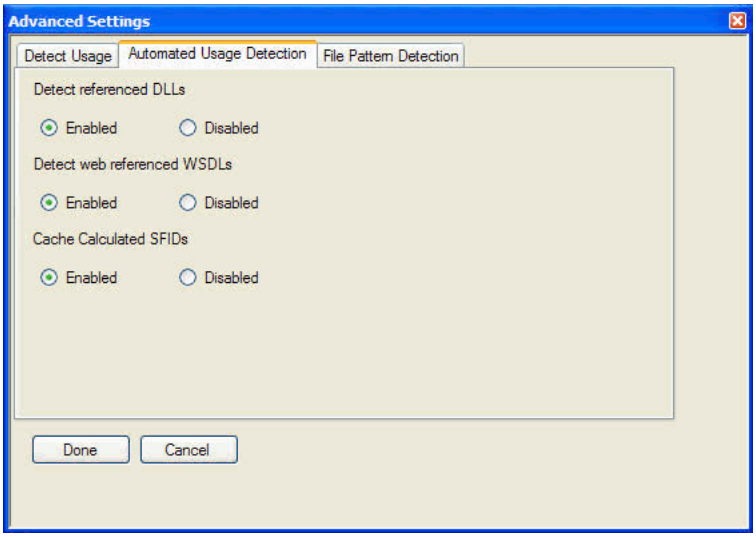
4. Click the **Advanced Options** button to open the Advanced Settings window. Use the Detect Usage tab to enable usage detection for VS .NET Solution Projects, as shown in Figure 9–23.

Figure 9–23 The Advanced Settings Dialog - Detect Usage Tab



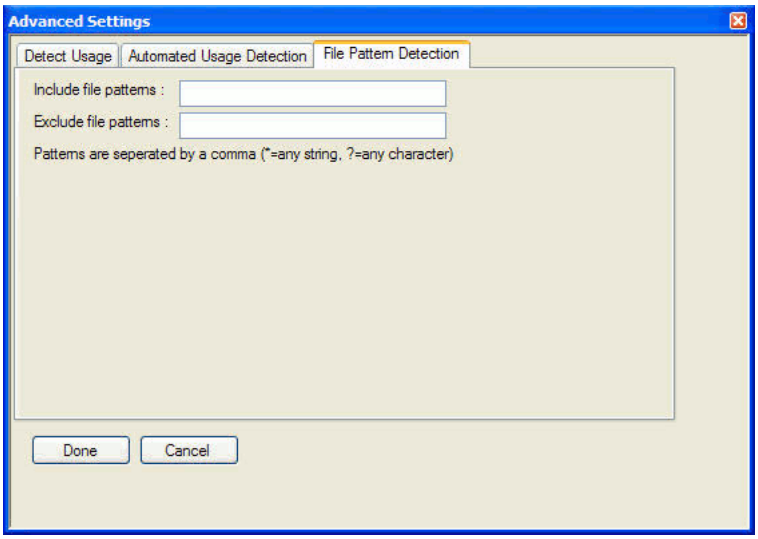
5. Click the **Automated Usage Detection** tab to enable usage detection of referenced DLLs, WSDLs, and allow local caching of SFIDs, as shown in Figure 9–24.

Figure 9–24 The Advanced Settings Dialog - Automated Usage Detection Tab



6. Click the **File Pattern Detection** tab to specify include and exclude file name patterns, as shown in Figure 9–25.

Figure 9–25 The Advanced Settings Dialog - File Pattern Detection Tab



7. Click **Done** to save your settings.

10

Using the IDE to Interact with Oracle Enterprise Repository

This chapter describes using the various IDEs to interact with Oracle Enterprise Repository. This chapter provides an overview of the production and consumption processes, and encompasses the development environment use cases. It contains the following topics:

- Section 10.1, "Using Oracle JDeveloper"
- Section 10.2, "Using Eclipse"
- Section 10.3, "Using VS .NET"

10.1 Using Oracle JDeveloper

The Oracle Enterprise Repository provides a flexible meta model for cataloguing all assets within the SOA ecosystem and their dependencies. It is primarily used during the plan, design, and build phase of the lifecycle as a single source of truth for service and composite application development.

Oracle SOA Suite provides a complete set of service infrastructure components for designing, deploying, and managing composite applications. Oracle SOA Suite enables services to be created, managed, and orchestrated into composite applications and business processes. Composites enable you to easily assemble multiple technology components into one SOA composite application. Oracle SOA Suite plugs into heterogeneous IT infrastructures and enables enterprises to incrementally adopt SOA.

You can use Oracle SOA Suite with the following versions of Oracle Enterprise Repository:

- Oracle Enterprise Repository 10.3 (with Oracle WebLogic Server 10.3)
- Oracle Enterprise Repository 11*g*

This section describes the following use cases:

- Section 10.1.1, "Harvest Artifacts"
- Section 10.1.2, "Search Oracle Enterprise Repository"
- Section 10.1.3, "View Asset Details"
- Section 10.1.4, "Download Artifacts"
- Section 10.1.5, "Prescriptive Reuse"

10.1.1 Harvest Artifacts

Oracle Enterprise Repository can harvest BPEL, WSDL, XSD, and XSLT files and file directories. After harvesting, Oracle Enterprise Repository automatically creates assets, populates asset metadata, and generates relationship links based on the information in the artifact files. The harvesting function is available from the command line, and can be integrated into Oracle JDeveloper or into the build process.

> **Note:** The Harvester is not restricted to Oracle products, it is used to harvest standards-based artifacts generated from any tooling.

You can publish or harvest a Oracle SOA Suite project to the Oracle Enterprise Repository either from the command-line or from Oracle JDeveloper, or using an Ant task. The Harvester harvests Oracle SOA Suite artifacts, including BPEL, WSDL, XSD and XSLT files and file directories, and automatically creates assets, populates asset metadata, and generates relationship links based on the information in the artifact files.

To publish a SOA project from Oracle JDeveloper 11*g* R1, perform the following steps:

1. In Oracle JDeveloper, right-click the **SOA** project and select **Harvest SOA Composite Project**. The Harvest SOA Project dialog is displayed.
2. Click **OK**. This runs an Antscript to harvest the SOA composite to Oracle Enterprise Repository.

For more information about harvesting artifacts from the JDeveloper 11*g* R2 version, see Section 9.1.1, "Integrating with Oracle JDeveloper 11g R1 Patchset Releases".

For more information about harvesting artifacts from the JDeveloper 10*g* version, see Section 9.1.3, "Integrating with Oracle JDeveloper 10g".

10.1.2 Search Oracle Enterprise Repository

You can access the assets and artifacts available in the Oracle Enterprise Repository through Oracle JDeveloper. Through Oracle JDeveloper, you can search for assets matching various criteria or view assets that may be of interest to a development project.

To search for assets in Oracle Enterprise Repository, perform the following steps:

1. In Oracle JDeveloper, click **Resource Palette**. The Resource Palette tab with the IDE Connections is displayed.
2. In the Search text field, enter the search criteria, for example, the name of the asset that you want to view the details for, and click **Start Search**. The Search Results pane is displayed with the assets.

10.1.3 View Asset Details

For selected assets, you can view asset details such as description, usage history, expected savings, and relationships. Within the asset metadata, links to the supporting documentation, user guides, test cases are provided to better enable you to reuse the existing functionality.

10.1.4 Download Artifacts

You can download an asset's artifacts (i.e., payload) into your project. Typically an asset payload is the functionality that you must use a service (such as a WSDL file) or incorporate into your code base (such as a binary or a BPEL file).

You can consume services, schemas, xslts and events from Oracle Enterprise Repository. To track the usage of an asset, you have to first associate a JDeveloper application with Oracle Enterprise Repository project. This section contains the following topics:

- Section 10.1.4.1, "Associating JDeveloper Application with Oracle Enterprise Repository"
- Section 10.1.4.2, "Consuming WSDL/Service from Oracle Enterprise Repository"

10.1.4.1 Associating JDeveloper Application with Oracle Enterprise Repository

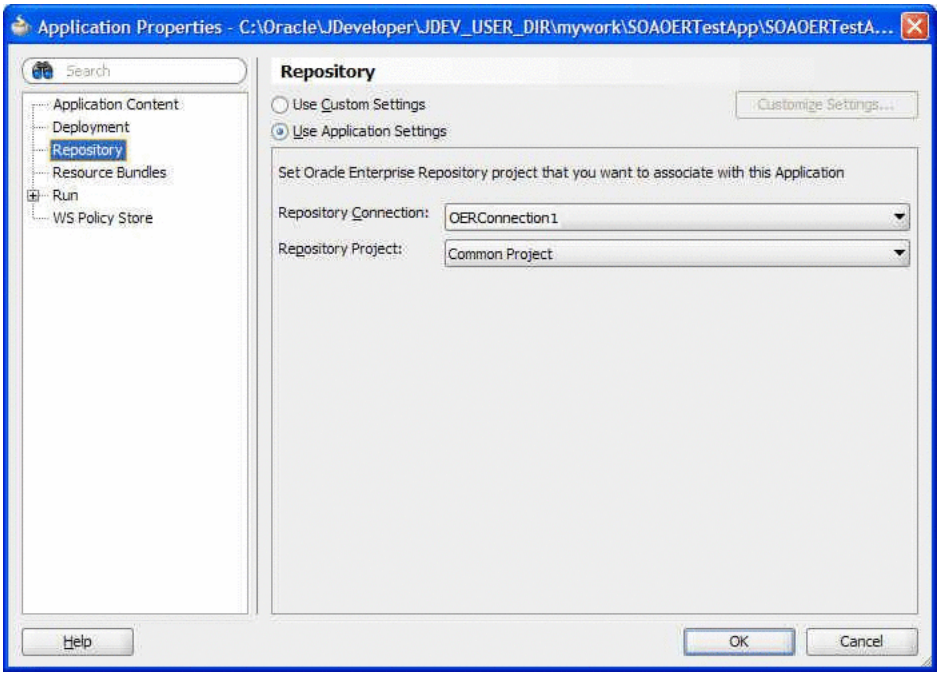
To associate JDeveloper application with Oracle Enterprise Repository project, perform the following steps:

1. In Oracle JDeveloper, click **Application**, and then **Application Properties**. The Application Properties dialog is displayed.
2. Select **Repository**. The Repository page is displayed.

> **Note:** To consume assets from Oracle Enterprise Repository within JDeveloper 11*g*, one has to configure application-level properties and add repository.

3. Select the following options, as shown in Figure 10–1.
   - Repository Connection: Select the Oracle Enterprise Repository connection that you want to use for usage tracking.
   - Repository Project: Select the Oracle Enterprise Repository project that you want to use for usage tracking.

Figure 10–1 Application Properties - Repository Page



4. Click **OK**.

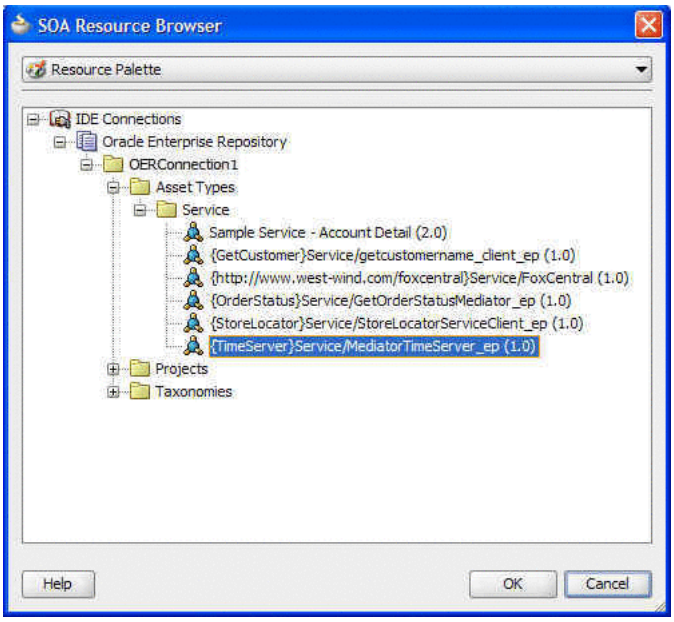
You can now consume assets from the connection that you have selected. And usage is added to the Oracle Enterprise Repository project that you selected.

10.1.4.2 Consuming WSDL/Service from Oracle Enterprise Repository

To consume a WSDL file or a service from Oracle Enterprise Repository, perform the following steps:

1. In Oracle JDeveloper, double-click the **composite.xml** file. The composite.xml page is displayed.
2. Drag and drop the Web Service component from the Component Palette to the External References swim lane. The Create Web Service dialog is displayed.
3. Click the **Finding Existing WSDLs** icon at the end of the WSDL URL field. The SOA Resource Browser dialog is displayed.
4. Select **Resource Palette** from the list and then select **IDE Connections**, **Oracle Enterprise Repository**, *<Connection Name>*, **Asset Types**, **Service**, as shown in Figure 10–2.

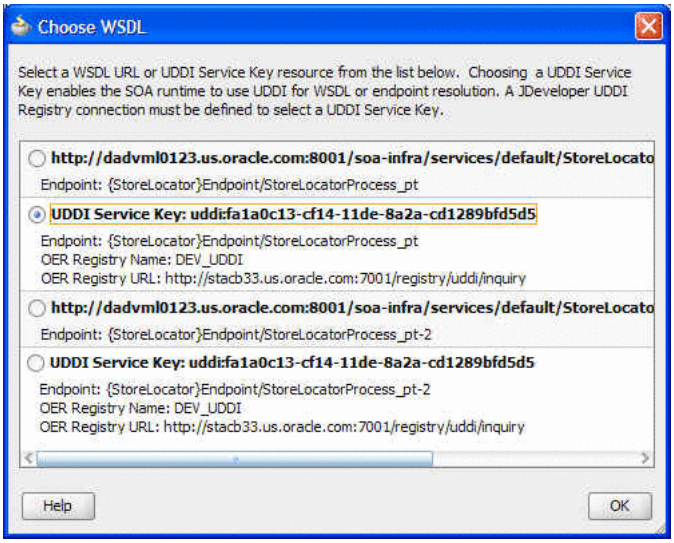
Figure 10–2 SOA Resource Browser Dialog



5. Choose the service that you want to invoke/consume.

   If the service has only one WSDL or UDDI key associated with it, then the same WSDL or UDDI key is used to create the reference. If service has more than one WSDL and/or UDDI key associated with it, then the Choose WSDL dialog is displayed, as shown in Figure 10–3.

Figure 10–3 Choose WSDL Dialog



   You must select an URLs/UDDI keys to consume. For resolving UDDI keys in JDeveloper, you have to create a UDDI connection prior to creating the reference, without which you can not select UDDI keys.

10.1.5 Prescriptive Reuse

Through the Oracle Enterprise Repository, analysts, architects, technical leads, and others who are involved in the design stages of a project can create a list of assets that fulfills a project's requirements. The list of assets are captured in compliance templates in the repository and the compliance templates are associated with an Oracle Enterprise Repository project.

From within the Oracle JDeveloper, you can view a list of assets appearing in all of the Compliance Templates assigned to your project. You can see which of the assets have been used and/or other project members. For more information about compliance templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

10.2 Using Eclipse

The Oracle Enterprise Repository plug-in for Eclipse development environment use cases are as follows:

- Section 10.2.1, "Submit Files"
- Section 10.2.2, "Harvest Artifacts"
- Section 10.2.3, "Search Oracle Enterprise Repository"
- Section 10.2.4, "View Asset Details"
- Section 10.2.5, "Download Artifacts"
- Section 10.2.6, "Prescriptive Reuse"
- Section 10.2.7, "Automatic Usage Detection"

10.2.1 Submit Files

The Oracle Enterprise Repository plug-in for Eclipse allows you to select files to submit to the Oracle Enterprise Repository. It packages the files into a .zip format for archive submission. The Archive Submission Wizard allows you to submit single and/or compound-payload assets to Oracle Enterprise Repository through an archive ZIP file.

To submit assets to Oracle Enterprise Repository through an archive ZIP file, perform the following steps:

1. Right-click an Eclipse project and select **Submit to Oracle Enterprise Repository** from the shortcut menu.
2. In the Archive Name field, enter the path to an existing project archive field or browse to an archive using the Ellipses button. You can also create a new archive file. When an existing file is selected, its fully qualified path is placed in the Archive Name text field. A valid project archive must have a .zip file extension.
3. When selecting an existing archive, click **OK** to confirm that it is okay to overwrite the selected project archive.
4. Use the resulting project folder structure to select at least one file from the project to submit to Oracle Enterprise Repository.
5. If necessary, you can click the **Select Types** button to open a dialog where can select certain file types to include in the archive.
6. After selecting the files you want to include in the archive, click **Next**. All artifacts selected from the project are zipped into the archive file.

7. After the archive and its contents have been specified, you can enter asset submission data, such as a version number, the type of asset to be submitted, a description, and associated comments.
8. Click **Finish**.
9. Click **OK** in the confirmation window to complete the submission process.

10.2.2 Harvest Artifacts

Oracle Enterprise Repository can harvest BPEL, WSDL, XSD, and XSLT files and file directories. After harvesting, Oracle Enterprise Repository automatically creates assets, populates asset metadata, and generates relationship links based on the information in the artifact files. The harvesting function is available from the command line, and can be integrated into Eclipse or into the build process.

To publish/harvest a project, perform the following steps:

1. In Eclipse, right-click the project and select **Submit this project to Oracle Enterprise Repository**. The Submit this Project to Oracle Enterprise Repository dialog is displayed.
2. Click **OK**. This runs a script to harvest the project to Oracle Enterprise Repository.

10.2.3 Search Oracle Enterprise Repository

You can access the assets and artifacts available in the Oracle Enterprise Repository through the Oracle Enterprise Repository plug-in for Eclipse. Through Eclipse, you can search for assets matching various criteria or view assets that may be of interest to a development project.

To search for the asset details, perform the following steps:

1. In Eclipse, click **Window**, **Show View**.
2. Select **Other**.
3. Select the **Enterprise Repository Access** option. The Enterprise Repository Access view is displayed as a tabbed pane containing Search and Project Team Assets panes.

   The Search tab enables querying of assets and displays results based upon specified criteria. The Search tab displays a toolbar at the top that is visible whether the active view is the Query pane or the Results pane. You can toggle between the two displays by clicking either the Query link or the Results link, depending on which pane is active at the time.

10.2.4 View Asset Details

For selected assets, you can view asset details such as description, usage history, expected savings, and relationships. Within the asset metadata, links to the supporting documentation, user guides, test cases are provided to better enable you to reuse the existing functionality.

To view the asset details, perform the following steps:

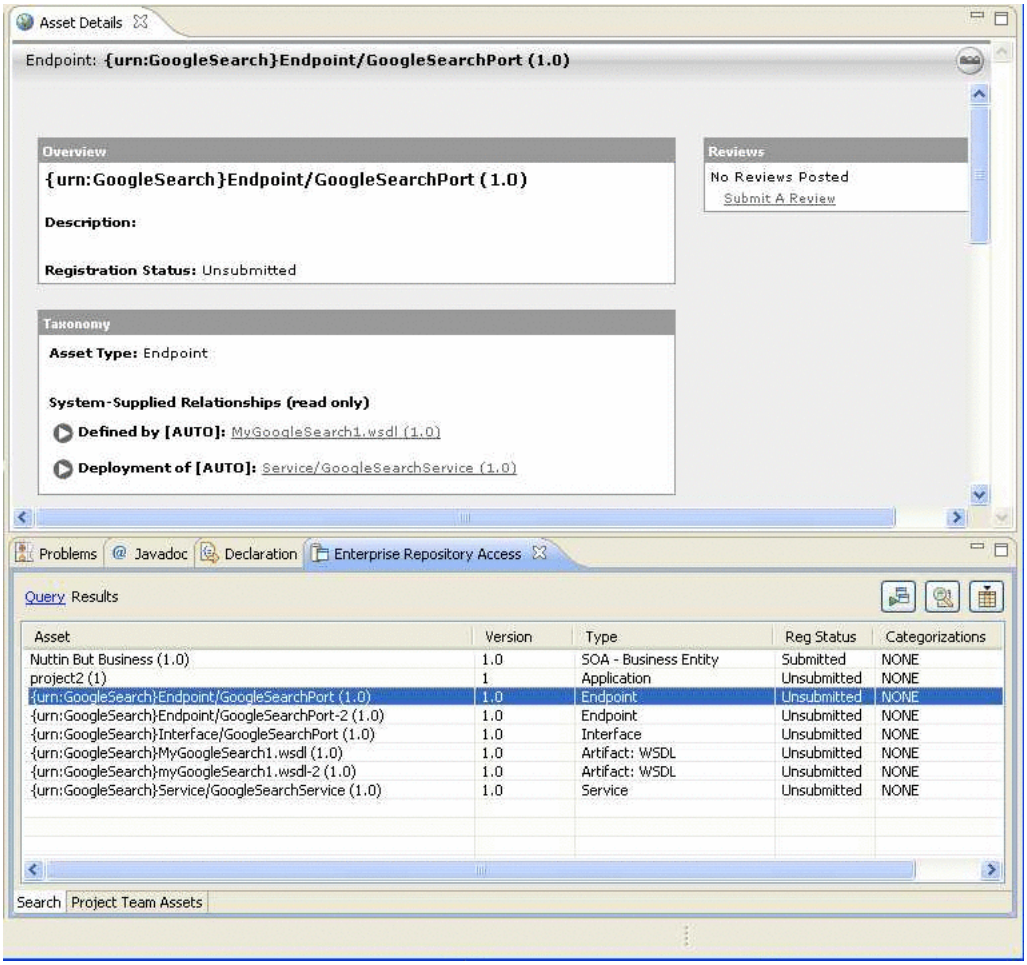
1. In Eclipse, click **Window**, **Show View**.
2. Select **Other**.

3. Select the **Enterprise Repository Access** option. The Enterprise Repository Access view is displayed as a tabbed pane containing Search and Project Team Assets panes.

   The Search tab enables querying of assets and displays results based upon specified criteria. The Search tab displays a toolbar at the top that is visible whether the active view is the Query pane or the Results pane. You can toggle between the two displays by clicking either the Query link or the Results link, depending on which pane is active at the time.

4. Right-click a search result and select **Show in Asset Details** View. The Asset Details view for the selected asset is displayed, as shown in Figure 10–4.

Figure 10–4 Asset Details Dialog



5. Select an asset in the search results and click the Display the asset properties and relationship views button to view the asset relationships. This Properties tab is displayed with the asset properties and the the Enterprise Repository Asset Relations tab is displayed with the relationships and relationship roles for the selected asset, as shown in Figure 10–5 and Figure 10–6 respectively.

Figure 10–5 Properties Tab

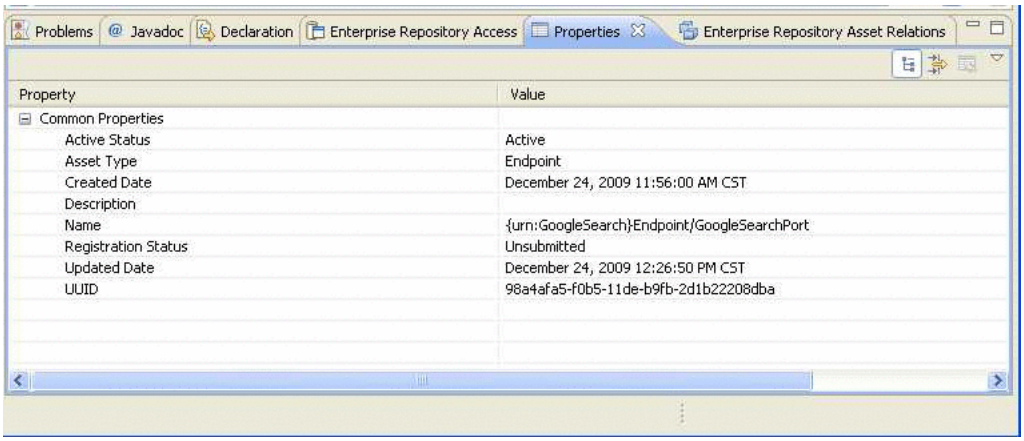
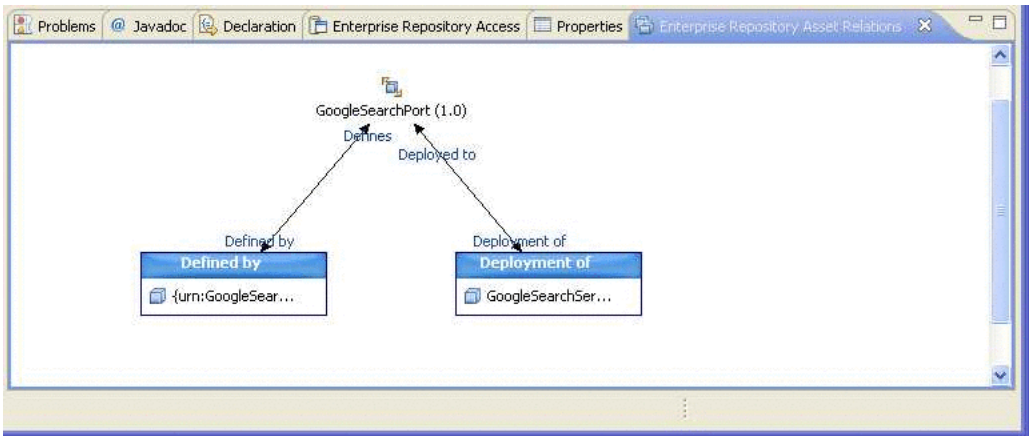


Figure 10–6 Enterprise Repository Access Relations Tab



10.2.5 Download Artifacts

You can download an asset's artifacts (i.e., payload) into your project. Typically an asset payload is the functionality that you must use a service (such as a WSDL file) or incorporate into your code base (usually a binary). To download artifacts, perform the following steps:

1. Query the repository for the desired asset(s), as described in Section 10.2.3, "Search Oracle Enterprise Repository".
2. Right-click the appropriate asset in the Results pane and if there are available artifacts, select **Download Artifacts** from the shortcut menu to open the Download Artifacts window.
3. In the Download Location section:
   - Use the **Download Folder** field to navigate to an Eclipse project and select the destination folder for the download.
   - Select the **Overwrite Existing Files** option to overwrite existing versions of the artifacts in the selected project folder.
4. In the Repository Governance section:
   - Select a valid project from the Repository Project list.

- Select the **Subscribe to Associated Assets** option to subscribe to all of the selected assets that had files associated with them, plus any associated artifact assets and dependencies.

5. Verify your selection in the list of artifacts to download, and then click **OK**. Artifacts associated with the selected asset are downloaded to the specified location.
6. Click **OK** in the status confirmation window.
7. Open the selected destination folder to confirm the presence of the selected artifact file(s).

10.2.6 Prescriptive Reuse

Through the Oracle Enterprise Repository, analysts, architects, technical leads, and others who are involved in the design stages of a project can create a list of assets that fulfills a project's requirements. The list of assets are captured in compliance templates in the repository and the compliance templates are associated with an Oracle Enterprise Repository project.

From within the Oracle Enterprise Repository plug-in for Eclipse, you can view a list of assets appearing in all of the Compliance Templates assigned to your project. You can see which of the assets have been used and/or other project members. For more information about compliance templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

For prescriptive reuse of assets, perform the following steps:

1. In Eclipse, click **Window**, **Show View**.
2. Select **Other**.
3. Select the **Enterprise Repository Access** option. The Enterprise Repository Access view is displayed as a tabbed pane containing Search and Project Team Assets panes.

   The Project Team Assets tab allows you to view all assets associated with a specified Oracle Enterprise Repository project.
4. From the Repository Project menu, select a project from the set of all enterprise repository projects associated with the current repository connection. The items in the list are prefixed with the name of the repository in which the associated project resides.
5. Click the **Query** button to query the enterprise repository for all assets associated with the specified repository project.

   If a repository connection has been established, the Project Team Assets table is populated with the set of assets associated with the specified project. Each of the columns in the table identifies how the asset relates to the project in the repository. You can also sort each column using the column header.
6. If necessary, click the **Refresh** button to repopulate the table with the results of this query.

10.2.7 Automatic Usage Detection

Oracle Enterprise Repository can automatically detect asset reuse within the development environment. This allows development teams to ensure that they get asset reuse credit, regardless of whether the assets have been downloaded through Oracle Enterprise Repository or pulled from another source, such as the developer's

desktop. Automated Usage Detection relies on a fingerprinting process, called Software File Identification (SFID), which tags selected files within an asset with a unique ID. This SFID is then used to detect when and where an asset is used, even if the asset was acquired through means other than the Oracle Enterprise Repository Use - Download process. An instance of usage is recorded by Oracle Enterprise Repository when tagged files within the asset are brought into the Oracle Enterprise Repository plug-in for Eclipse, and a new build or build clean occurs. For more information, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

To configure automatic usage detection, perform the following steps:

1. In Eclipse, click **Preferences** from the Window menu. The Preferences dialog is displayed.
2. Select **Oracle Enterprise Repository** and then **Workspace Automatic Usage Detection**. The Workspace Automatic Usage Detection page is displayed.
3. Click the **Detect Usage in Workspace Projects** option, and then activate the desired usage detection features, as appropriate:
   - Enable usage detection in new workspace projects by default - monitors new projects
   - Detect usage of files on classpath - monitors files on classpath
   - Detect usage of Java Runtime JARs - monitors Java Runtime JARs
   - Cache calculated SFIDs (recommended) - caches calculated SFIDs (enhances performance)
   - Detect usage of files matching pattern - monitors files matching specified patterns
4. Enter the appropriate information in the File Pattern text boxes:
   - Include File Pattern - Includes indicated file pattern
   - Exclude File Pattern - Excludes the indicated file pattern
5. Specify which project directories are targets for automatic usage detection by using the individual check boxes or by using the **Select All** and/or **Unselect All** buttons.
6. Click **OK** when finished.

10.3 Using VS .NET

The Oracle Enterprise Repository plug-in for VS .NET development environment use cases are as follows:

- Section 10.3.1, "Submit Files"
- Section 10.3.2, "Harvest Artifacts"
- Section 10.3.3, "Search Oracle Enterprise Repository"
- Section 10.3.4, "View Asset Details"
- Section 10.3.5, "Download Artifacts"
- Section 10.3.6, "Automatic Usage Detection"

10.3.1 Submit Files

The Oracle Enterprise Repository plug-in for VS .NET allows you to select files to submit to the Oracle Enterprise Repository. It packages the files into a .zip format for archive submission. The Archive Submission Wizard allows you to submit single and/or compound-payload assets to Oracle Enterprise Repository through an archive ZIP file.

10.3.2 Harvest Artifacts

Oracle Enterprise Repository can harvest BPEL, WSDL, XSD, and XSLT files and file directories. After harvesting, Oracle Enterprise Repository automatically creates assets, populates asset metadata, and generates relationship links based on the information in the artifact files. The harvesting function is available from the command line, and can be integrated into VS .NET or into the build process.

To publish/harvest a project, perform the following steps:

1. In VS .NET, right-click the project and select **Submit this project to Oracle Enterprise Repository**. The Submit this Project to Oracle Enterprise Repository dialog is displayed.
2. Click **OK**. This runs a script to harvest the project to Oracle Enterprise Repository.

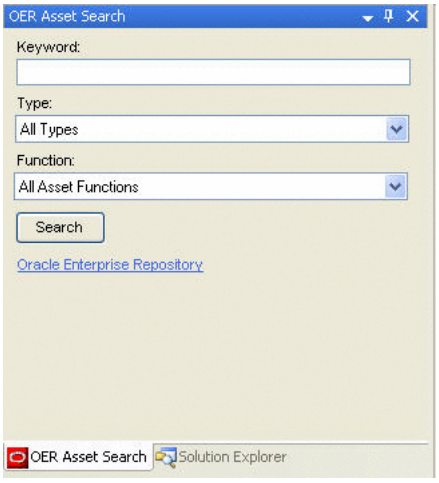
10.3.3 Search Oracle Enterprise Repository

You can access the assets and artifacts available in the Oracle Enterprise Repository through the Oracle Enterprise Repository plug-in for VS .NET. Through VS .NET, you can search for assets matching various criteria or view assets that may be of interest to a development project.

Perform a simple keyword search in VS .NET to locate an asset in Oracle Enterprise Repository to view the asset's metadata.

1. Open the View menu.
2. Click the **Oracle Enterprise Repository Add-in for Visual Studio .NET** option and then select **View Asset Search Window** from the submenu. The Oracle Enterprise Repository Asset Search window is displayed, as shown in Figure 10–7.

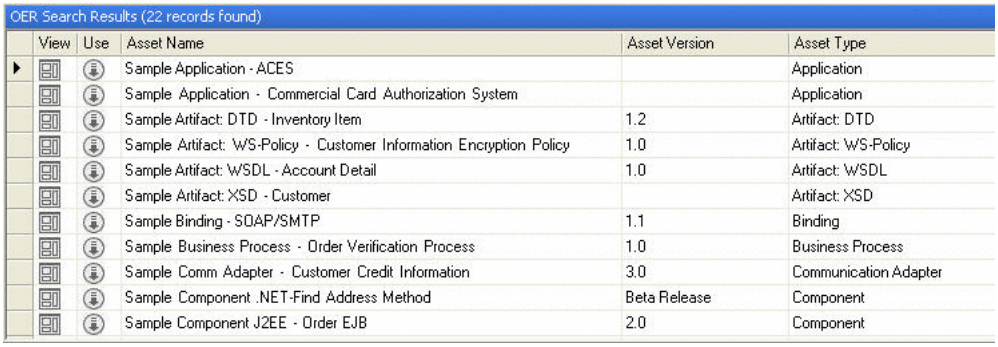
Figure 10–7 OER Asset Search Dialog



3. Type a keyword or phrase into the Keyword text box.

4. Use the Type and Function lists to refine the search.
5. Click the **Search** button. The Oracle Enterprise Repository Search Results window displays a list of all assets matching the search criteria, as shown in Figure 10–8.

Figure 10–8 OER Search Results Window



OER Search Results (22 records found)

View	Use	Asset Name	Asset Version	Asset Type
		Sample Application - ACES		Application
		Sample Application - Commercial Card Authorization System		Application
		Sample Artifact: DTD - Inventory Item	1.2	Artifact: DTD
		Sample Artifact: WS-Policy - Customer Information Encryption Policy	1.0	Artifact: WS-Policy
		Sample Artifact: WSDL - Account Detail	1.0	Artifact: WSDL
		Sample Artifact: XSD - Customer		Artifact: XSD
		Sample Binding - SOAP/SMTP	1.1	Binding
		Sample Business Process - Order Verification Process	1.0	Business Process
		Sample Comm Adapter - Customer Credit Information	3.0	Communication Adapter
		Sample Component .NET-Find Address Method	Beta Release	Component
		Sample Component J2EE - Order EJB	2.0	Component

6. Click the **Display Details** icon for any listed asset (or double-click the row) to view the asset's detail display.
7. Click the **Download** icon for any listed asset to download the asset.

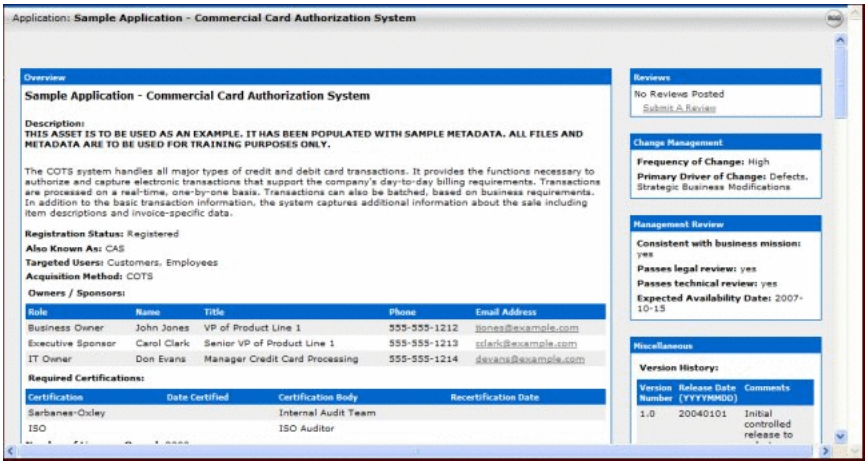
10.3.4 View Asset Details

For selected assets, you can view asset details such as description, usage history, expected savings, and relationships. Within the asset metadata, links to the supporting documentation, user guides, test cases are provided to better enable you to reuse the existing functionality.

The Asset Details view provides asset details for any listed asset in an embedded Web-based browser view, which calls out the enterprise repository application for details associated with the selected asset.

1. Perform a keyword search to locate an asset in Oracle Enterprise Repository, as described in Searching for Assets.
2. In the Oracle Enterprise Repository Search Results window, click the **Display Details** icon. Oracle Enterprise Repository opens to display information about the selected asset, as shown in Figure 10–9.

Figure 10–9 Oracle Enterprise Repository Overview Tab



This section contains the following topics:

- Section 10.3.4.1, "Accessing the Repository Assets Pane"
- Section 10.3.4.2, "Accessing the Oracle Enterprise Repository Log"

10.3.4.1 Accessing the Repository Assets Pane

The Repository Assets view displays a list of assets that have been prescribed to your project, as well as assets that are already in use in the project. This section contains the following topics:

- Section 10.3.4.1.1, "About Oracle Enterprise Repository Projects"
- Section 10.3.4.1.2, "Accessing the Repository Assets View"

10.3.4.1.1 About Oracle Enterprise Repository Projects

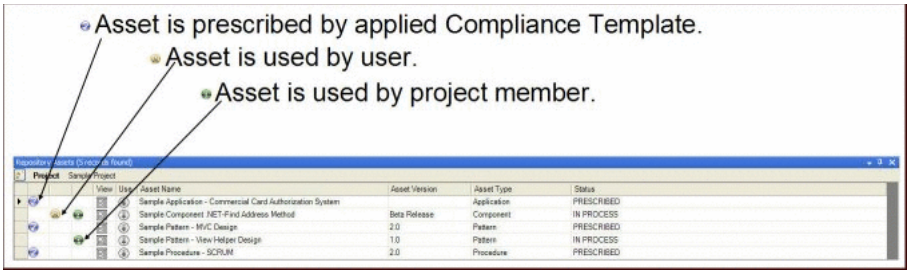
Through the Enterprise Repository, analysts, architects, technical leads, and others that are involved in the design stages of a project, can create a list of assets that might fulfill a project's requirements. The lists of assets are captured in compliance templates in the repository, and the compliance templates are associated with an Oracle Enterprise Repository project. For more information about compliance templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

From the Repository Assets view, you can view a list of assets appearing in all of the Compliance Templates assigned to your project. The viewer indicates which of the assets have been used by you and/or other project members. The viewer also displays other assets that are already in use in the project.

10.3.4.1.2 Accessing the Repository Assets View

1. Open the View menu.
2. Click the **Oracle Enterprise Repository Add-in for Visual Studio .NET** option and then select **Oracle Enterprise Repository Assets** from the submenu. The Repository Assets window is displayed, as shown in Figure 10–10.

Figure 10–10 Oracle Enterprise Repository Assets Window



3. Click the **Display Details** icon for any listed asset (or double-click the row) to view the asset's detail display.
4. Click the **Download** icon for any listed asset to download the asset.

10.3.4.2 Accessing the Oracle Enterprise Repository Log

To access the Oracle Enterprise Repository log:

1. Open the View menu.

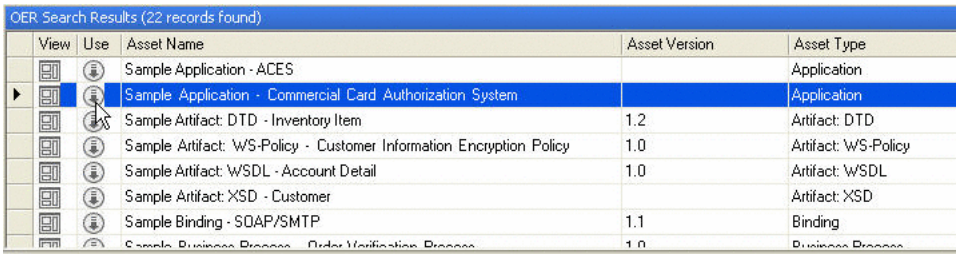
2. Click the **Oracle Enterprise Repository Add-in for Visual Studio .NET** option and then select **Oracle Enterprise Repository Log** from the submenu. The Oracle Enterprise Repository Log window is displayed.
3. Click **Clear** to remove the log entries.

10.3.5 Download Artifacts

You can download an asset's artifacts (i.e., payload) into your project. Typically an asset payload is the functionality that you must use a service (such as a WSDL file) or incorporate into your code base (usually a binary). Within the asset metadata, links to supporting documentation, user guides, test cases, etc., are provided to better enable developers to reuse existing functionality.

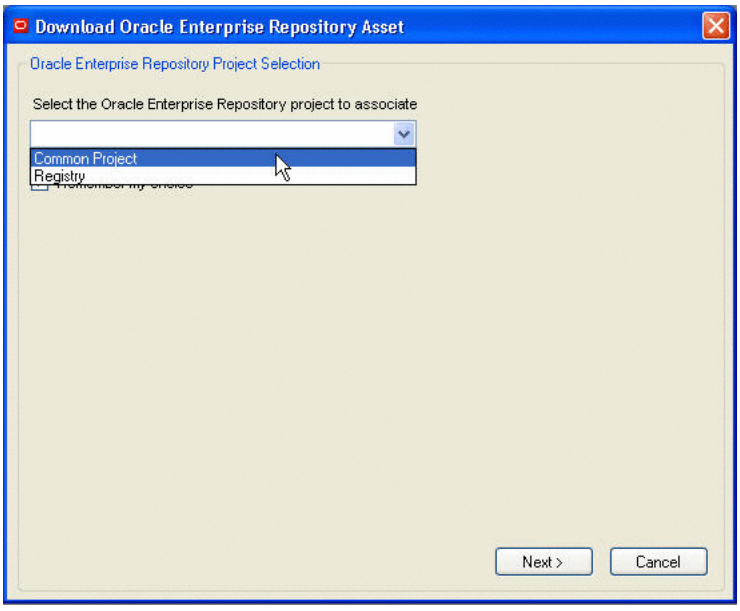
1. Perform a keyword search to locate an asset in Oracle Enterprise Repository, as described in Searching for Assets.
2. In the Oracle Enterprise Repository Search Results window, click the **Download** icon, as shown in Figure 10–11.

Figure 10–11 OER Search Results - Click Download



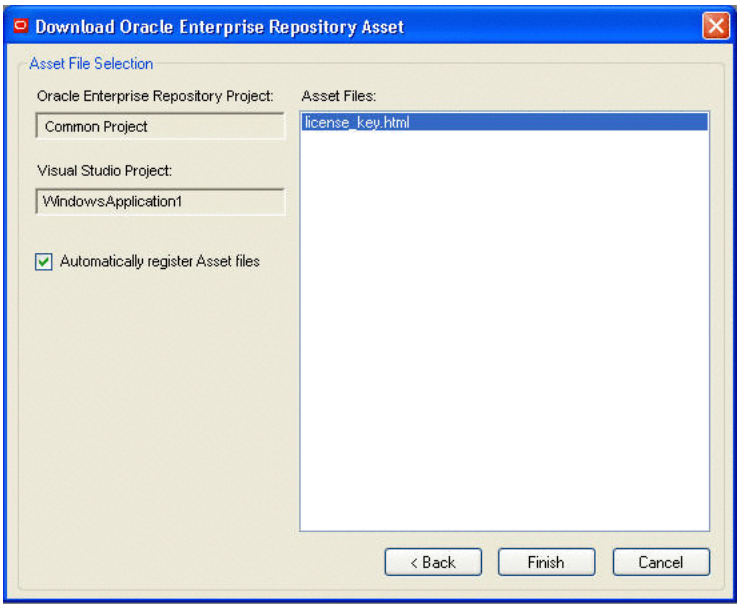
3. Select the VS project that you want to download the asset's files into, as shown in Figure 10–12, and then click **Next**.

Figure 10–12 Download Oracle Enterprise Repository Asset



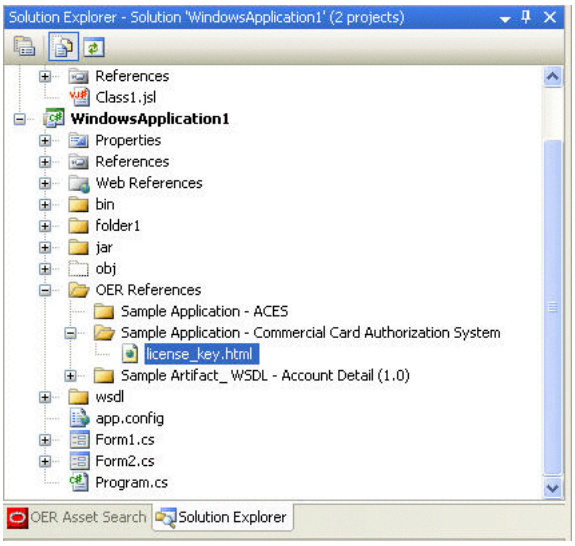
4. Select the asset files to download, as shown in Figure 10–13, and then click **Finish**.

Figure 10–13 Asset File Selection



5. If necessary, launch the Solution Explorer. A new folder labeled Oracle Enterprise Repository References is displayed in the project's file tree, as shown in Figure 10–14. This folder contains a subfolder bearing the name of the downloaded asset (for example, "xmd5 (1.0)" in the illustration below). This folder contains the asset's artifacts.

Figure 10–14 Solution Explorer Window



10.3.6 Automatic Usage Detection

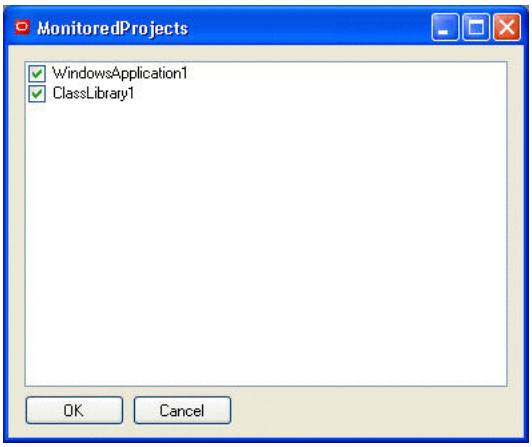
Oracle Enterprise Repository can automatically detect asset reuse within the development environment. This allows development teams to ensure that they get asset reuse credit, regardless of whether the assets have been downloaded through Oracle Enterprise Repository or pulled from another source, such as the developer's desktop. Automated Usage Detection relies on a fingerprinting process, called Software File Identification (SFID), which tags selected files within an asset with a

unique ID. This SFID is then used to detect when and where an asset is used, even if the asset was acquired through means other than the Oracle Enterprise Repository Use - Download process. An instance of usage is recorded by Oracle Enterprise Repository when tagged files within the asset are brought into the Oracle Enterprise Repository plug-in for VS .NET, and a new build or build clean occurs. For more information, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.

To automatically detect usage, a .NET project must be monitored.

1. Open the .NET Solution Explorer.
2. Right-click the solution in the file tree and select the Oracle Enterprise Repository Add-in for Visual Studio .NET option from the context menu.
3. Click **Projects Monitored** from the submenu. The Monitored Projects window is displayed.
4. Select the .NET projects that you want monitored for automated usage detection, as shown in Figure 10–15.

Figure 10–15 MonitoredProjects Dialog



5. Click **OK** when finished.

11 Integration with SAP

This chapter describes how to provide greater visibility into the services that are available from the SAP application and platform by integrating SAP with Oracle Enterprise Repository, from a SOA Governance perspective.

This chapter contains the following sections:

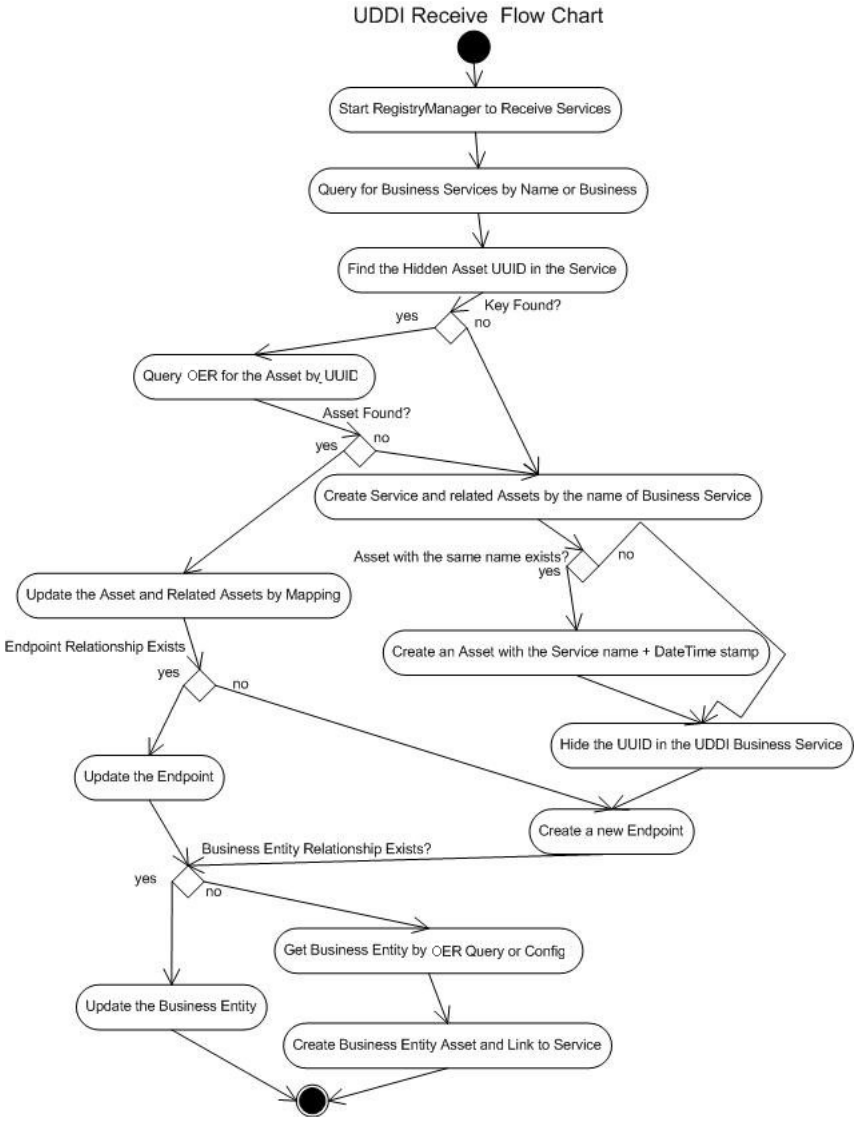
- Section 11.1, "Overview"
- Section 11.2, "Prerequisites"
- Section 11.3, "Configuring Integration with SAP"

11.1 Overview

A key goal of SOA Governance is to provide visibility and control across the different services across multiple packaged and custom applications built on different development platforms. SAP provides both the SAP packaged application as well as the NetWeaver middleware platform. The primary goal is to provide visibility into services that are available from the SAP application and platform. These services are consumed or reused as part of development of new composite applications, and used by Oracle Service Bus to generate proxy services for SAP business services.

SAP Enterprise Service Registry deploys the service into the SAP UDDI client, from where the Oracle Enterprise Repository-Exchange Utility connects and receives the service assets. Figure 11–1 describes the integration with the SAP repository and SAP Enterprise Registry.

Figure 11–1 SAP - OER Integration



Each time a new service is published in SAP, you can sync these to its corresponding service assets created in Oracle Enterprise Repository. You can also sync the ongoing service updates in SAP with the corresponding assets in Oracle Enterprise Repository. Then, you can consume the SAP assets like any other assets in Oracle Enterprise Repository.

11.2 Prerequisites

You must ensure that the following prerequisites are

- The SAP Respository and SAP UDDI Registry should be configured.
- The SAP Environment should be configured.

11.3 Configuring Integration with SAP

This section describes the detailed use cases for providing visibility in Oracle Enterprise Repository to services from the SAP application as well as the NetWeaver

platform, and allowing those services to be consumed through IDEs, including JDeveloper, Eclipse and VS .Net.

This section contains the following topics:

- Section 11.3.1, "Aligning the Taxonomy Between SAP and Oracle Enterprise Repository Assets"
- Section 11.3.2, "Exporting the Selected SAP Service Assets from SAP Enterprise Service Registry to OER"
- Section 11.3.3, "Receiving New Services from SAP"
- Section 11.3.4, "Configuring Ongoing Updates from SAP"
- Section 11.3.5, "End User Visibility of Available SAP Services"

11.3.1 Aligning the Taxonomy Between SAP and Oracle Enterprise Repository Assets

To align the taxonomy between SAP and Oracle Enterprise Repository assets:

1. Open the **UDDIMappings.xml** file.
2. Modify the tmodelkey attribute values under <tmodels>, as follows:

```
<tmodels>
<keyedReference tModelKey="uddi:bea.com:aleruuid" keyName="bea-com:aler:uuid"
/>
<keyedReference tModelKey="uddi:oracle.com:internalname"
keyName="oracle-com:oer:internalname" />
<keyedReference tModelKey="uddi:oracle.com:version"
keyName="oracle-com:oer:version" />
</tmodels>
```

3. Save and close the UDDIMappings.xml file.

11.3.2 Exporting the Selected SAP Service Assets from SAP Enterprise Service Registry to OER

To export the selected SAP service assets from SAP Enterprise Service Registry to Oracle Enterprise Repository, perform the following steps:

1. Bootstraping Oracle Enterprise Repository with services from SAP. This brings the initial load of SAP services into Oracle Enterprise Repository.
   a. Configure **orrxu.xml** for Oracle Enterprise Repository and SAP UDDI connection settings.
   b. Configure the XU receive query <registryQuery> as

```
<services>
   <service name="%"/>
</services>
```

   This would bring an initial load of all valid services available from SAP UDDI to Oracle Enterprise Repository. It should create service assets and its associated artifact assets in Oracle Enterprise Repository.

> **Note:** Currently, you can accomplish this by running the Exchange Utility *only* from the command prompt.

11.3.3 Receiving New Services from SAP

Each time a new service is deployed into SAP UDDI, it can be received into Oracle Enterprise Repository by running the Exchange Utility tool. Once an initial load of services is created in Oracle Enterprise Repository, the Exchange Utility can be run either periodically or each time a new service is published in SAP UDDI.

If you know the service name that is created in SAP, then you can configure the receive query, `<registryQuery>`, to provide service name to receive the service. Alternatively, you can run the query as `service name="%"` to create new service assets in Oracle Enterprise Repository and also re-sync the existing services in Oracle Enterprise Repository with SAP.

11.3.4 Configuring Ongoing Updates from SAP

Each time a service is modified in the SAP UDDI, the updates need to be synced to its corresponding service assets created in Oracle Enterprise Repository. You can achieve this by running the Exchange Utility from the command-line.

You can periodically run Exchange Utility by using the wild character query for `<registryQuery>` as `service name="%"` to re-sync the existing services in Oracle Enterprise Repository with SAP and create new service assets in Oracle Enterprise Repository.

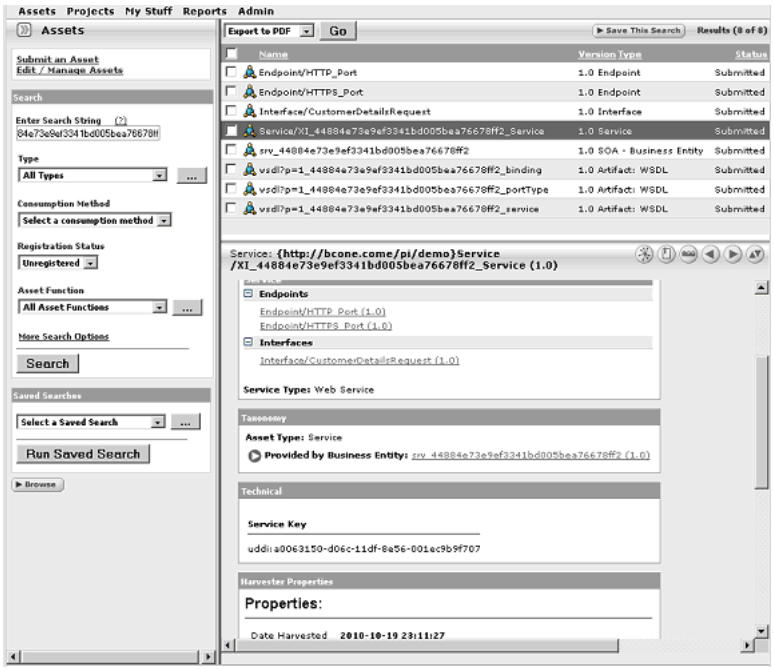
Note: Currently, you can accomplish this by running the Exchange Utility *only* from the command prompt.

11.3.5 End User Visibility of Available SAP Services

Once the SAP services are in Oracle Enterprise Repository, you can search and consume the SAP services from the IDE and consume SAP assets from Oracle Enterprise Repository Web console like any other asset in Oracle Enterprise Repository.

Figure 11–2 describes the image of a SAP asset as it appears in Oracle Enterprise Repository.

Figure 11–2 SAP Asset in Oracle Enterprise Repository



For more information about searching and consuming services from the IDE, see Section 10.1, "Using Oracle JDeveloper".

Part IV

Developing Custom Integrations

This part describes how to get started with the Extensibility Framework (REX) and use the various APIs that REX provides.

This part contains the following chapters:

- Chapter 12, "Repository Extensibility Framework"
- Chapter 13, "ArtifactStore API"
- Chapter 14, "AcceptableValueLists API"
- Chapter 15, "Asset API"
- Chapter 16, "AssetType API"
- Chapter 17, "Categorization Types and Categorizations API"
- Chapter 18, "CMF Entry Type API"
- Chapter 19, "Custom Access Settings API"
- Chapter 20, "Department API"
- Chapter 21, "Extraction API"
- Chapter 22, "Localization of REX Clients"
- Chapter 23, "Notification API"
- Chapter 24, "Policy API"
- Chapter 25, "Projects API"
- Chapter 26, "Relationship Types API"
- Chapter 27, "Role API"
- Chapter 28, "Subscriptions API"
- Chapter 29, "System Settings API"
- Chapter 30, "User API"
- Chapter 31, "Vendor API"

12

Repository Extensibility Framework

This chapter provides an overview of Repository Extensibility Framework (REX).

This chapter contains the following sections:

- Section 12.1, "Introduction to REX"
- Section 12.2, "REX Architecture"
- Section 12.3, "Basic Concepts"

> **Note:** As a result of Oracle's acquisition of BEA/Flashline, Inc., the product Flashline Registry has been rebranded as Oracle Enterprise Repository. The bulk of the REX documentation has been revised to reflect this change. However, to avoid confusion, certain instances of the words "flashline" and "registry" are unchanged in this documentation, particularly where they appear in the java package structure and in the REX class names.

12.1 Introduction to REX

REX is a Web Services API for programmatic integration into Oracle Enterprise Repository. It is based on accepted industry standards, and designed with a focus on interoperability and platform independence. REX uses Remote Procedure Call (RPC) Web Services described by the Web Services Description Language (WSDL v1.1). This allows clients to interact with Oracle Enterprise Repository using any platform and any implementation language that supports Web Services. For example, while Oracle Enterprise Repository is a J2EE application, REX allows programmatic interaction with a .NET client.

If your Oracle Enterprise Repository is or will be configured to be secured by Siteminder, you must configure the policy server to ignore (or unprotect) the following URL to allow OpenAPI integration to function properly:

http://appserver.example.com/oer/services/

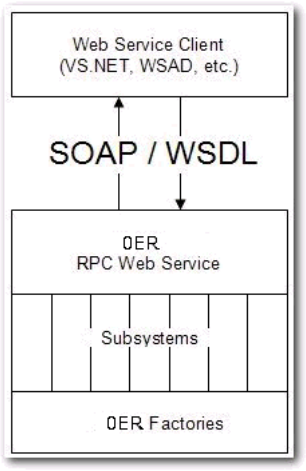
> **Note:** Every example provided in the documentation is provided for illustrative purposes and is necessarily compile due to package structure changes between versions of the Oracle Enterprise Repository WSDL. The user is expected to change the package structure to appropriately target their version of Oracle Enterprise Repository. Full sample code for your version of Oracle Enterprise Repository is provided as part of the install media.

For more information about REX, see the REX Javadoc at *Oracle Fusion Middleware Extensibility Framework (REX) for Oracle Enterprise Repository*.

12.2 REX Architecture

Figure 12–1 describes the REX architecture.

Figure 12–1 REX Architecture



The high-level architecture of Oracle Enterprise Repository and REX is designed with several high-level goals in mind:

- **Flexibility**

  Any client platform that conforms to accepted industry standards, such as SOAP, WSDL, and HTTP, can interact with Oracle Enterprise Repository through the REX interface. Proper functioning of the API with the most common client platforms has been validated.

- **Extensibility**

  Oracle Enterprise Repository's layered architecture simplifies the process of adding subsystems to provide access to new features as they are added to Oracle Enterprise Repository. For more information, see Section 12.2.4, "Versioning Considerations for the Oracle Enterprise Repository REX".

- **Simplicity**

  End users find it easy to take advantage of the extensive feature set available in REX.

This section contains the following topics:

- Section 12.2.1, "Subsystems Overview"
- Section 12.2.2, "CRUD-Q Naming Convention"
- Section 12.2.3, "Fundamental WSDL Data Types"
- Section 12.2.4, "Versioning Considerations for the Oracle Enterprise Repository REX"

12.2.1 Subsystems Overview

Oracle Enterprise Repository's REX provides access to a variety of subsystems. These subsystems loosely group system functionality into logical categories roughly equivalent to the type of entity on which they operate. Much of this document is organized into sections related to these subsystems.

REX methods are named using a scheme based on the various subsystems. For more information about the description of the algorithm used in this process, see Section 12.2.2, "CRUD-Q Naming Convention". The subsystems defined in REX include:

- acceptableValue
- asset
- assetType
- authToken
- categorization
- catergorizationType
- department
- extraction
- import/export
- project
- relationship
- role
- user
- vendor

12.2.2 CRUD-Q Naming Convention

The scheme used in naming the Open API methods is based on the CRUD-Q mnemonic. CRUD-Q represents five operations:

- C - Create
- R - Read
- U - Update
- D - Delete
- Q - Query

Each method starts with the name of the subsystem to which it belongs, followed by a description of the operation to be performed within that subsystem, as in the following example:

<subsystem><Operation>

For example, the method to perform a create operation in the asset subsystem would be:

```
assetCreate(...)
```

This naming convention would also produce:

```
assetRead(...)
assetUpdate(...)
assetDelete(...)
assetQuery(...)
```

Subsystems are likely to have operations beyond the CRUD-Q set, and may not include all of CRUD-Q. For example, since it is impossible to delete a user, there is no userDelete method. There is, however, a userDeactivate method. Table 12–1provides a detailed list of the detailed operations that the subsystem can have apart from the CRUD-Q operations.

Table 12–1 Subsytems and CRUD-Q Convention Relationship

	Create	Read	Update	Delete	Query	Other Features
Acceptable Value List	Yes	Yes	Yes	Yes	Yes	Accept, Activate, Assign, Deactivate, Register, Retire, Submit, Unaccept, Unassign, Unregister, Unsubmit, Modify Custom Access Settings
Asset	Yes	Yes	Yes	Yes	Yes	
Asset Type	Yes	Yes	Yes	Yes	Yes	
Categorization Type	Yes	Yes	Yes	Yes	Yes	
Department	Yes	Yes	Yes	No	Yes	
Extraction	Yes	Yes	Yes	No	Yes	
Project	Yes	Yes	Yes	Yes	Yes	Close, Open, Reassign extractions,Remove user
Relationship	Yes	Yes	Yes	No	Yes	
Role	Yes	Yes	Yes	Yes	Yes	
User	Yes	Yes	Yes	No	Yes	Activate, Deactivate, Lockout,Unapprove
Vendor	Yes	Yes	Yes	Yes	Yes	
Contact	Yes	Yes	Yes	Yes	Yes	

12.2.2.1 Atomicity of Method Calls

Unless otherwise noted, every call to REX is atomic. That is, each call either succeeds completely, or fails completely.

For example, one version of the `categorizationUpdate` method takes as an argument an array of categorization updates. In this case, if one categorization update fails, all categorization updates fail.

12.2.2.2 No Inter-call Transaction Support

REX does not currently support inter-call transactions. For example, in the event of an error it is impossible to roll back operations associated with a series of REX calls.

12.2.3 Fundamental WSDL Data Types

REX uses the following fundamental WSDL data types, in addition to the complex types defined in the WSDL.

Arrays of any of these types may be returned:

- xsd:int
- xsd:long
- xsd:string
- xsd:boolean
- xsd:dateTime

You can dynamically generate API Stubs by consuming the REX WSDL by pointing their IDEs or Web services toolkits at the following URL:

http://appserver/oer/services/FlashlineRegistry?WSDL

For example, Java stubs for the Oracle Enterprise Repository REX WSDL may be created using the AXIS WSDL2java utility:

```
java -cp .;axis.jar; xerces.jar; commons-discovery.jar;
commons-logging.jar; jaxen-full.jar; jaxrpc.jar;
saaj.jar; wsdl4j.jar; xalan.jar org.apache.axis.wsdl.WSDL2Java
```

http://appserver/oer/services/?FlashlineRegistry?WSDL

The JAR files required to complete this conversion process are:

- `axis.jar`
- `xerces.jar`
- `commons-discovery.jar`
- `commons-logging.jar`
- `jaxen-full.jar`
- `jaxrpc.jar`
- `saaj.jar`
- `wsdl4j.jar`
- `xalan.jar`

Note: Replace "appserver" in the URL with the name of the server on which Oracle Enterprise Repository is installed.

12.2.4 Versioning Considerations for the Oracle Enterprise Repository REX

Naturally, the evolution of the Oracle Enterprise Repository REX parallels the evolution of Oracle Enterprise Repository. However, as a result of this process, incompatibilities may emerge between older and newer versions of REX. While full version compatibility is our goal, backwards-compatibility is subject to unpredictable and therefore potentially unavoidable limitations. In future, Oracle Enterprise Repository releases REX includes the following backwards-compatible enhancements:

- Addition of new methods to the Oracle Enterprise Repository Web service
- Definition of new complex types in the WSDL

With regard to these backwards-compatible changes, the regeneration of client proxies is necessary only when the need arises to take advantage of new features and functionality.

The namespace of the service changes only when incompatible changes are unavoidable. Examples of such a change would include the modification of an existing

complex type, or a change in the signature of a method in the service. In this event, client proxy regeneration is necessary, as are the minimal code changes. Client proxies generated from prior versions of REX are unable to connect to the new service.

The namespace of complex types never changes.

12.3 Basic Concepts

This section describes the basic concepts of REX such as getting started with enabling the OpenAPI and consuming the WSDL. This section contains the following topics:

- Section 12.3.1, "Getting Started - Enabling the OpenAPI within the Oracle Enterprise Repository"
- Section 12.3.2, "Getting Started - Consuming the WSDL"

12.3.1 Getting Started - Enabling the OpenAPI within the Oracle Enterprise Repository

The procedure is performed on the Oracle Enterprise Repository Admin screen.

1. Click **System Settings**.
2. Enter the property cmee.extframework.enabled in the Enable New System Setting text box.
3. Click **Enable**. The Open API section is displayed.
4. Ensure the **cmee.extframework.enabled** property is set to `True`.
5. Click **Save**. REX is now enabled within your instance of Oracle Enterprise Repository.

12.3.2 Getting Started - Consuming the WSDL

The first step in using REX is to generate the client-side stubs necessary to communicate with the Oracle Enterprise Repository server. This is generally accomplished using the automated tools provided by the specific Web services toolkit in use. This section describes how to generate client stubs using a variety of integrated development environments and toolkits.

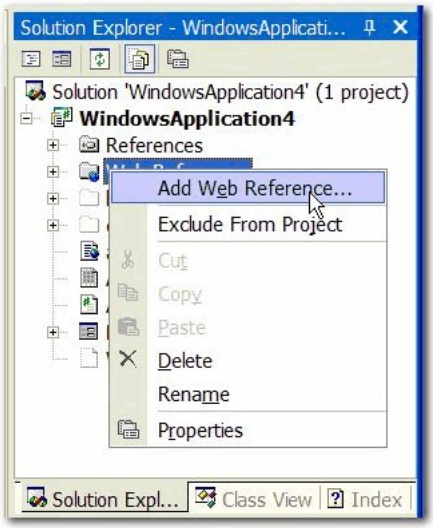
This section contains the following topics:

- "Visual Studio .NET"
- "Eclipse - Lomboz plugin"
- "Authentication and Authorization"
- "Exception Handling"
- "Java and AXIS"
- ".NET"
- "Validation"
- "Query Considerations in REX"
- "Sending Binary Data (Attachments)"
- "Using DIME attachments with .NET and the Microsoft Web Services Enhancement (WSE) Kit"
- "Using SOAP with Attachments and Java AXIS clients"

Visual Studio .NET

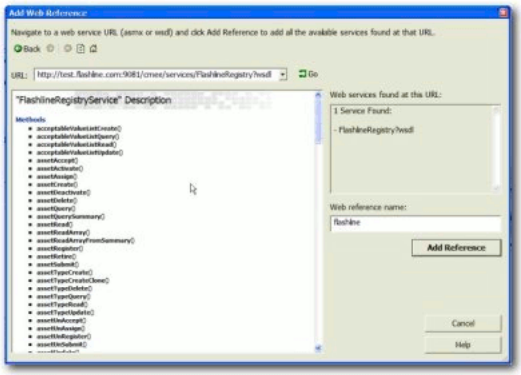
1. Right-click the **Web References** node in the Solution Explorer tree and then select Add Web Reference, as shown in Figure 12–2. The Add Web Reference dialog is displayed.

Figure 12–2 Solution Explorer Window



2. Specify the name of the Web reference, as shown in Figure 12–3.

Figure 12–3 Add Web Reference Dialog



You may load the Oracle Enterprise Repository WSDL file directly from the application server (if it is running) by specifying the proper URL, or from a static file local to your Eclipse project. If you wish to use a URL, it can be found in the following location (replace "yourserver" and "appname" with the appropriate values):

http://yourserver:port/appname/services/FlashlineRegistry?wsdl

3. Connect to the service endpoint. Once the Web reference has been created, your application can use it by establishing an instance of the client service proxy:

```
flashline.FlashlineRegistryService registry = new
flashline.FlashlineRegistryService();
```

4. If the default URL of the service (as contained in the WSDL file used to establish the Web reference) is not the actual address of the Web service, the endpoint address can be changed as follows:

   registry.Url = "http://appserver/oer/services/FlashlineRegistry";

Your application is ready to interact with Oracle Enterprise Repository through REX.

Eclipse - Lomboz plugin

The Lomboz plugin works with Eclipse but any type of Eclipse plugin that provides support for Web Services should work. Most of these tools/plugins usually ask for:

- The location of the WSDL file
- The source directory from your project in which generated code should live
- The WSDL version with which the WSDL file complies.

The Oracle Enterprise Repository WSDL file can be loaded directly from the application server (if it is running) by specifying the proper URL, or from a static file local to your Eclipse project. The WSDL URL can be found in the following location (replace "www.example.com" and "appname" with the appropriate values):

http://www.example.com/appname/services/FlashlineRegistry?wsdl

Choose a source directory on your project's build path as the target of the generated client proxy classes.

Oracle Enterprise Repository WSDL conforms to version 1.1 of the WSDL standard.

Authentication and Authorization

The first step in using REX is authenticating with the server. Authentication is performed using the authTokenCreate method. This method takes a user ID and password as arguments to be used in authenticating with Oracle Enterprise Repository. If the ID and password are successfully authenticated, an authentication token is returned. This token must be used in every subsequent call to REX.

If a valid AuthToken is not included for every REX method, an OpenAPIException is thrown. The applies to all methods except authTokenCreate and authTokenDelete.

The following example shows how to retrieve an AuthToken and use it in subsequent REX calls.

```
package com.example.flashlineclient;
//The imports below are assumed for any of the included examples
import javax.xml.rpc.ServiceException;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Asset;
public class FlexTest {
public FlexTest () {
}
public static void main(String[] pArgs)throws OpenAPIException, RemoteException,
ServiceException {
try {
FlashlineRegistry lRegistry = null;
```

```
AuthToken lAuthToken = null;
URL lURL = null;
lURL = new URL("http://www.example.com/appname/services/FlashlineRegistry");
//"www.example.com" should be your server address
//"appname" is the application name of the location that the Registry is running
 on
//These two things must be changed to the proper values in every example
lRegistry = new FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
lAuthToken = lRegistry.authTokenCreate("username", "password");
System.out.println(lAuthToken.getToken());
//displaying the authtoken as a string to the screen
Asset lAsset = lRegistry.assetRead(lAuthToken, 559);
//reading asset number 559
System.out.println(lAsset.getName());
//displaying the name of asset 559 to the screen
} catch(OpenAPIException lEx) {
System.out.println("ServerCode = "+ lEx.getServerErrorCode());
System.out.println("Message = "+ lEx.getMessage());
System.out.println("StackTrace:");
lEx.printStackTrace();
} catch (RemoteException lEx) {
lEx.printStackTrace();
} catch (ServiceException lEx) {
lEx.printStackTrace();
} catch (MalformedURLException lEx) {
lEx.printStackTrace();
}
System.out.println("execution completed");
System.exit(0);
}
}
```

Authorization

REX enforces the same authorization rules as the Oracle Enterprise Repository application. The user ID and password used to authenticate determines the privileges available to the user through REX. For example, if the authenticated user does not have EDIT privileges assigned for projects, and attempts to create a project using the projectCreate REX method, an OpenAPIException is thrown.

Exception Handling

Open API communicates server errors to the client through a SOAP Fault. The manner in which SOAP Faults are handled varies according the language and SOAP toolkit in use.

This section suggests ways to detect and deal with exceptions generated by the Open API within client code, using the most common platform/toolkit combinations.

Java and AXIS

Exceptions thrown by the Open API are transferred as SOAP Faults, and then de-erialized by the AXIS client toolkit as Java Exceptions. That is, AXIS makes an attempt to map the SOAP Fault to a corresponding client-side OpenAPIException class. Server-side errors are represented to the client as `com.flashline.registry.openapi.OpenAPIException` instances. Consequently, client code can catch exceptions with the code listed below which is from the code above:

```
try {
lAsset = lRegistry.assetCreate(..);
} catch(OpenAPIException lEx) {
```

```
System.out.println("ServerCode = "+ lEx.getServerErrorCode());
System.out.println("Message = "+ lEx.getMessage());
System.out.println("StackTrace:");
lEx.printStackTrace();
} catch (RemoteException lEx) {
lEx.printStackTrace();
} catch (ServiceException lEx) {
lEx.printStackTrace();
} catch (MalformedURLException lEx) {
lEx.printStackTrace();
}
```

.NET

Using a consumed Web service in .NET is a bit more complicated. All service exceptions are caught on the client side as exceptions of type `System.Web.Services.Protocols.SoapException`. This makes it somewhat tricky to retrieve the extended information available in the `OpenAPIException` thrown by the Open API.

The .NET SoapException property represents the SOAP Fault message. However, the additional fields provided by the OpenAPIException, beyond what is explicitly mapped to the standard SOAP Fault, must be obtained by manually parsing the XML Detail property of the .NET SoapException. For example, code similar to the following could be used to view the server-side error code and stack trace returned with an OpenAPIException:

```
try
{
registry.testException();
}
catch(SoapException exc)
{
XmlNode lNode = null;
lNode = exc.Detail.SelectSingleNode("*/serverErrorCode");
if(lNode != null)
Console.Out.WriteLine("Error Code: "+lNode.InnerText);
lNode = exc.Detail.SelectSingleNode("*/serverStackTrace");
if(lNode != null)
Console.Out.WriteLine("Server Stack Trace: \n"+lNode.InnerText);
}
```

It is a good idea to use a more explicit XPath expression than */serverErrorCode to eliminate the chance that the returned SOAP Fault includes more than one XML Element with the name serverErrorCode.

The following SOAP response illustrates an `OpenAPIException` represented as a SOAP Fault message:

```
<?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>
<soapenv:Fault>
<faultcode>soapenv:Server.userException</faultcode>
<faultstring>Error [100], Severity [SEVERE]:An unkown server-side error occured.
Please record stack trace (if available) and contact technical
 support.</faultstring>
```

```
<detail>
<com.flashline.cmee.openapi.OpenAPIException xsi:type="ns1:OpenAPIException"
xmlns:ns1="http://base.openapi.registry.flashline.com">
<message xsi:type="xsd:string">Error [100],
Severity [SEVERE]:An unkown server-side error occured.
Please record stack trace (if available) and contact technical support.</message>
<serverErrorCode xsi:type="xsd:int">100</serverErrorCode>
<serverStackTrace xsi:type="xsd:string">java.lang.NullPointerException&#xd;
at java.util.HashMap.&lt;init&gt;(HashMap.java:214)&#xd;
...
at java.lang.Thread.run(Thread.java:534)&#xd;
</serverStackTrace>
<severity xsi:type="xsd:string">SEVERE</severity>
</com.flashline.cmee.openapi.OpenAPIException>
</detail>
</soapenv:Fault>
</soapenv:Body>
</soapenv:Envelope>
```

Validation

When attempting to save an entity in Oracle Enterprise Repository, the system attempts to validate the input. Any missing or invalid data causes the server to throw an OpenAPIException containing a list of fields and their respective errors. See documentation above regarding exception handling.

For more information, see "Exception Handling".

Query Considerations in REX

The criteria object model is currently moving to a more flexible representation of terms and grouping. As they occur, these changes affects the availability of certain API features when executing a query using a criteria object. The subsystems only directly evaluate their corresponding criteria objects, and do not make use of the extended capabilities of the underlying SearchTermGroup, unless otherwise noted in this documentation.

Sending Binary Data (Attachments)

Various REX methods require sending or receiving potentially large sets of binary data between the client and server. For example, the import/export subsystem provides methods for sending a payload from which to import, and methods for retrieving a payload representing a set of exported assets.

Typically, binary data is transferred through Web services RPC invocations through Dynamic Internet Message Exchange (DIME); SOAP with Attachments (SwA); or Base-64 Encoding. Each has its advantages and disadvantages, but few client toolkits directly support all three.

The Oracle Enterprise Repository OpenAPI supports all three mechanisms for transferring binary data. Details are provided in the following sections. Any method that provides for the binary transfer of data has three versions, each one supporting a different transfer mechanism. For example, to retrieve the results of an export, a user can select any one of the following methods:

- **importGetResultsB64**

  Retrieve results of export in base-64 encoded format. This is the lowest common denominator, and can be used on any platform, provided that the client can encode/decode base-64 data.

- **importGetResultsDIME**

  Retrieve export results as an attached file, using the DIME protocol. This is the preferred option for most .NET clients.

- **importGetResultsSwA**

  Retrieve export results as an attached file, using the SOAP with Attachments (SwA) protocol (MIME-based)

Using DIME attachments with .NET and the Microsoft Web Services Enhancement (WSE) Kit

Microsoft provides an extension to the standard .Net Web service toolkit. The Microsoft Web Services Enhancement (WSE) kit provides advanced functionality, such as sending and receiving attachments through Web services using the Dynamic Internet Messaging Exchange (DIME) protocol.

The following code snippet gives an example of sending data through a DIME attachment:

```
// relax the requirement for the server to understand ALL headers. This MUST be
// done, or the call with the attachment fails. After the call, if you wish,
// you can set this back to "true"
registry.RequestSoapContext.Path.EncodedMustUnderstand= "false";
// clear the attachments queue
registry.RequestSoapContext.Attachments.Clear();
registry.RequestSoapContext.Attachments.Add(
new Microsoft.Web.Services.Dime.DimeAttachment("0", "application/zip",
Microsoft.Web.Services.Dime.TypeFormatEnum.MediaType, "c:\\tmp\\import.zip"));
// start an import running on the server
registry.importExecute(lAuthToken, "flashline", null, "FEA Flashpack Import",
null);
// do some polling (calls to importStatus) to monitor the import progress,
// if you wish
```

The following code snippet gives an example of receiving data through a DIME attachment:

```
// relax the requirement for the server to understand ALL headers. This MUST be
// done, or the call with the attachment fails. After the call, if you wish,
// you can set this back to "true"
registry.RequestSoapContext.Path.EncodedMustUnderstand= "false";
// clear the attachments queue
registry.RequestSoapContext.Attachments.Clear();
// start an export
flashline.ImpExpJob lJob =
registry.exportExecute(lAuthToken, "flashline", null, "Complete Export",
 "flashline",
"<entitytypes>
<entitytype type=\"acceptableValueList\">
<entities>
<entity id=\"100\"/>
</entities>
</entitytype>
</entitytypes>");
// do some polling (calls to exportStatus) to watch the progress of the
// export, if you wish...
// this call blocks until either the method returns (or an exception is
 thrown),
// or the call times out.
registry.exportGetResultsDIME(lAuthToken, lJob);
```

```
// check to see if the call resulted in attachments being returned...
if(registry.ResponseSoapContext.Attachments.Count > 0)
{
Stream lStream = registry.ResponseSoapContext.Attachments[0].Stream;
// write the data out somewhere...
}
```

Using SOAP with Attachments and Java AXIS clients

The Axis client provides functions to handle SOAP attachments in Java. For more information, see *http://www-106.ibm.com/developerworks/webservices/library/ws-soapatt/*.

The following code snippet gives an example of receiving data:

```
byte[] lResults = null;
ImpExpJob lExportJob =
mFlashlineRegistrySrc.exportExecute(mAuthTokenSrc,"flashline",null,
"Export Assets","default", createAssetQuery().toString());
lExportJob =
mFlashlineRegistrySrc.exportStatus(mAuthTokenSrc, lExportJob);
lResults =
mFlashlineRegistrySrc.exportGetResultsB64(mAuthTokenSrc, lExportJob);
// write the results out to disk in a temp file
File lFile = null;
String lTempDirectory =
System.getProperty("java.io.tmpdir");
lFile = new File(lTempDirectory + File.separator + "impexp.zip");
FileOutputStream lOS = new FileOutputStream(lFile);
BufferedOutputStream lBOS = new BufferedOutputStream(lOS);
lBOS.write(lResults);
lBOS.flush();
lBOS.close();
lOS.close();
```

The following code snippet gives an example of sending data through a DIME attachment:

```
// open file and attach as data source
InputStream lIS = new FileInputStream(lFile);
((Stub)mFlashlineRegistryDest)._setProperty
(Call.ATTACHMENT_ENCAPSULATION_FORMAT, Call.ATTACHMENT_ENCAPSULATION_FORMAT_DIME);
ByteArrayDataSource lDataSource = new ByteArrayDataSource(lIS,
 "application/x-zip-compressed");
DataHandler lDH = new DataHandler(lDataSource);
// add the attachment
((Stub)mFlashlineRegistryDest).addAttachment(lDH);
ImpExpJob lJob =
mFlashlineRegistryDest.importExecute(mAuthTokenDest, "flashline", null, "Import
 Assets Test", null);
```

13

ArtifactStore API

This chapter provides an overview of ArtifactStore API and describes the use cases using this API.

This chapter contains the following sections:

- Section 13.1, "Overview"
- Section 13.2, "Use Cases"

13.1 Overview

The ArtifactStore Subsystem provides a Web Services-based mechanism that is used to query and create Oracle Enterprise Repository ArtifactStores.

13.2 Use Cases

This section describes the use cases using the Artifact Store API. It contains the following topics:

- Section 13.2.1, "Use Case: Create missing ArtifactStore"

13.2.1 Use Case: Create missing ArtifactStore

Description

This use case describes creating a missing artifactstore.

Sample code is as follows:

```
package com.flashline.sample.artifactstoreapi;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.ArtifactStoreBean;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.ArtifactStoreCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ArtifactStores {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
  ServiceException {
    try {
```

```
        ////////////////////////////////////////////////////////
        // Connect to Oracle Enterprise Repository
        ////////////////////////////////////////////////////////
        URL lURL = null;
        lURL = new URL(pArgs[0]);
        FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
        // ////////////////////////////
        // Authenticate with OER
        // ////////////////////////////
        AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
        // ---------------------------------------
        // query for an artifact store
        ArtifactStoreCriteria lArtifactStoreCriteria = null;
        ArtifactStoreBean[] lArtifactStoreBeans = null;
        ArtifactStoreBean lArtifactStoreBean = null;
        lArtifactStoreCriteria = new ArtifactStoreCriteria();
        lArtifactStoreCriteria.setHostCriteria("existing-artifact-store.com");
        lArtifactStoreCriteria.setBasepathCriteria("/");
        lArtifactStoreBeans = repository.artifactStoreQuery(authToken,
 lArtifactStoreCriteria, false);
        // create a missing artifact store if missing and based on the criteria
        lArtifactStoreCriteria = new ArtifactStoreCriteria();
        lArtifactStoreCriteria.setHostCriteria("missing-artifact-store.com");
        lArtifactStoreCriteria.setBasepathCriteria("/");
        // a new artifact store is created
        lArtifactStoreBeans = repository.artifactStoreQuery(authToken,
 lArtifactStoreCriteria, true);
        lArtifactStoreBean = lArtifactStoreBeans[0];
      } catch(Exception e) {
        throw new RuntimeException(e.getMessage());
      }
    }
  }
```

14

AcceptableValueLists API

This chapter provides an overview of AcceptableValueLists API and describes the use cases using this API.

This chapter contains the following sections:

- Section 14.1, "Overview"
- Section 14.2, "Use Cases"

14.1 Overview

Acceptable Value Lists are used in single- and multiple-selection drop-down box metadata elements.

When creating or editing an asset type, Acceptable Value Lists are used as metadata elements. These metadata elements are referenced by ID in the editor and viewer XML for the asset type/compliance template.

When creating or editing assets, values contained in Acceptable Value Lists are used as options for the metadata elements defined for the particular asset type/compliance template. To use the acceptable values for an Acceptable Value List, the custom data for the asset (Asset.GetCustomData()) is modified to reference the ID of the acceptable value.

14.2 Use Cases

This section describes the use cases using the Acceptable ValueLists API. It contains the following topics:

- Section 14.2.1, "Use Case: Create and Edit an Acceptable Value List"
- Section 14.2.2, "Use Case: Find an Acceptable Value List and use it in an asset"

14.2.1 Use Case: Create and Edit an Acceptable Value List

Description

Create a new acceptable value list and enter it into Oracle Enterprise Repository.

Sample code is as follows:

```
package com.flashline.sample.acceptablevaluelists;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
```

```
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AcceptableValue;
import com.flashline.registry.openapi.entity.AcceptableValueList;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateAndEditAcceptableValueList {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ///////////////////////////////
      // Build an array of acceptable values for the list.
      // ///////////////////////////////
      String newAcceptableValueListName = "My AcceptableValueList
 "+Calendar.getInstance().getTimeInMillis();
      AcceptableValue[] acceptableValues = new AcceptableValue[3];
      acceptableValues[0] = new AcceptableValue();
      acceptableValues[0].setValue("My Value");
      acceptableValues[1] = new AcceptableValue();
      acceptableValues[1].setValue("My Next Value");
      acceptableValues[2] = new AcceptableValue();
      acceptableValues[2].setValue("My Last Value");
      // ///////////////////////////////
      // Create the AcceptableValueList in Repository
      // ///////////////////////////////
      AcceptableValueList newAcceptableValueList = repository
          .acceptableValueListCreate(authToken, newAcceptableValueListName,
              acceptableValues);
      System.out.println("The new acceptableValueList id  =\""
          + newAcceptableValueList.getID() + "\"");
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

14.2.2 Use Case: Find an Acceptable Value List and use it in an asset

Description

Populate an asset's single or multiple selection lists with acceptable values.

Sample code is as follows:

```
package com.flashline.sample.acceptablevaluelists;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AcceptableValue;
import com.flashline.registry.openapi.entity.AcceptableValueList;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AcceptableValueListCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class FindAcceptableValueListAndUseInAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ////////////////////////////////
      // Find the AcceptableValueList
      // ////////////////////////////////
      AcceptableValueListCriteria criteria = new AcceptableValueListCriteria();
      criteria.setNameCriteria("My AcceptableValueList");
      AcceptableValueList[] acceptableValueLists = repository
          .acceptableValueListQuery(authToken, criteria);
      AcceptableValueList myAcceptableValueList = acceptableValueLists[0];
      AcceptableValue[] acceptableValues = myAcceptableValueList
          .getAcceptableValues();
      // ////////////////////////////////
      // Find one value within the AcceptableValueList
      // ////////////////////////////////
      AcceptableValue myAcceptableValue = null;
      for (int i = 0; i < acceptableValues.length; i++) {
        if (acceptableValues[i].getValue().equals("My Value")) {
          myAcceptableValue = acceptableValues[i];
          break;
        }
```

```
        }
        long myAcceptableValueID = myAcceptableValue.getID();
        Asset myAsset = repository.assetRead(authToken, 561);
        String customData = myAsset.getCustomData();
        // ////////////////////////////
        // Modify customData to use myAcceptableValueID.
        // ////////////////////////////
        String modifiedCustomData = customData;
        // ...
        // ////////////////////////////
        // save modified custom data
        // ////////////////////////////
        myAsset.setCustomData(modifiedCustomData);
        repository.assetUpdate(authToken, myAsset);
      } catch (OpenAPIException lEx) {
        System.out.println("ServerCode = " + lEx.getServerErrorCode());
        System.out.println("Message    = " + lEx.getMessage());
        System.out.println("StackTrace:");
        lEx.printStackTrace();
      } catch (RemoteException lEx) {
        lEx.printStackTrace();
      } catch (ServiceException lEx) {
        lEx.printStackTrace();
      } catch (MalformedURLException lEx) {
        lEx.printStackTrace();
      }
    }
  }
```

15

Asset API

This chapter provides an overview of Asset API and describes the use cases using this API.

This chapter contains the following sections:

- Section 15.1, "Overview"
- Section 15.2, "Use Cases"

15.1 Overview

Assets are the core entities in the Oracle Enterprise Repository. This document covers the Asset Subsystem actions Create, Read, Update, Delete, and Query. It also covers the modification of:

- Asset active status:
  - Activate
  - Deactivate
  - Retire
- Asset registration status:
  - Submit
  - Accept
  - Register
  - Unsubmit
  - Unaccept
  - Unregister
- Asset assignment status:
  - Assign
  - Unassign

Several issues must be taken into consideration when working with assets in Oracle Enterprise Repository using REX. Trade-offs in memory consumption, and performance should be carefully weighed.

- **Memory Consumption**

  Assets and their metadata consume a significant amount of memory on both the REX server and on the client software using REX. When searching Oracle

Enterprise Repository using REX, it is possible that a large number of assets may be returned. To avoid Denial of Service interruptions to the system, administrators can limit the maximum number of results that can be returned in a call to the REX method assetQuery. The system setting cmee.extframework.assets.max controls the maximum number of search results that can be returned as a result of a query. If the number of results matching a query exceeds the maximum, an exception is generated by REX.

In cases where it is expected that a potentially large number of assets matches a query, the assetQuerySummary method is recommended. This alternative method of querying Oracle Enterprise Repository matches exactly the same assets as a call to assetQuery, but returns lightweight asset summary objects, rather than the full asset objects. These summary objects consume a nominal amount of memory, and the possibility of exhausting resources as a result of a query is consequently negligible.

Once a summary query has been performed, the full asset objects can be retrieved for assets of interest using either assetRead, or assetReadArrayFromSummary. If multiple assets are desired, use of the assetReadArrayFromSummary method is recommended. See the API documentation for details on using this method.

- **Performance**

  REX is based on standard Web Services technology, which provides many significant advantages in flexibility and portability. However, as with any Web Services-based technology, performance can be challenging, particularly in high data volume situations (e.g., large numbers of assets being manipulated). REX provides options that allow developers to avoid potential performance problems.

  - **Iterative Reads**

    The primary overhead in web services technology is incurred in the serialization and de-serialization of data using XML, combined with network transfer. Much of this overhead can be avoided in situations where a number of assets are to be read. For example, if 50 assets are to be retrieved from Oracle Enterprise Repository using REX, the developer could perform 50 assetRead calls. A better approach, however, would be to use the assetReadArray method, passing the IDs of the desired assets as a single argument. This would retrieve all 50 assets in one call, dramatically improving performance.

  - **Listing Operations**

    Often data is retrieved from REX for the purpose of displaying a listing to an end user, who then is expected to select an asset for closer inspection. In cases like these, the full extent of asset metadata is not required to generate a summary list. As discussed in the section on memory above, consider using the summary methods provided in REX.

- **Access Control**

  - Which assets a user of REX can see, and to some extent the information in those assets, is controlled by access settings. The same access restrictions that exist for a user accessing the system through the web gui also apply to the REX asset subsystem.
  - Query Restrictions - users can only retrieve assets in a call to assetQuery or assetRead for which they have view permission.
  - Update Restrictions - users can only update assets for which they have edit permission.

- File restrictions - users can only view the files for which they have download permissions as set in the File type Custom Access Settings applied to each individual file. This means that a user might be able to view an asset, but might not be able to view any of the asset's files. Each file can have it's own permissions, different than the asset's permissions. If specific File type permissions are not applied to a file, these permissions is inherited from the asset's permissions to which the files belong.

15.1.1 Definitions

This section contains the following definitions:

- **ID and UUID**
  - ID is an internal unique identifier (numeric) used to identify an asset uniquely within a single Oracle Enterprise Repository instance.
  - UUID is a universally unique identifier (128-bit numeric represented as a hexidecimal string) used to identify an asset uniquely across any instance of Oracle Enterprise Repository. Each asset's UUID is exposed primarily for purposes of reading and searching. Oracle strongly advises not modifying this field using REX. However, if an adiminstrator does choose to modify an asset's UUID, then the format must be consistent (00000000-0000-0000-0000-000000000000) and the UUID must be unique within the target Oracle Enterprise Repository instance; otherwise, the change operation fails.
- **Name and Version**

  String fields that combine to uniquely identify an asset.
- **Custom Data**

  Customizable metadata for an asset is stored in an XML structure within this string. The sample code describes the custom data methods effectively.

15.1.2 Sample Code

```
package com.flashline.sample.assetapi;
import java.io.StringReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.text.Format;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import javax.xml.rpc.ServiceException;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;
import org.jdom.xpath.XPath;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateCustomData {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException, ServiceException {
```

```
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
      .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////////
      // assetUpdateCustomDataString
      ////////////////////////////////////////
      // Find and Modify a custom data field value with a value supplied on the
 command line
      Long currentLicenses = new
 Long(repository.assetReadCustomDataString(authToken, 589, "/
_-total-licenses-owned"));
      currentLicenses = new Long(currentLicenses.intValue() + 1);
      //    save the modifications
      //  NOTICE: A leading slash is required to specify the appropriate path to
 the element being updated.
      repository.assetUpdateCustomDataString(authToken, 589, "/
_-total-licenses-owned", currentLicenses.toString());
      ////////////////////////////////////////
      // assetUpdateCustomDataStringArray
      ////////////////////////////////////////
      // Add a custom data field value with a value supplied on the command line
      Format formatter = new SimpleDateFormat("yyyyMMdd");
      String dateFormat = formatter.format(new Date());
      // NOTICE: for the following method, there is no leading slash for the
 elements being updated.
      String[] versionHistory = {"version-history/version-history/version-number",
 "version-history/version-history/production-date-yyyymmdd-",
 "version-history/version-history/comments"};
      String[] versionHistoryValues = {currentLicenses.toString(), dateFormat,
 "Updated version History: " + dateFormat};
      //    save the modifications
      repository.assetUpdateCustomDataStringArray(authToken, 589, versionHistory,
 versionHistoryValues);
      ////////////////////////////////////////
      // assetUpdateCustomDataNode
      ////////////////////////////////////////
      //The following updates a specific custom data element called
 "document-name" that is a child of "document",
      //which is a child of "documentation"
      XPath lXPath = null;
      List lElms = null;
      //First read the Node "documentation" of the specific asset
      String lXMLDocumentation = repository.assetReadCustomDataNode(authToken,
 589, "documentation");
      //Using DOM, convert the XML to a Document
      Document lDoc = null;
      SAXBuilder lBuilder = new SAXBuilder();
      StringReader lReader = null;
      lReader = new StringReader(lXMLDocumentation);
      lBuilder.setValidation(false);
```

```
     lBuilder.setIgnoringElementContentWhitespace(true);
     lDoc = lBuilder.build(lReader);
     lXPath = XPath.newInstance("documentation/document");
     lElms = lXPath.selectNodes(lDoc);
     //Cycle through the "document" elements until we find the one we want.  Then
update it.
     for (int i=0;i<lElms.size();i++) {
       Element lElm = (Element)lElms.get(i);
       List lChildElms = lElm.getChildElements();
       for (int x=0;x<lChildElms.size();x++) {
         Element lChildElm = (Element)lChildElms.get(x);
         if (lChildElm.getName().equals("document-name") &&
lChildElm.getValue().equals("API")) {
           lChildElm.setText("API KHAN");
         } else {
           lChildElm.setText(lChildElm.getValue());
         }
       }
     }
     //Convert the Document back to an XML string and update the asset's custom
data.
     repository.assetUpdateCustomDataNode(authToken, 589, "documentation", new
XMLOutputter().outputString(lDoc));
     ////////////////////////////////////////
     // assetUpdateCustomDataNodeArray
     ////////////////////////////////////////
     try {
       //The following updates multiple custom data elements.  One is called
"document-name" that is a child of "document",
       //which is a child of "documentation"
       //The other is the element called "also-known-as"
       lXPath = null;
       lElms = null;
       //First read the Node "documentation" of the specific asset
       lXMLDocumentation = repository.assetReadCustomDataNode(authToken, 589,
"documentation");
       //Using DOM, convert the XML to a Document
       lDoc = null;
       lBuilder = new SAXBuilder();
       lReader = null;
       lReader = new StringReader(lXMLDocumentation);
       lBuilder.setValidation(false);
       lBuilder.setIgnoringElementContentWhitespace(true);
       lDoc = lBuilder.build(lReader);
       lXPath = XPath.newInstance("documentation/document");
       lElms = lXPath.selectNodes(lDoc);
       //Cycle through the "document" elements until we find the one we want.
Then update it.
       for (int i=0;i<lElms.size();i++) {
         Element lElm = (Element)lElms.get(i);
         List lChildElms = lElm.getChildElements();
         for (int x=0;x<lChildElms.size();x++) {
           Element lChildElm = (Element)lChildElms.get(x);
           if (lChildElm.getName().equals("document-name") &&
lChildElm.getValue().equals("API")) {
             lChildElm.setText("API KHAN");
           } else {
             lChildElm.setText(lChildElm.getValue());
           }
         }
```

```
        }
        String lDoc1 = new XMLOutputter().outputString(lDoc);
        //Get the next element
        lXMLDocumentation = repository.assetReadCustomDataNode(authToken, 589,
"also-known-as");
        lDoc = null;
        lBuilder = new SAXBuilder();
        lReader = null;
        lReader = new StringReader(lXMLDocumentation);
        lBuilder.setValidation(false);
        lBuilder.setIgnoringElementContentWhitespace(true);
        lDoc = lBuilder.build(lReader);
        lXPath = XPath.newInstance("also-known-as");
        lElms = lXPath.selectNodes(lDoc);
        //Get the also-known-as element
        for (int i=0;i<lElms.size();i++) {
          Element lElm = (Element)lElms.get(i);
          lElm.setText("Modified Alias");
        }
        String lDoc2 = new XMLOutputter().outputString(lDoc);
        //Convert the Document back to an XML string and update the asset's custom
data.
        repository.assetUpdateCustomDataNodeFromStringArray(authToken, 589, new
String[] {"documentation", "also-known-as"}, new String[] {lDoc1, lDoc2});
      } catch (Exception e) {
        e.printStackTrace();
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

15.1.3 Related Subsystems

This sections describes the following related subsystems that are used with the Asset API:

- **AssetType**

  All assets need an active and valid asset type. This asset type defines the metadata that can be stored in the custom data for the asset.

- **Vendor**

  If desired, an asset may be linked to a vendor. This linking is done by the vendor ID.

- **AcceptableValueLists**

  When creating or editing assets, acceptable values contained in acceptable value lists are used to as options for the metadata elements that were defined for the asset type. To use the acceptable values for an acceptable value list, modify the custom data for the asset (Asset.GetCustomData()) to have it reference the id of the acceptable value.

- **RelationshipType**

  Relationship types define the kinds of relationships that can exist between assets.

- **Categorization Types**

  Categorization types are top-level groups of categorizations added to asset types. Categorizations describe an asset.

- **Projects**

  Assets can be produced by projects. The producing projects for an asset are stored in an array of ID's.

- **Users**

  Users can be assigned to assets. They are the person who is responsible for working up the metadata.

15.2 Use Cases

This section describes the use cases using the Asset API. It contains the following topics:

- Section 15.2.1, "Use Case: Creating a new asset"
- Section 15.2.2, "Use Case: Creating a new asset from XML"
- Section 15.2.3, "Use Case: Modifying an asset"
- Section 15.2.4, "Use Case: Assign users to an asset"
- Section 15.2.5, "Use Case: Building an asset search"
- Section 15.2.6, "Use Case: Upgrading asset status"
- Section 15.2.7, "Use Case: Downgrading asset status"
- Section 15.2.8, "Use Case: Apply and remove Compliance Templates from a project"
- Section 15.2.9, "Use Case: Creating the new version of an asset and retiring the old version"
- Section 15.2.10, "Use Case: "Housekeeping""
- Section 15.2.11, "Use Case: Finding assets and updating custom-data"
- Section 15.2.12, "Use Case: Reading an Asset's Tabs"
- Section 15.2.13, "Use Case: Retrieve An Asset's Tab Based on TabType"
- Section 15.2.14, "Use Case: Approving and Unapproving a tab"

15.2.1 Use Case: Creating a new asset

Description

Create a new asset and enter it into Oracle Enterprise Repository.

Sample code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////
      // Login to OER
      ////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////
      // Create new asset
      ////////////////////////////////////
      Asset myAsset = repository.assetCreate(authToken,
          "My Asset Name108", "My Version
 "+Calendar.getInstance().getTimeInMillis(), 144);
      ////////////////////////////////////////////
      //The following demonstrates how to modify a custom data Date element on an
 asset.
      //This date must be in a specific format and the name of the element must by
 known.
      //In this example, the name of the element is "testdate".  This element must
 have been created in the
      //asset type as a Date element
      //Update the testdate field to January 1, 2007
      //Note: the format of the date should match the system setting for Short
 Date.
      repository.assetUpdateCustomDataString(authToken, myAsset.getID(),
 "testdate", "2007-1-1");
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
```

```
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.2 Use Case: Creating a new asset from XML

Description

It is also possible to create a new asset from an XML representation of the asset. Schemas are used to validate the asset XML before creation. The schema for an asset type is available through the Open API as can be seen in the example below.

It is not necessary to do validation yourself, the asset XML is validated internally before the create happens. If you do want to do your own validation, then you must find a validating XML parser such as Xerces 2.0.

Sample code is as follows:

```
package com.flashline.sample.assetapi;
import java.io.IOException;
import java.io.StringReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.rpc.ServiceException;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.DefaultHandler;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewAssetFromXML {
  public static void main(String pArgs[]) throws ServiceException,
      ParserConfigurationException, SAXException, IOException {
    String SCHEMA_LANGUAGE =
 "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
    String XML_SCHEMA = "http://www.w3.org/2001/XMLSchema";
    String SCHEMA_SOURCE = "http://java.sun.com/xml/jaxp/properties/schemaSource";
    SAXParserFactory lSaxParserFactory = null;
    SAXParser lSaxParser = null;
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
```

```
        lURL = new URL(pArgs[0]);
        FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
            .getFlashlineRegistry(lURL);
        ////////////////////////////////////
        // Login to OER
        ////////////////////////////////////
        AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
        // Anonymous class to handle validation errors
        DefaultHandler lDefaultHandler = new DefaultHandler() {
          public void error(SAXParseException exception) throws SAXException {
            throw exception;
          }
          public void fatalError(SAXParseException exception) throws SAXException {
            throw exception;
          }
        };
        ////////////////////////////////////////////////////////////////////
        // Define the asset you want to create in XML
        ////////////////////////////////////////////////////////////////////
        // This is the XML of the asset we're creating. Typically it would
        // come from a GUI or other asset creation mechanism. It is hard
        // coded for this example.
        ////////////////////////////////////////////////////////////////////
        String assetXML = "<asset id=\"0\">"
            + "           <asset-type id=\"145\" icon=\"component.gif\"
lastSavedDate=\"17 Jul 2007 12:00:00 AM\">Component</asset-type>"
            + "           <mandatory-data>"
            + "               <name>NewComponent</name>"
            + "
<version>"+Calendar.getInstance().getTimeInMillis()+"</version>"
            + "               <description><![CDATA[My Description]]></description>"
            + "               <keywords/>"
            + "               <notification-email/>"
            + "               <applied-policies/>"
            + "               <vendor id=\"0\"/>"
            + "               <file-informations/>"
            + "               <hash-informations/>"
            + "               <producing-projects/>"
            + "               <submission-files/>"
            + "               <applied-compliance-templates/>"
            + "               <contacts/>"
            + "               <relationships/>"
            + "               <categorization-types/>"
            + "           </mandatory-data>"
            + "           <admin-data>"
            + "           </admin-data>"
            + "       </asset>";
        ////////////////////////////////////////////////////////////////////
        // This returns the Schema for the asset type of the asset we're
        // creating
        ////////////////////////////////////////////////////////////////////
        String schema = repository.assetTypeSchemaRead(authToken, 144);
        ////////////////////////////////////////////////////////////////////
        // This block of code shows validating the asset XML against
        // the schema
        ////////////////////////////////////////////////////////////////////
        lSaxParserFactory = SAXParserFactory.newInstance();
        lSaxParserFactory.setNamespaceAware(true);
        lSaxParserFactory.setValidating(true);
        lSaxParser = lSaxParserFactory.newSAXParser();
```

```
      lSaxParser.setProperty(SCHEMA_LANGUAGE, XML_SCHEMA);
      lSaxParser.setProperty(SCHEMA_SOURCE, new InputSource(new StringReader(
          schema)));
      lSaxParser.parse(new InputSource(new StringReader(assetXML)),
          lDefaultHandler);
      ////////////////////////////////////////////////////////////////////
      // If no exception was thrown the asset XML validates and
      // the creation should not fail due to XML formatting errors.
      ////////////////////////////////////////////////////////////////////
      repository.assetCreateFromXML(authToken, assetXML);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.3 Use Case: Modifying an asset

Description

Modify the metadata for an existing asset.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Categorization;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.query.CategorizationTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ModifyExistingAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
```

```
        ////////////////////////////////////////////////////////////
        // Login to OER
        ////////////////////////////////////////////////////////////
        AuthToken authToken = repository.authTokenCreate(
            pArgs[1],pArgs[2]);
        ////////////////////////////////////////////////////////////
        // Read the asset you want to modify
        ////////////////////////////////////////////////////////////
        Asset myAsset = repository.assetRead(authToken, 559);
        // 559 is the example asset number
        ////////////////////////////////////////////////////////////
        // Modify the name, version, description, and notification
        // email
        ////////////////////////////////////////////////////////////
        myAsset.setName("New Name");
        myAsset.setVersion("New Version");
        myAsset.setDescription("New Description");
        myAsset.setNotificationEmail("user@example.com");
        ////////////////////////////////////////////////////////////
        // Modify Categorizations on the asset
        ////////////////////////////////////////////////////////////
        // Setup arrays used for assigning categorizations
        CategorizationType[] lAllCatTypes = null;
        Categorization[] lAllCats = null;
        CategorizationType[] lCatTypes = new CategorizationType[1];
        Categorization[] lCats = new Categorization[1];
        ////////////////////////////////////////////////////////////
        // Search for all categorizations that are asset assignable
        ////////////////////////////////////////////////////////////
        CategorizationTypeCriteria categorizationTypeCriteria = new
CategorizationTypeCriteria();
        categorizationTypeCriteria.setNameCriteria("");
        lAllCatTypes = repository.categorizationTypeQuery(authToken,
            categorizationTypeCriteria);
        ////////////////////////////////////////////////////////////
        // Find all the categorizations to be assigned to the asset
        ////////////////////////////////////////////////////////////
        for (int i = 0; i < lAllCatTypes.length; i++) {
          CategorizationType lCatType = repository.categorizationTypeRead(
              authToken, lAllCatTypes[i].getID());
          lAllCats = repository.categorizationReadByType(authToken,
              lCatType, true, true);
          if (lAllCats.length > 0) {
            lCatTypes[0] = lCatType;
            // when we find the first one, use it
            break;
          }
        }
        lCats[0] = lAllCats[0];
        ////////////////////////////////////////////////////////////
        // Modify the asset to use the categorizations
        ////////////////////////////////////////////////////////////
        myAsset.setCategorizations(lCats);
        myAsset.setCategorizationTypes(lCatTypes);
        ////////////////////////////////////////////////////////////
        // Modify the custom access settings for the asset
        ////////////////////////////////////////////////////////////
        String[] lCasTypes = repository.customAccessSettingTypesGet(authToken);
        String[] lCustomAccessSettings = null;
        if (lCasTypes!=null && lCasTypes.length>0) {
```

```
        lCustomAccessSettings = repository.customAccessSettingNamesGet(authToken,
 lCasTypes[0]);
      }
      if (lCustomAccessSettings!=null && lCustomAccessSettings.length>0) {
        String[] myCustomAccessSettings = { lCustomAccessSettings[0] };
        myAsset.setCustomAccessSettings(myCustomAccessSettings);
      }
      ////////////////////////////////////////////////////////////
      // Add producing projects to the asset
      ////////////////////////////////////////////////////////////
      long[] producingProjectsIDs = new long[1];
      producingProjectsIDs[0] = 50000;
      myAsset.setProducingProjectsIDs(producingProjectsIDs);
      ////////////////////////////////////////////////////////////
      // save the modifications
      ////////////////////////////////////////////////////////////
      repository.assetUpdate(authToken, myAsset);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.4 Use Case: Assign users to an asset

Description

Multiple users can be assigned to an asset.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssignedUser;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.query.UserCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AssignUsers {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
```

```
try {
  ////////////////////////////////////////////////////////
  // Connect to Oracle Enterprise Repository
  ////////////////////////////////////////////////////////
  URL lURL = null;
  lURL = new URL(pArgs[0]);
  FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
      .getFlashlineRegistry(lURL);
  ////////////////////////////////////////////////////////
  // Login to OER
  ////////////////////////////////////////////////////////
  AuthToken authToken = repository.authTokenCreate(
      pArgs[1],pArgs[2]);
  ////////////////////////////////////////////////////////
  // Retrieve desired asset
  ////////////////////////////////////////////////////////
  Asset myAsset = repository.assetRead(authToken, 559);
  // 559 is the example asset number
  ////////////////////////////////////////////////////////
  // Create array of AssignedUser objects
  ////////////////////////////////////////////////////////
  AssignedUser[] lUsers = new AssignedUser[3];
  ////////////////////////////////////////////////////////
  // NOTE:
  //
  // The AssignedUser object has two methods:
  // setUserID(long)
  // setAssignedDate(Calendar).
  // (Specifies the date the user was assigned to the
  // asset. If no date is specified, the current date
  // is used.)
  ////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////
  // Add AssignedUser objects to the array
  ////////////////////////////////////////////////////////
  AssignedUser lUser = new AssignedUser();
  lUser.setUserID(99); // 99 is the admin user id
  lUsers[0] = lUser;
  lUser = new AssignedUser();
  RegistryUser lRegistryUser1 = createRegistryUser(repository, authToken);
  lUser.setUserID(lRegistryUser1.getID());
  lUsers[1] = lUser;
  lUser = new AssignedUser();
  RegistryUser lRegistryUser2 = createRegistryUser(repository, authToken);
  lUser.setUserID(lRegistryUser2.getID());
  lUsers[2] = lUser;
  ////////////////////////////////////////////////////////
  // Add array to the asset that is being updated
  ////////////////////////////////////////////////////////
  myAsset.setAssignedUsers(lUsers);
  ////////////////////////////////////////////////////////
  // save the modifications
  ////////////////////////////////////////////////////////
  repository.assetUpdate(authToken, myAsset);
} catch (OpenAPIException lEx) {
  System.out.println("ServerCode = " + lEx.getServerErrorCode());
  System.out.println("Message    = " + lEx.getMessage());
  System.out.println("StackTrace:");
  lEx.printStackTrace();
} catch (RemoteException lEx) {
```

```
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static RegistryUser createRegistryUser(FlashlineRegistry repository,
 AuthToken authToken) throws OpenAPIException, RemoteException {
    RegistryUser lRet = null;
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    lRet = repository.userCreate(authToken, lUserName, "", lUserName,
 lUserName+"@example.com", lUserName, false, false, false);
    return lRet;
  }
}
```

15.2.5 Use Case: Building an asset search

Description

Finding all assets that meet certain criteria.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetSummary;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.DateRangeSearchTerm;
import com.flashline.registry.openapi.query.SearchTerm;
import com.flashline.registry.openapi.query.TabStatusSearchTerm;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class FindAssets {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////////////////////////////
      // Search for all assets
```

```
        ////////////////////////////////////////////////////////////
        AssetCriteria criteria = new AssetCriteria();
        AssetSummary[] assets = repository.assetQuerySummary(authToken, criteria);
        ////////////////////////////////////////////////////////////
        // try a general search which includes Name, version,
        // description, and keywords
        ////////////////////////////////////////////////////////////
        criteria = new AssetCriteria();
        criteria.setGeneralCriteria("My Asset");
        assets = repository.assetQuerySummary(authToken, criteria);
        ////////////////////////////////////////////////////////////
        // Search for assets that contain a specific search string
        // in one particular field.
        ////////////////////////////////////////////////////////////
        criteria = new AssetCriteria();
        criteria.setNameCriteria("My Name");
        criteria.setVersionCriteria("My version");
        criteria.setDescriptionCriteria("My Description");
        assets = repository.assetQuerySummary(authToken, criteria);
        ////////////////////////////////////////////////////////////
        // Implementing a Search through the AssetCriteria Object
        // -----------------------------------------------
        // If no operator is specified when implementing a search
        // using the setSearchTerms method in the AssetCriteria
        // object, the system defaults to the operator EQUALS.
        // The operator LIKE must be specified if required for the
        // search.
        ////////////////////////////////////////////////////////////
        criteria = new AssetCriteria();
        SearchTerm lSearchTerm = new SearchTerm();
        lSearchTerm.setKey("name");
        lSearchTerm.setOperator("LIKE");
        lSearchTerm.setValue("Test");
        SearchTerm[] lTerms = new SearchTerm[1];
        lTerms[0] = lSearchTerm;
        criteria.setSearchTerms(lTerms);
        assets = repository.assetQuerySummary(authToken, criteria);
        //Search for assets where a specific DATETIME field is a given age
        ////////////////////////////////////////////////////////////////
        criteria = new AssetCriteria();
        //Not specifying an operator defaults to 'equals'.
        DateRangeSearchTerm lTerm = new DateRangeSearchTerm();
        ////////////////////////
        //date-range is the query key.  registereddate is the date field we are
searching on
        //Allowable fields to search on:
        //submitteddate
        //registereddate
        //accepteddate
        //createddate
        //updateddate
        //To do a search for all assets that were registered more than 5 days ago
        lTerm.setKey("date-range");
        //Set the value to a day 5 days older than the current date.  Assume today's
date is 1/10/2007
        //The format defaults to OER's system setting for Short Date Format.
        //The format can be set to any valid date format using setDateFormat() and
passing the date format.
        lTerm.setDateFormat("yyyy-MM-dd");
        lTerm.setBeginDate("2007-1-5");
```

```
     lTerm.setBeginOperator("lt");
     lTerm.setDateField("registereddate");
     lTerms = new SearchTerm[1];
     lTerms[0] = lTerm;
     criteria.setSearchTerms(lTerms);
     assets = repository.assetQuerySummary(authToken, criteria);
     //Search for assets where a given date field is within a date range
     ////////////////////////////////////////////////////////////////
     criteria = new AssetCriteria();
     lTerm = new DateRangeSearchTerm();
     ////////////////////////
     //date-range is the query key.  registereddate is the date field we are
searching on
     //Allowable fields to search on:
     //submitteddate
     //registereddate
     //acceptteddate
     //createddate
     //updateddate
     //The following SearchTerm translates to "Assets where the registereddate is
greater than or equal to Jan. 1, 2007
     lTerm.setKey("date-range");
     //The format defaults to OER's system setting for Short Date Format.
     //The format can be set to any valid date format using setDateFormat() and
passing the date format.
     lTerm.setDateField("registereddate");
     lTerm.setDateFormat("yyyy-MM-yy");
     lTerm.setBeginDate("2007-01-01");
     lTerm.setBeginOperator("gte");
     lTerm.setEndDate("2007-01-10");
     lTerm.setEndOperator("lte");
     //The following SearchTerm translates to "Assets where the registereddate is
less than or equal to Jan. 10, 2007
     criteria.setSearchTerms(new SearchTerm[] {lTerm});
     //This query returns all assets that were registered between January 1 and
January 10, including those days.
     assets = repository.assetQuerySummary(authToken, criteria);
     //Search for assets where a given tab has been approved or unapproved
     ////////////////////////////////////////////////////////////////
     criteria = new AssetCriteria();
     TabStatusSearchTerm lTabTerm = new TabStatusSearchTerm();
     //tabstatus is the type of search we want to do.  overview is the name of
the tab we want to search on
     lTabTerm.setKey("tabstatus");
     lTabTerm.setTabNames(new String[] {"overview"});
     lTabTerm.setApproved(true);
     criteria.setSearchTerms(new SearchTerm[] {lTabTerm});
     //This query returns all assets with the Overview tab being approved
     assets = repository.assetQuerySummary(authToken, criteria);
     //You may also search by a date range.
     lTabTerm.setKey("tabstatus");
     lTabTerm.setTabNames(new String[] {"overview"});
     lTabTerm.setApproved(false);
     lTabTerm.setBeginDate("2007-1-01");
     lTabTerm.setBeginOperator("lte");
     criteria.setSearchTerms(new SearchTerm[] {lTabTerm});
     //The following returns all assets that have the Overview tab unapproved
since or before January 1, 2007
     assets = repository.assetQuerySummary(authToken, criteria);
     //Search for assets where a custom field date has a specific value.
```

```
      //This test returns all assets that have a testdate of January 1, 2007.
      //The testdate field is a custom data date element.
      ////////////////////////////////////////////////////////////////
      criteria = new AssetCriteria();
      DateRangeSearchTerm lDateRangeTerm = new DateRangeSearchTerm();
      //Test Equals
      lDateRangeTerm.setKey("/asset/custom-data/testdate");
      lDateRangeTerm.setBeginDate("2007-01-1");
      lDateRangeTerm.setBeginOperator("eq");
      criteria.setSearchTerms(new SearchTerm[] {lDateRangeTerm});
      assets = repository.assetQuerySummary(authToken, criteria);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.6 Use Case: Upgrading asset status

Description

Stepping an asset through status levels, from Unsubmitted to Submitted to Accepted to Registered.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.KeyValuePair;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class PromoteAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
```

```
      ////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      long lAssetID = 559;
      // ------------------------------------------------------
      // asset with id 559 would have to be unsubmitted for this to work
      AssetCriteria lAssetCriteria = new AssetCriteria();
      lAssetCriteria.setIDCriteria(lAssetID);
      KeyValuePair lKeyValuePair = repository.assetEvaluate(authToken,
lAssetCriteria, "Registration Status");
      if (!lKeyValuePair.getValue().equalsIgnoreCase("unsubmitted")) {
        unregisterAsset(repository, authToken, lAssetID);
      }
      ////////////////////////////////////////////////////////
      // promote the asset from unsubmitted to submitted
      ////////////////////////////////////////////////////////
      repository.assetSubmit(authToken, lAssetID);
      // asset 559 would have to be unsubmitted for this to work
      ////////////////////////////////////////////////////////
      // promote the asset from submitted to accepted
      ////////////////////////////////////////////////////////
      repository.assetAccept(authToken, lAssetID);
      // asset 561 would have to be submitted for this to work
      ////////////////////////////////////////////////////////
      // promote the asset from accepted to registered
      ////////////////////////////////////////////////////////
      repository.assetRegister(authToken, lAssetID);
      // asset 563 would have to be accepted for this to work
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static void unregisterAsset(FlashlineRegistry repository, AuthToken
authToken, long pAssetID) {
    try {
      repository.assetUnRegister(authToken, pAssetID);
    } catch (Exception e) {
    }
    try {
      repository.assetUnAccept(authToken, pAssetID);
    } catch (Exception e) {
    }
    try {
      repository.assetUnSubmit(authToken, pAssetID);
    } catch (Exception e) {
    }
  }
}
```

15.2.7 Use Case: Downgrading asset status

Description

The reverse of the previous use case, stepping an asset through status levels, from Registered to Accepted to Submitted to Unsubmitted.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.KeyValuePair;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class DemoteAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      long lAssetID = 559;
      // ------------------------------------------------------
      // asset with id 559 would have to be registered for this to work
      AssetCriteria lAssetCriteria = new AssetCriteria();
      lAssetCriteria.setIDCriteria(lAssetID);
      KeyValuePair lKeyValuePair = repository.assetEvaluate(authToken,
 lAssetCriteria, "Registration Status");
      if (!lKeyValuePair.getValue().equalsIgnoreCase("registered")) {
        registerAsset(repository, authToken, lAssetID);
      }
      ////////////////////////////////////////////////////////////
      // demote the asset from registered to accepted
      ////////////////////////////////////////////////////////////
      repository.assetUnRegister(authToken, lAssetID);
      ////////////////////////////////////////////////////////////
      // demote the asset from accepted to submitted
      ////////////////////////////////////////////////////////////
      repository.assetUnAccept(authToken, lAssetID);
      ////////////////////////////////////////////////////////////
      // demote the asset from submitted to unsubmitted
      ////////////////////////////////////////////////////////////
      repository.assetUnSubmit(authToken, lAssetID);
    } catch (OpenAPIException lEx) {
```

```
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static void registerAsset(FlashlineRegistry repository, AuthToken
 authToken, long pAssetID) {
    try {
      repository.assetSubmit(authToken, pAssetID);
    } catch (Exception e) {
    }
    try {
      repository.assetAccept(authToken, pAssetID);
    } catch (Exception e) {
    }
    try {
      repository.assetRegister(authToken, pAssetID);
    } catch (Exception e) {
    }
  }
}
```

15.2.8 Use Case: Apply and remove Compliance Templates from a project

Description

Compliance Templates can be added and removed from multiple projects.

> **Note:** An OpenAPIException occurs if an asset is applied to a project and that asset is NOT a Compliance Template.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.AssetTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AddRemoveTemplate {
  public static void main(String pArgs[]) throws OpenAPIException,
```

```
RemoteException,
     ServiceException {
   try {
     URL lURL = null;
     lURL = new URL(pArgs[0]);
     ////////////////////////////////////////////////////////
     // Connect to Oracle Enterprise Repository
     ////////////////////////////////////////////////////////
     FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
         .getFlashlineRegistry(lURL);
     ////////////////////////////////////////////////////////
     // Login to OER
     ////////////////////////////////////////////////////////
     AuthToken authToken = repository.authTokenCreate(pArgs[1],
         pArgs[2]);
     ////////////////////////////////////////////////////////
     // Read or Create a Compliance Template Type and Asset
     ////////////////////////////////////////////////////////
     AssetType ctType = null;
     AssetTypeCriteria lAssetTypeCriteria = new AssetTypeCriteria();
     lAssetTypeCriteria.setArcheTypeCriteria("Compliance Template Type");
     AssetType[] lAssetTypes =
       repository.assetTypeQuery(authToken, lAssetTypeCriteria);
     if (lAssetTypes!=null && lAssetTypes.length>0) {
       ctType = lAssetTypes[0];
     } else {
       ctType = repository.assetTypeCreateComplianceTemplate(authToken,
           "My Compliance Template
Type"+Calendar.getInstance().getTimeInMillis());
     }
     Asset lComplianceTemplateAsset = null;
     AssetCriteria lAssetCriteria = new AssetCriteria();
     lAssetCriteria.setAssetTypeCriteria(ctType.getID());
     Asset[] lAssets = repository.assetQuery(authToken, lAssetCriteria);
     if (lAssets!=null && lAssets.length>0) {
       lComplianceTemplateAsset = lAssets[0];
     } else {
       lComplianceTemplateAsset = repository.assetCreate(authToken, "My
Compliance Template",
           ""+Calendar.getInstance().getTimeInMillis(), ctType.getID());
     }
     ////////////////////////////////////////////////////////
     // Create a String array of Project IDs that the Compliance
     // Template is applied to.
     ////////////////////////////////////////////////////////
     String[] lProjectIDs = { "50000" };
     // ////////////////////////////////////////////////////////
     // Apply template to the projects.
     // ////////////////////////////////////////////////////////
     repository.assetApplyToProjects(authToken, lProjectIDs,
         lComplianceTemplateAsset);
     ////////////////////////////////////////////////////////
     // Retrieve an array of Projects that this template is
     // applied to.
     ////////////////////////////////////////////////////////
     Project[] lProjects = repository.assetReadAppliedToProjects(
         authToken, lComplianceTemplateAsset);
     String lMsg = "Compliance Template '" + lComplianceTemplateAsset.getName();
     lMsg += "' applied to Project(s): ";
     for (int i=0; lProjects!=null && i<lProjects.length; i++) {
```

```
        lMsg += ""+lProjects[i].getName()+(i+1==lProjects.length ? "." : ", ");
      }
      System.out.println(lMsg);
      ////////////////////////////////////////////////////////
      // Create a String array of Project IDs that the Compliance
      // Template is removed from.
      ////////////////////////////////////////////////////////
      String[] lRemoveProjectIDs = { "50000" };
      ////////////////////////////////////////////////////////
      // Remove template from the projects.
      ////////////////////////////////////////////////////////
      repository.assetRemoveAppliedToProjects(authToken,
          lRemoveProjectIDs, lComplianceTemplateAsset);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.9 Use Case: Creating the new version of an asset and retiring the old version

Description

Update the repository to reflect the availability of a new version of an asset, and the retirement of the asset's previous version.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RelationshipType;
import com.flashline.registry.openapi.query.RelationshipTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewVersionOfAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
```

```
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////////////////////////
      // Read old asset.
      // Update metadata as necessary.
      // Save as new asset.
      ////////////////////////////////////////////////////////
      Asset myAsset = repository.assetRead(authToken, 561);
      ////////////////////////////////////////////////////////
      // Find the "next-version" relationship for the asset
      ////////////////////////////////////////////////////////
      RelationshipType[] allRelationshipTypes =
getAllRelationshipTypes(repository, authToken);
      for (int i = 0; i < allRelationshipTypes.length; i++) {
        if (allRelationshipTypes[i].getName().equals("next-version")) {
          ////////////////////////////////////////////////////////
          // This is the relationship type, modify the assets that are related
          // using it
          ////////////////////////////////////////////////////////
          RelationshipType myRelationshipType = allRelationshipTypes[i];
          ////////////////////////////////////////////////////////
          // Add the old version to list of previous versions of the
          // newly created asset
          ////////////////////////////////////////////////////////
          long[] oldSecondaryIDs = myRelationshipType.getSecondaryIDs();
          long[] newSecondaryIDs = new long[oldSecondaryIDs.length + 1];
          for (int j = 0; j < oldSecondaryIDs.length; j++) {
            newSecondaryIDs[j] = oldSecondaryIDs[j];
          }
          newSecondaryIDs[newSecondaryIDs.length - 1] = 561;
          myRelationshipType.setSecondaryIDs(newSecondaryIDs);
        }
      }
      Asset myNewAsset = repository.assetCreate(authToken,
          myAsset.getName(), ""+Calendar.getInstance().getTimeInMillis(),
myAsset.getTypeID());
      myNewAsset.setRelationshipTypes(allRelationshipTypes);
      ////////////////////////////////////////////////////////
      // Update the new asset
      ////////////////////////////////////////////////////////
      myNewAsset = repository.assetUpdate(authToken, myNewAsset);
      ////////////////////////////////////////////////////////
      // retire the old asset
      ////////////////////////////////////////////////////////
      repository.assetRetire(authToken, 561);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
```

```
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  /**
   * This method returns every relationship type in the repository
   * @param repository
   * @param authToken
   * @return
   * @throws RemoteException
   */
  public static RelationshipType[] getAllRelationshipTypes(FlashlineRegistry
 repository, AuthToken authToken) throws RemoteException {
    //Create an empty relationship type criteria object
    RelationshipTypeCriteria criteria = new RelationshipTypeCriteria();
    criteria.setNameCriteria("");
    RelationshipType[] allRelationshipTypes =
 repository.relationshipTypeQuery(authToken, criteria);
    return allRelationshipTypes;
  }
}
```

15.2.10 Use Case: "Housekeeping"

Description

Deleting groups of assets that no longer belong in the repository.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class DeleteAssets {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
```

```
            pArgs[1],pArgs[2]);
        ////////////////////////////////////////////////////////
        // find the assets to delete
        ////////////////////////////////////////////////////////
        AssetCriteria criteria = new AssetCriteria();
        criteria.setGeneralCriteria("delete me");
        Asset[] assets = repository.assetQuery(authToken, criteria);
        ////////////////////////////////////////////////////////
        // Iterate through assets, deleting them one at a time.
        ////////////////////////////////////////////////////////
        for (int i = 0; i < assets.length; i++) {
          repository.assetDelete(authToken, assets[i].getID());
        }
      } catch (OpenAPIException lEx) {
        System.out.println("ServerCode = " + lEx.getServerErrorCode());
        System.out.println("Message    = " + lEx.getMessage());
        System.out.println("StackTrace:");
        lEx.printStackTrace();
      } catch (RemoteException lEx) {
        lEx.printStackTrace();
      } catch (ServiceException lEx) {
        lEx.printStackTrace();
      } catch (MalformedURLException lEx) {
        lEx.printStackTrace();
      }
    }
}
```

Pitfalls:

Asset deletion is permanent. The OpenAPI provides no method for restoring deleted assets.

Methods to Avoid:

The following methods serve no purpose in the context of the OpenAPI, and should therefore be avoided:

- setAcceptedByID
- setAcceptedByName
- setAcceptedByDate
- setActiveStatus
- setAssigned
- setAssignedToID
- setAssignedDate
- setCategorizationTypes
- setCreatedByID
- setCreatedByName
- setCreatedByDate
- setDeleted
- setEntityType
- setExtractable

- setFullAsset
- setInactive
- setKey
- setLoadedDate
- setLongName
- setNotifyUpdatedRelationships
- setRegisteredByID
- setRegisteredByName
- setRegisteredDate
- setRegistrationStatus
- setRegistrationStatusBaseName
- setRegistrationStatusRegistered
- setRegistrationStatusRejected
- setRegistrationStatusSubmittedPendingReview
- setRegistrationStatusSubmittedUnderReview
- setRegistrationStatusUnsubmitted
- setRejectionReason
- setRetired
- setSubmittedByID
- setSubmittedByName
- setSubmittedDate
- setTypeIcon
- setTypeName
- setUpdatedDate
- setVendorName
- setVisible

Avoiding Common Mistakes

- Rules for Assets
  - The Asset must be assigned to an active and valid Asset Type.
  - An Asset's name/version strings must be a unique pair.
  - A new Asset's ID must be 0.
  - A new Asset's active status must be 'active'.

Missing Features

- Helper methods for modifying customData
- Additional validation

  When saving an asset, Oracle Enterprise Repository currently validates that:

  - The Asset type is valid and active

- # The Name/Version is unique
- When creating an asset, that the active status is valid
- When updating an asset, that the asset already exists
- Contacts are not duplicated
- Categorizations are valid
- Future versions of the repository validates that:
  * CustomData is well formed XML
  * CustomData contains XML that is valid based on the asset type

15.2.11 Use Case: Finding assets and updating custom-data

Description

Perform a search for all assets with a specific custom-data value, and update some custom-data for each of those assets. Note: The asset is automatically saved when using the assetUpdateCustomDataNode method.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetSummary;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.SearchTerm;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateAssetTestResults {
  public static void main(String pArgs[]) throws OpenAPIException,
      RemoteException, ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      ////////////////////////////////////////////////////////
      // create a criteria object searching for all assets with a
      // custom-data element for test-frequency equal to 'DAILY'
      ////////////////////////////////////////////////////////
      SearchTerm[] searchTermArray = new SearchTerm[1];
      SearchTerm term = new SearchTerm();
      term.setKey("/asset/custom-data/test-frequency");
```

```
      term.setValue("DAILY");
      searchTermArray[0] = term;
      AssetCriteria criteria = new AssetCriteria();
      criteria.setSearchTerms(searchTermArray);
      ////////////////////////////////////////////////////////
      // perform search, getting back summary objects. loop through
      // objects and perform an action on each one
      ////////////////////////////////////////////////////////
      AssetSummary[] assets = repository.assetQuerySummary(authToken,
          criteria);
      ////////////////////////////////////////////////////////
      // Loop through search results
      ////////////////////////////////////////////////////////
      for (int i = 0; i < assets.length; i++) {
        long assetID = assets[i].getID();
        String testResult = null;
        ////////////////////////////////////////////////////////
        // Update value in the asset
        ////////////////////////////////////////////////////////
        repository.assetUpdateCustomDataNode(
            authToken, assetID, "/asset/custom-data/test-result", testResult);
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.12 Use Case: Reading an Asset's Tabs

Description

Read the tabs of an asset.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.TabBean;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AssetReadTabs {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
```

```
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      TabBean[] lTabBeans = null;
      ///////////////////////////////////
      // Login to OER
      ///////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      ///////////////////////////////////
      // read an asset's tabs
      ///////////////////////////////////
      lTabBeans = repository.assetTabsRead(authToken, 559);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.13 Use Case: Retrieve An Asset's Tab Based on TabType

Description

Get a specific asset tab by tabtype.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.TabBean;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AssetGetTabByType {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
```

```
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      Asset lAsset = null;
      TabBean lTabBean = null;
      ////////////////////////////////////
      // Login to OER
      ////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      ////////////////////////////////////
      // read an asset
      ////////////////////////////////////
      lAsset = repository.assetRead(authToken, 559);
      ////////////////////////////////////
      // get an asset's tab by tabbeantype
      ////////////////////////////////////
      lTabBean = repository.assetTabRead(authToken, lAsset.getID(), 458);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

15.2.14 Use Case: Approving and Unapproving a tab

Description

Approve or unapprove an asset's tab.

Sample Code is as follows:

```
package com.flashline.sample.assetapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

public class ApproveUnapproveTab {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
```

```
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      ////////////////////////////////////
      // Login to OER
      ////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      ////////////////////////////////////
      // approve an asset tab
      ////////////////////////////////////
      repository.assetTabApprove(authToken, 559, 1864);
      ////////////////////////////////////
      // unapprove an asset tab
      ////////////////////////////////////
      repository.assetTabUnapprove(authToken, 559, 1864);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

16

AssetType API

This chapter provides an overview of AssetType API and describes the use cases using this API.

This chapter contains the following sections:

- Section 16.1, "Overview"
- Section 16.2, "Use Cases"

16.1 Overview

Types (Asset Types and Compliance Templates) define the structure of assets. Types consist of two main parts:

- **Editor**

  Defines the metadata that is stored for the assets and determines how metadata elements are organized in the Asset Editor.

- **Viewer**

  Defines how the metadata elements are displayed in the asset detail in Oracle Enterprise Repository.

When creating or editing a Type, acceptable value lists and categorization types are used as metadata elements. These metadata elements are referenced by ID in the Editor and Viewer XML for the Type.

When creating or editing assets, Types define the metadata elements that are used in the custom data for the asset (Asset.GetCustomData()).

Note:

- While the code examples in this section refer to Asset Types, please be aware that the processes described in the use cases are used to create both Asset Types and Compliance Templates. For more information about Compliance Templates, see *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository*.
- Editor and viewer metadata is represented as CDATA escaped XML. Consequently, if a large number of AssetTypes are returned through a call to assetTypeQuery, it is possible that some XML parsers may exceed their entity expansion limit. On some popular parsers, the default entity expansion limit is set to 64,000. If this limit is exceeded, it can be increased on JAXP compliant processors by passing a command-line parameter called entityExpanionLimit. For example, passing the following parameter to the JVM increases the entity expansion limit to 5 MB:

```
java -DentityExpansionLimit=5242880
com.example.MyApp
```

16.2 Use Cases

This section describes the use cases using the Asset Type API. It contains the following topics:

- Section 16.2.1, "Use Case: Create and edit a new Type"
- Section 16.2.2, "Use Case: Create a Compliance Template Type"
- Section 16.2.3, "Use Case: Find Types"
- Section 16.2.4, "Use Case: Read tab types"
- Section 16.2.5, "Use Case: Retrieve all Asset Type tabs"

16.2.1 Use Case: Create and edit a new Type

Description

Adding a new Type to the repository.

Sample code is as follows:

```
package com.flashline.sample.assettypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewAssetType {
  public static void main(String pArgs[]) throws RemoteException,
 OpenAPIException,
      ServiceException {
```

```
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ////////////////////////////////
      // Select an existing Type on which to base the new one
      // ////////////////////////////////
      String newAssetTypeName = "My AssetType";
      long baseAssetTypeID = 151;
      AssetType newAssetType = repository.assetTypeCreateClone(
          authToken, baseAssetTypeID, newAssetTypeName);
      System.out.println("The new Asset Type id =\"" + newAssetType.getID()
          + "\"");
      // ////////////////////////////////
      // Manipulate xml strings
      // ////////////////////////////////
      String lEditorXML = newAssetType.getEditorXML();
      String lViewerXML = newAssetType.getViewerXML();
      // Perform XML manipulation on the editor and viewer definitions...
      // ////////////////////////////////
      // Set the new editor/viewer definitions on the asset type, and save the
      // type back to OER
      // ////////////////////////////////
      newAssetType.setEditorXML(lEditorXML);
      newAssetType.setViewerXML(lViewerXML);
      repository.assetTypeUpdate(authToken, newAssetType);
      // ----------------------------------------
      // clean up sample
      repository.assetTypeDelete(authToken, newAssetType.getID());
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

16.2.2 Use Case: Create a Compliance Template Type

Description

Adding a new Compliance Template Type to the repository.

Sample code is as follows:

```
package com.flashline.sample.assettypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewComplianceTemplateType {
  public static void main(String pArgs[]) throws RemoteException,
 OpenAPIException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(
          pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ///////////////////////////////
      // Create a new compliance template.
      // ///////////////////////////////
      String newAssetTypeName = "My Compliance
 Template"+Calendar.getInstance().getTimeInMillis();
      AssetType newAssetType = repository
          .assetTypeCreateComplianceTemplate(authToken, newAssetTypeName);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

16.2.3 Use Case: Find Types

Description

Locating a Type in the repository.

Sample Code is as follows:

```
package com.flashline.sample.assettypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetTypeCriteria;
import com.flashline.registry.openapi.query.SearchTerm;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class FindAssetType {
  public static void main(String pArgs[]) throws RemoteException,
 OpenAPIException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ///////////////////////////////
      // Create SearchTerms and set them on the AssetSearchCriteria
      // ///////////////////////////////
      AssetTypeCriteria assetTypeCriteria = new AssetTypeCriteria();
      SearchTerm[] searchTerms = new SearchTerm[1];
      searchTerms[0] = new SearchTerm();
      searchTerms[0].setKey("name");
      searchTerms[0].setValue("Component");
      assetTypeCriteria.setSearchTerms(searchTerms);
      // ///////////////////////////////
      // Perform the search using the specified criteria
      // ///////////////////////////////
      AssetType[] assetTypes = repository.assetTypeQuery(authToken,
          assetTypeCriteria);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

Methods to Avoid:

- setIcon
- setID

16.2.4 Use Case: Read tab types

Description

Retrieve the list of tabs available for an asset type.

Sample Code is as follows:

```
package com.flashline.sample.assettypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.TabTypeBean;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ReadTabTypes {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      TabTypeBean[] lTabTypeBeans = null;
      ////////////////////////////////////
      // Login to OER
      ////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      ////////////////////////////////////
      // read the tab types of an assettype
      ////////////////////////////////////
      lTabTypeBeans = repository.assetTypeTabsRead(authToken, 100);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

16.2.5 Use Case: Retrieve all Asset Type tabs

Description

Retrieves all asset type tabs within the Oracle Enterprise Repository.

Sample Code is as follows:

```
package com.flashline.sample.assettypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.TabTypeBean;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ReadTabTypes {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      TabTypeBean[] lTabTypeBeans = null;
      ////////////////////////////////////
      // Login to OER
      ////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      ////////////////////////////////////
      // read the tab types of an assettype
      ////////////////////////////////////
      lTabTypeBeans = repository.assetTypeTabsRead(authToken, 100);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

Example of the RelationshipTypeQuery

```
try
{
   RelationshipTypeCriteria rCriteria = new RelationshipTypeCriteria();
```

```
   RelationshipType[] allRelationshipTypes =
 FlashlineRegistry.relationshipQuery(lAuthToken, rCriteria);
}
catch (OpenAPIException e)
{
     e.printStackTrace();
}
catch (RemoteException re)
{
    re.printStackTrace();
}
```

17

Categorization Types and Categorizations API

This chapter provides an overview of Categorization Types and Categorizations API and describes the use cases using this API.

This chapter contains the following sections:

- Section 17.1, "Overview"
- Section 17.2, "Use Cases"

17.1 Overview

It is important to understand the difference between Categorizations and Categorization Types.

Categorization Types provide the means to define custom taxonomies. Categorizations are subsets of Categorization Types, and are assigned to assets. For example, a Categorization Type called Technology might contain Java, .NET and COBOL as its assignable Categorizations. Additionally, sub-categorizations under Java might include Servlet and Applet. So an asset in Oracle Enterprise Repository might be assigned to the Java Categorization, or it might be more specifically assigned to the Servlet Sub-categorization.

The Categorizations to which a particular asset may be assigned are determined by the Categorization Types defined for that asset's Type. As in the example above, if the Technology Categorization Type is defined for Asset Type XYZ, assets of type XYZ may be assigned to the Java, .NET, or COBOL, Categorizations, or to the Sub-categorizations Servlet or Applet. This taxonomy structure allows assets to be grouped and viewed in a variety of ways within Oracle Enterprise Repository.

Categorization types can also be associated with projects (if Oracle Enterprise Repository is configured for that feature). Any project-assignable Categorization Type is available to all projects. As with assets, a project can be associated with any of the Categorizations available within its assigned Categorization Type(s).

Rules for Categorization Types and Categorizations include:

- The Repository can contain 0 to n Categorization Types with each Categorization Type containing 0 to n Categorizations.
- Each Categorization can contain 0 to n Sub-categorizations.
- Each Categorization Type must have a unique name. This name cannot contain spaces or special characters.

- As an option, Categorizations within a given Categorization Type may be made mutually exclusive. That is, when a list of Categorizations is presented, only one may be selected.
- When the Mutually Exclusive option is selected for a Categorization Type, Oracle Enterprise Repository enforces the rule for future usage only. Existing references to multiple selected categorizations within the Type are unchanged.
- When so configured, Categorization Types can be assigned to projects. This allows projects in Oracle Enterprise Repository to be organized by Categorization Type/Categorization.
- If the configuration of a specific Categorization Type is changed to prevent its assignment to Projects, the change affects only subsequent Project assignment. The change does not affect the Categorization Type's assignment to existing Projects.
- Categorization Types may be deleted from Oracle Enterprise Repository. However, doing so also deletes all categorizations within the deleted Categorization Type. Exercise caution when performing this task.
- Categorizations may be deactivated. Deactivation prevents future use of the Categorization (and all sub-categorizations) but does not delete it from Oracle Enterprise Repository. Existing references to a Categorization are unaffected by deactivation.
- Deactivated Categorizations may be reactivated, reversing the aforementioned process.
- Within a given Categorization Type, all Categorizations must be uniquely named. However, the same name may be shared by multiple Categorizations residing in different Categorization Types.

The following methods provide the ability to create, update, list, query, and delete categorization types.

17.2 Use Cases

This section describes the use cases using the Categorization Types and Categorizations API. It contains the following topics:

- Section 17.2.1, "Use Case: Create a Categorization Type"
- Section 17.2.2, "Use Case: Manipulate Categorization Types"
- Section 17.2.3, "Use Case: Manipulate Categorizations"

17.2.1 Use Case: Create a Categorization Type

Description

The element name given to a newly created Categorization Type cannot contain special characters or spaces. There are no restrictions on the characters used for singular and plural display names. The pExclusiveAssign Boolean determines whether one or multiple Categorizations can be assigned within the Categorization Type. The pExclusiveAssign Boolean determines if the Categorization Type is project-assigned. The method prevents duplication of existing Categorizations, and returns the created Categorization Type.

Sample code is as follows:

```
package com.flashline.sample.categorizationtypesandapi;
```

```
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateCategorizationType {
  public static void main(String pArgs[]) throws java.rmi.RemoteException,
      OpenAPIException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ///////////////////////////////
      // Create the categorization type
      // ///////////////////////////////
      String pName = "Name";
      String pSingularDisplay = "SingularDisplay";
      String pPluralDisplay = "PluralDisplay";
      boolean pExclusiveAssign = true;
      boolean pProjectAssign = true;
      CategorizationType lCategorizationType =
 repository.categorizationTypeCreate(authToken, pName,
          pSingularDisplay, pPluralDisplay, pExclusiveAssign, pProjectAssign);
      // ---------------------------------------
      // clean up
      repository.categorizationTypeDelete(authToken, lCategorizationType);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

17.2.2 Use Case: Manipulate Categorization Types

Description

The following operations are demonstrated in the example below:

- Retrieve Categorization Types by ID
- Update a Categorization Type
- Delete a Categorization Type
- Excercise Caution! This method deletes the entire Categorization Type and all Categorizations contained therein.
- Query a Categorization Type
- Use various terms, including name, type, and if a Categorization Type is assigned to projects.
- Retrieve all Categorization Types

Sample code is as follows:

```
package com.flashline.sample.categorizationtypesandapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Categorization;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.query.CategorizationTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CategorizationExamples {
  public static void main(String pArgs[]) throws ServiceException,
 RemoteException,
      OpenAPIException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ////////////////////////////////
      // Create a Categorization Type
      // ////////////////////////////////
      CategorizationType categorizationType = repository
          .categorizationTypeCreate(authToken, "exampleType", "Example Type",
              "Example Types", false, true);
      // ////////////////////////////////
      // Find and update a categorization type
      // ////////////////////////////////
```

```
        CategorizationTypeCriteria categorizationTypeCriteria = new
    CategorizationTypeCriteria();
        boolean projectAssign = true;
        categorizationTypeCriteria.setProjectAssignCriteria(projectAssign + "");
        CategorizationType[] categorizationTypes = repository
            .categorizationTypeQuery(authToken, categorizationTypeCriteria);
        // Set plural display name
        categorizationType.setDisplayPlural("Updated Example Types");
        // Set singular display name
        categorizationType.setDisplaySingular("Updated Example Type");
        // Set Categorization Type name
        categorizationType.setName("updatedExampleType");
        // Set Categorization Type exclusive assign
        categorizationType.setExclusiveAssign(true);
        // Set Categorization Type project Assignable
        categorizationType.setProjectAssignable(false);
        // Update Categorization Type
        categorizationType = repository.categorizationTypeUpdate(
            authToken, categorizationType);
        // Read a Categorization Type
        CategorizationType categorizationTypeRead = repository
            .categorizationTypeRead(authToken, categorizationType.getID());
        // ////////////////////////////
        // Create a Categorization within a Categorization Type
        // ////////////////////////////
        Categorization categorization = repository.categorizationCreate(
            authToken, "Example Categorization", categorizationType);
        // ////////////////////////////
        // Create a Categorization within a Categorization (a.k.a. a
        // sub-categorization or child categorization)
        //
        // method validates that:
        //   - no child categorization with the same name exists within the
        //     parent categorization.
        //   - the categorization type is valid
        //   - the parent categorization is valid.
        // ////////////////////////////
        Categorization childCategorization = repository
            .categorizationChildCreate(authToken, "Example Child Categorization",
                categorizationType, categorization);
        childCategorization.setName("Updated Example Child Categorization");
        childCategorization = repository.categorizationUpdate(authToken,
            childCategorization, categorizationType);
        // ////////////////////////////
        // Observe various properties of a categorization. Note that the
        // properties are not being assigned to local variables in
        // the interest of brevity...
        // ////////////////////////////
        Categorization categorizationRead = repository.categorizationRead(
            authToken, childCategorization.getID());
        // Get Categorization parent id
        //categorizationRead.getParentID();
        // Get Categorization active status
        categorizationRead.getActiveStatus();
        // Get Categorization ID
        categorizationRead.getID();
        // Get Categorization name
        categorizationRead.getName();
        // Get Categorization recursive name
        categorizationRead.getRecursiveName();
```

```
      // Get Categorization Categorization Type ID
      categorizationRead.getTypeID();
      // ////////////////////////////
      // Retrieve the full tree of categorizations for a Categorization Type.
      // This means that the Categorization entity returned has children
      // which contain their children (if any), and so on.
      // ////////////////////////////
      // Get active Categorizations full tree
      boolean active = true;
      boolean fullTree = true;
      Categorization[] categorizationArray = repository.categorizationReadByType(
          authToken, categorizationType, active, fullTree);
      // Get children categorizations for categorization
      Categorization[] childCategorizations =
 repository.categorizationChildRead(authToken,
          categorizationType, categorization, active);
      // ////////////////////////////
      // Deactivate a Categorization
      // ////////////////////////////
      categorization = repository.categorizationDeactivate(authToken,
          categorization, categorizationType);
      // pActive is set to "true" so that the method returns
      // only active categorizations
      Categorization[] activeCategorizations = repository
          .categorizationChildRead(authToken, categorizationType,
              categorization, true);
      // Get inactive child Categorizations
      Categorization[] inactiveCategorizations =
 repository.categorizationChildRead(authToken,
          categorizationType, categorization, false);
      // ////////////////////////////
      // Reactivate Categorizations
      // ////////////////////////////
      for(int i=0; i<inactiveCategorizations.length; i++){
        categorization = repository.categorizationReactivate(authToken,
            inactiveCategorizations[i], categorizationType);
      }
      // ////////////////////////////
      // Delete a Categorization Type
      // ////////////////////////////
      repository.categorizationTypeDelete(authToken, categorizationType);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

Methods to Avoid

The methods listed below are for internal Oracle Enterprise Repository use only. Incorrect use of these methods may disrupt the functionality of Categorization Types

(though permanent damage is unlikely) . The functionality provided by these methods is incorporated in the Oracle Enterprise Repository CategorizationType methods.

- getActiveStatus() int - CategorizationType
- getCategorizations() Categorizations[] - CategorizationType
- GetEntityType() String - CategorizationType
- getKey() String - CategorizationType
- getTypeDesc() TypeDesc - CategorizationType
- hashCode() int - CategorizationType
- isCustom() boolean - CategorizationType
- isFlat() boolean - CategorizationType
- isSystemOnly() boolean - CategorizationType
- setActiveStatus(int activeStatus) void - CategorizationType
- setAssetAssignable(boolean assetAssignable) void - CategorizationType
- setCategorizations(Categorizations[] categorizations) void - CategorizationType
- setCustom(boolean custom) void - CategorizationType
- setEntityType(String entityType) void - CategorizationType
- setFlat(boolean flat) void - CategorizationType
- setID(long ID) void - CategorizationType
- setKey(String key) void - CategorizationType
- setSystemOnly(boolean systemOnly) void - CategorizationType

17.2.3 Use Case: Manipulate Categorizations

Description

The following code sample illustrates creation, updating, listing, and deactivation/reactivation of Categorizations. As stated above, Categorizations are subordinate to Categorization Types. That is, a Categorization belongs to a Categorization Type.

Sample Code is as follows:

```
package com.flashline.sample.categorizationtypesandapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Categorization;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.query.CategorizationTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CategorizationExamples {
  public static void main(String pArgs[]) throws ServiceException,
 RemoteException,
```

```
      OpenAPIException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ////////////////////////////////
      // Create a Categorization Type
      // ////////////////////////////////
      CategorizationType categorizationType = repository
          .categorizationTypeCreate(authToken, "exampleType", "Example Type",
              "Example Types", false, true);
      // ////////////////////////////////
      // Find and update a categorization type
      // ////////////////////////////////
      CategorizationTypeCriteria categorizationTypeCriteria = new
CategorizationTypeCriteria();
      boolean projectAssign = true;
      categorizationTypeCriteria.setProjectAssignCriteria(projectAssign + "");
      CategorizationType[] categorizationTypes = repository
          .categorizationTypeQuery(authToken, categorizationTypeCriteria);
      // Set plural display name
      categorizationType.setDisplayPlural("Updated Example Types");
      // Set singular display name
      categorizationType.setDisplaySingular("Updated Example Type");
      // Set Categorization Type name
      categorizationType.setName("updatedExampleType");
      // Set Categorization Type exclusive assign
      categorizationType.setExclusiveAssign(true);
      // Set Categorization Type project Assignable
      categorizationType.setProjectAssignable(false);
      // Update Categorization Type
      categorizationType = repository.categorizationTypeUpdate(
          authToken, categorizationType);
      // Read a Categorization Type
      CategorizationType categorizationTypeRead = repository
          .categorizationTypeRead(authToken, categorizationType.getID());
      // ////////////////////////////////
      // Create a Categorization within a Categorization Type
      // ////////////////////////////////
      Categorization categorization = repository.categorizationCreate(
          authToken, "Example Categorization", categorizationType);
      // ////////////////////////////////
      // Create a Categorization within a Categorization (a.k.a. a
      // sub-categorization or child categorization)
      //
      // method validates that:
      //   - no child categorization with the same name exists within the
      //     parent categorization.
      //   - the categorization type is valid
      //   - the parent categorization is valid.
      // ////////////////////////////////
```

```
        Categorization childCategorization = repository
            .categorizationChildCreate(authToken, "Example Child Categorization",
                categorizationType, categorization);
        childCategorization.setName("Updated Example Child Categorization");
        childCategorization = repository.categorizationUpdate(authToken,
            childCategorization, categorizationType);
        // ////////////////////////////
        // Observe various properties of a categorization. Note that the
        // properties are not being assigned to local variables in
        // the interest of brevity...
        // ////////////////////////////
        Categorization categorizationRead = repository.categorizationRead(
            authToken, childCategorization.getID());
        // Get Categorization parent id
        //categorizationRead.getParentID();
        // Get Categorization active status
        categorizationRead.getActiveStatus();
        // Get Categorization ID
        categorizationRead.getID();
        // Get Categorization name
        categorizationRead.getName();
        // Get Categorization recursive name
        categorizationRead.getRecursiveName();
        // Get Categorization Categorization Type ID
        categorizationRead.getTypeID();
        // ////////////////////////////
        // Retrieve the full tree of categorizations for a Categorization Type.
        // This means that the Categorization entity returned has children
        // which contain their children (if any), and so on.
        // ////////////////////////////
        // Get active Categorizations full tree
        boolean active = true;
        boolean fullTree = true;
        Categorization[] categorizationArray = repository.categorizationReadByType(
            authToken, categorizationType, active, fullTree);
        // Get children categorizations for categorization
        Categorization[] childCategorizations =
    repository.categorizationChildRead(authToken,
            categorizationType, categorization, active);
        // ////////////////////////////
        // Deactivate a Categorization
        // ////////////////////////////
        categorization = repository.categorizationDeactivate(authToken,
            categorization, categorizationType);
        // pActive is set to "true" so that the method returns
        // only active categorizations
        Categorization[] activeCategorizations = repository
            .categorizationChildRead(authToken, categorizationType,
                categorization, true);
        // Get inactive child Categorizations
        Categorization[] inactiveCategorizations =
    repository.categorizationChildRead(authToken,
            categorizationType, categorization, false);
        // ////////////////////////////
        // Reactivate Categorizations
        // ////////////////////////////
        for(int i=0; i<inactiveCategorizations.length; i++){
          categorization = repository.categorizationReactivate(authToken,
              inactiveCategorizations[i], categorizationType);
        }
```

```
      // ////////////////////////////
      // Delete a Categorization Type
      // ////////////////////////////
      repository.categorizationTypeDelete(authToken, categorizationType);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

Methods to Avoid:

The methods listed below are for internal Oracle Enterprise Repository use only and should not be used. Incorrect use of these methods could cause improper functioning of categorizations. The functions provided by these methods provide are incorporated in the Oracle Enterprise Repository categorization methods.

- `getDescription() String - Categorization`
- `GetEntityType() String - Categorization`
- `getKey() String - Categorization`
- `getLevel() long - Categorization`
- `getType() CategorizationType - Categorization`
- `getTypeDesc() TypeDesc - Categorization`
- `hashCode() int - Categorization`
- `set_super(Categorization _super) void - Categorization`
- `setActiveStatus(int activeStatus) void - Categorization`
- `setDeleted(boolean deleted) void - Categorization`
- `setDescription(String description) void - Categorization`
- `setEntityType(String entityType) void - Categorization`
- `setID(long ID) void - Categorization`
- `setKey(String key) void - Categorization`
- `setRecursiveName(String recursiveName) void - Categorization`
- `setType(CategorizationType type) void - Categorization`
- `setTypeID(long ID) void - Categorization`

18

CMF Entry Type API

This chapter provides an overview of CMF Entry Type API and describes the use cases using this API.

This chapter contains the following sections:

- Section 18.1, "Overview"
- Section 18.2, "Use Cases"

18.1 Overview

CMF Entry Types describe metadata that may be attached to assets. CMF Entry Types are identified by an id and a single name string.

Validation - When saving a CMF Entry Type, Oracle Enterprise Repository currently validates that:

- The CMF Entry Type name length is in bounds
- The CMF Entry Type name is unique
- When updating a CMF Entry Type, a CMF Entry Type ID is present

Related Subsystems

A CMF Entry is linked to an asset from the perspective of the asset. CMF Entry Types define parameters for these entries.

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.MetadataEntryTypeSummary;
import com.flashline.registry.openapi.query.MetadataEntryTypeCriteria;
```

18.2 Use Cases

This section describes the use cases using the CMF Entry Type API. It contains the following topics:

- Section 18.2.1, "Use Case: Manipulating CMF Entry Types"

18.2.1 Use Case: Manipulating CMF Entry Types

Description

- Adding a new CMF Entry Type to Oracle Enterprise Repository.

- Assigning an existing CMF Entry Type to an asset.

Sample code is as follows:

```
package com.flashline.sample.metadataentrytypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.MetadataEntryTypeSummary;
import com.flashline.registry.openapi.query.MetadataEntryTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class MetadataEntryTypes {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      // ----------------------------------------
      // Create a new CMF Entry Type
      String newMetadataEntryTypeName = "Sample MetadataEntryType";
      MetadataEntryTypeSummary newMetadataEntryType =
 repository.metadataEntryTypeCreate(authToken, newMetadataEntryTypeName);
      System.out.println("The new MetadataEntryType id =\"" +
 newMetadataEntryType.getID() + "\"");
      // ----------------------------------------
      // Find a CMF Entry Type
      MetadataEntryTypeCriteria criteria = new MetadataEntryTypeCriteria();
      criteria.setNameCriteria("Sample");
      MetadataEntryTypeSummary[] metadataEntryTypes =
 repository.metadataEntryTypeQuery(authToken, criteria);
      long myMetadataEntryTypeID = metadataEntryTypes[0].getID();
      // ----------------------------------------
      // Read a CMF Entry Type
      MetadataEntryTypeSummary readMetadataEntryType =
 repository.metadataEntryTypeRead(authToken, myMetadataEntryTypeID);
      System.out.println("The MetadataEntryType name =\"" +
 readMetadataEntryType.getName() + "\"");
      // ----------------------------------------
      // Delete a CMF Entry Type
      repository.metadataEntryTypeDelete(authToken, myMetadataEntryTypeID);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
```

```
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

19

Custom Access Settings API

This chapter provides an overview of Custome Access Settings API and describes the use cases using this API.

This chapter contains the following sections:

- Section 19.1, "Overview"
- Section 19.2, "Use Cases"

19.1 Overview

The Custom Access Settings Subsystem is used to provide a Web Services-based mechanism to retrieve Oracle Enterprise Repository Custom Access Settings (CAS). Example 19–1 describes a sample Custom Access Settings code.

Example 19–1 Example of Custom Access Settings Code

Working code for Custom Access Settings Open API methods.

```
package com.flashline.sample.customaccesssettingsapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CustomAccesssettings {
  public static void main(String pArgs[]) throws java.rmi.RemoteException,
      OpenAPIException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      // ////////////////////////////////////////////////////////
```

```
        // Retrieve a List of Custom Access Setting Types
        // ////////////////////////////////////////////////////////////
        String[] lRoleContextTypes = null;
        lRoleContextTypes = repository
            .customAccessSettingTypesGet(authToken);
        // ////////////////////////////////////////////////////////////
        // Get Custom Access Setting Names
        // ////////////////////////////////////////////////////////////
        String[] lCustomAccessSettingNames = null;
        lCustomAccessSettingNames = repository
            .customAccessSettingNamesGet(authToken, "asset");
        // ////////////////////////////////////////////////////////////
        // Retrieve an array of Custom Access Setting Names of type "asset".
        // ////////////////////////////////////////////////////////////
        String[] rCustomAccessSettingNames = null;
        rCustomAccessSettingNames = repository
            .customAccessSettingDefaultNamesGet(authToken, "asset");
      } catch (OpenAPIException lEx) {
        System.out.println("ServerCode = " + lEx.getServerErrorCode());
        System.out.println("Message    = " + lEx.getMessage());
        System.out.println("StackTrace:");
        lEx.printStackTrace();
      } catch (RemoteException lEx) {
        lEx.printStackTrace();
      } catch (ServiceException lEx) {
        lEx.printStackTrace();
      } catch (MalformedURLException lEx) {
        lEx.printStackTrace();
      }
    }
}
```

19.2 Use Cases

This section describes the use cases using the Custom Access Settings API. It contains the following topics:

- Section 19.2.1, "Use Case: Retrieve a List of Custom Access Setting Types"
- Section 19.2.2, "Use Case: Get Default Custom Access Setting Names"

19.2.1 Use Case: Retrieve a List of Custom Access Setting Types

Description

This method is used to retrieve the list of Custom Access Settings Types as available in Oracle Enterprise Repository.

Sample code is as follows:

```
1.      String[] lCustomAccessSettingNames = null;
2.      lCustomAccessSettingNames =
 mFlashlineRegistry.customAccessSettingNamesGet(mAuthToken, "asset");
```

Annotations

Line 2 - Retrieve an array of Custom Access Setting Names of type "asset".

19.2.2 Use Case: Get Default Custom Access Setting Names

Description

This method is used to retrieve the list of Default Custom Access Settings of a particular type. An asset default Custom Access Setting is applied to all new assets, just as a file default Custom Access setting is applied to all new files, and so on.

Sample code is as follows:

```
1.      String[] lCustomAccessSettingNames = null;
2.      lCustomAccessSettingNames =
 mFlashlineRegistry.customAccessSettingDefaultNamesGet(mAuthToken, "asset");
```

Annotations

Line 2 - Retrieve an array of default Custom Access Setting Names of type "asset".

20

Department API

This chapter provides an overview of Department API and describes the use cases using this API.

This chapter contains the following sections:

- Section 20.1, "Overview"
- Section 20.2, "Use Cases"

20.1 Overview

Departments can be created, read, queried for, and modified. These operations are described below. Bear in mind that once a Department is created, it cannot be deleted. Only two Department attributes are meaningful to a user: name and description.

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.Department;
```

20.2 Use Cases

This section describes the use cases using the Department API. It contains the following topics:

- Section 20.2.1, "Use Case: Manipulate Departments"

20.2.1 Use Case: Manipulate Departments

Description

The following sample code illustrates typical tasks involving the manipulation of departments in Oracle Enterprise Repository. This includes creation, updating, querying, and deleting.

Sample code is as follows:

```
package com.flashline.sample.departmentapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
```

```
import com.flashline.registry.openapi.entity.Department;
import com.flashline.registry.openapi.query.DepartmentCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class Departments {
  public static void main(String pArgs[]) throws java.rmi.RemoteException,
      OpenAPIException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ////////////////////////////////
      // Create a new department
      // Each Department requires a unique name. Descriptions are optional.
      // ////////////////////////////////
      Department dept = repository.departmentCreate(authToken,
          "My Dept "+Calendar.getInstance().getTimeInMillis(), "A New
 Department");
      // ////////////////////////////////
      // Read a department
      // To read a Department you must have the Department name.
      // ////////////////////////////////
      Department dept2 = repository.departmentRead(authToken,
          "ADepartment");
      // ////////////////////////////////
      // Query for a department
      //
      // To query for a Department you must fill out a
      // DepartmentCriteria object with an array of SearchTerms. A SearchTerm
      // is a key/value pair. Currently the only valid key is "name".
      //
      // A query for name is a match if the value for the name term
      // occurs anywhere in the name of the department. For example,
      // a search for fred matches fred, alfred, and fredrick.
      // ////////////////////////////////
      DepartmentCriteria criteria = new DepartmentCriteria();
      criteria.setNameCriteria("DepartmentName");
      Department[] depts = repository.departmentQuery(authToken,
          criteria);
      // ////////////////////////////////
      // Update a department
      //
      // To update a Department you need only to modify a Department
      // reference and call departmentUpdate...
      // ////////////////////////////////
      String lOldName = dept.getName();
      String lNewName = "New " + dept.getName();
      Department dept3 = repository.departmentRead(authToken, lOldName);
      dept3.setName(lNewName);
      repository.departmentUpdate(authToken, dept3);
```

```
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

21

Extraction API

This chapter provides an overview of Extraction API and describes the use cases using this API.

This chapter contains the following sections:

- Section 21.1, "Overview"
- Section 21.2, "Use Cases"

21.1 Overview

As part of the Use - Download (extraction) process the user is asked to associate the selected asset with a particular project. These instances of usage, along with surveys (which are included in usage updates) are the primary drivers for metrics within Oracle Enterprise Repository.

> **Note:** In earlier product releases the term Extraction was used to describe the act of downloading or otherwise accessing an asset's payload. The term Extraction has since been replaced by the phrase Use - Download. Please note, however, that within the context of this Extraction API document, most instances of use of the term Extraction (particularly in code examples) were left intact to simplify the use of REX API.

Definitions

- **State**

  State refers to the usage status of an asset that has been selected for use/download. There are four possible states:

  - IN PROCESS
  - ACCEPTED
  - REJECTED
  - DEPLOYED (DEPLOYED is covered under Projects).

- **Extraction Download**

  Contains the file info associated with the extracted asset. Values for an extraction download can be 0 or 1.

- **File Info**

Information and URL links to the actual files associated with the asset make up the File Info as contained in an extraction download. File Info values for an extraction download can be 0 to n.

- **Related Asset**

  Within Oracle Enterprise Repository, a given asset can be associated with others through a number of pre-defined and/or custom-configured relationships. An asset can contain 0 to n related assets.

Related Subsytems

- AssetSubsystem
- ProjectSubsystem
- CategorizationTypeSubsystem
- SurveySubsystem

21.2 Use Cases

This section describes the use cases using the Extraction API. It contains the following topics:

- Section 21.2.1, "Use Case: Extract an Asset"
- Section 21.2.2, "Use Case: Read an Extraction"
- Section 21.2.3, "Use Case: Update an Extraction"

21.2.1 Use Case: Extract an Asset

Description

An Extraction is created when an asset is associated for use in a project by the user. A list of related assets is also made available for extraction during this process. In this case the user can simultaneously extract both the primary asset and any related assets. A unique extraction is then recorded for each asset. Creating an extraction results in an array containing 0 to n extraction downloads. The file info value for each download can be 0 to n. The file info contains information about the file. This information is used to create a link to the file.

To extract an asset the following conditions must be met:

- The user must be a member of the project to which the asset is to be extracted.
- The user must be assigned the appropriate role type(s).
- The project must be open.
- The asset(s) must be registered and active.
- If Custom Access Settings are enabled, the user performing the extraction must have appropriate access rights to the specified asset(s).
- If Custom Access Settings are enabled, the user performing the extraction receives file info only for those files to which the user has the appropriate permissions.
- These conditions are checked by the appropriate methods in which they are used. Exceptions are thrown if the conditions are not met.

Sample code is as follows:

```
package com.flashline.sample.extractionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.ExtractionDownload;
import com.flashline.registry.openapi.entity.FileInfo;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ExtractAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1], pArgs[2]);
      long ASSET_ID_1 = 589;     // must be a valid asset id in OER
      long ASSET_ID_2 = 569;     // must be a valid asset id in OER
      long PROJECT_ID = 50000;   // must be a valid project id in OER
      long EXTRACTION_ID = 0;
      // ---------------------------------------
      // Create a new extraction
      long[] lAssetIDs = { ASSET_ID_1, ASSET_ID_2 };
      ExtractionDownload[] extractionDownloads =
 repository.extractionCreate(authToken, PROJECT_ID, lAssetIDs);
      System.out.println("Number of new extraction downloads created: " +
 extractionDownloads.length);
      // ---------------------------------------
      // Read an extraction by project and asset
      Extraction extraction =
 repository.extractionReadByProjectAndAsset(authToken, PROJECT_ID, ASSET_ID_1);
      EXTRACTION_ID = extraction.getID();
      // ---------------------------------------
      // Read an extraction by ID
      Extraction extractionByID = repository.extractionRead(authToken, EXTRACTION
_ID);
      System.out.println("The extraction '"+extractionByID.getDisplayName()+"' was
 read by id ("+EXTRACTION_ID+")");
      // ---------------------------------------
      // Read asset extractions
      Extraction[] assetExtractions =
```

```
repository.extractionReadAssetExtractions(authToken, PROJECT_ID, ASSET_ID_1,
true);
     System.out.println("The number of extractions for this asset is:
"+(assetExtractions==null ? 0 : assetExtractions.length));
     // ----------------------------------------
     // Read project extractions
     Extraction[] projectExtractions =
repository.extractionReadProjectExtractions(authToken, PROJECT_ID, true);
     System.out.println("The number of extractions for this project is:
"+(projectExtractions==null ? 0 : projectExtractions.length));
     // ----------------------------------------
     // Read related assets
     Asset[] relatedAssets = repository.extractionReadRelatedAssets(authToken,
ASSET_ID_2);
     System.out.println("The number of related assets is: "+relatedAssets==null ?
0 : relatedAssets.length);
     // ----------------------------------------
     // Read File-Info for an extraction
     List fileInfosList = new ArrayList();
     if (projectExtractions != null) {
       for (int i = 0; i < projectExtractions.length; i++) {
         extraction = repository.extractionRead(authToken,
projectExtractions[i].getID());
         fileInfosList.add(repository.extractionReadFileInfos(authToken,
extraction));
       }
     }
     // ----------------------------------------
     // Get File
     List fileInfoList = new ArrayList();
     Iterator fileInfosListIter = fileInfosList.iterator();
     while (fileInfosListIter.hasNext()) {
       FileInfo[] fileInfos = (FileInfo[]) fileInfosListIter.next();
       for (int i = 0; i < fileInfos.length; i++) {
         fileInfoList.add(fileInfos[i]);
       }
     }
     String[] fileLinks = new String[fileInfoList.size()];
     for (int i = 0; i < fileInfoList.size(); i++) {
       FileInfo fileInfo = (FileInfo) fileInfoList.get(i);
       fileLinks[i] = repository.repositoryFileTranslator(authToken, fileInfo);
       System.out.println("Project extraction file-info link: "+fileLinks[i]);
     }
     // ----------------------------------
     // revert extractions
     repository.extractionResetDatabase();
   } catch (OpenAPIException lEx) {
     System.out.println("ServerCode = " + lEx.getServerErrorCode());
     System.out.println("Message    = " + lEx.getMessage());
     System.out.println("StackTrace:");
     lEx.printStackTrace();
   } catch (RemoteException lEx) {
     lEx.printStackTrace();
   } catch (ServiceException lEx) {
     lEx.printStackTrace();
   } catch (MalformedURLException lEx) {
     lEx.printStackTrace();
   }
  }
}
```

Notes about FileInfo Objects

FileInfo objects represent individual files associated with the extracted asset. The physical location of the file may be obtained by the two following methods.

1. Using the downloadURI property on the FileInfo object itself, i.e. fileInfo.getDownloadURI().
2. Using the OpenAPI method repositoryFileTranslator passing the FileInfo object, i.e. flashlineRegistry.repositoryFileTranslator(authToken, fileInfo).

> **Note:** DO NOT use the URI property on the FileInfo object which represents an Oracle Enterprise Repository specific path.

21.2.2 Use Case: Read an Extraction

Description

Several methods beyond those covered in the Extract an Asset use case is used to read extractions. Extractions can be grouped by asset, project, or user. The specific grouping of extractions determines the method to be used.

To read an extraction the following conditions must be met:

- The project must be open.
- The asset(s) must be registered and active.
- The extraction must be active.

These conditions are checked by the appropriate methods in which they are used. Exceptions are thrown if the conditions are not met.

Sample code is as follows:

```
package com.flashline.sample.extractionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ExtractRead {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
```

```
        // ////////////////////////////
        // Authenticate with OER
        // ////////////////////////////
        AuthToken authToken = repository.authTokenCreate(
            pArgs[1],pArgs[2]);
        long PROJECT_ID = 50000; // must be a valid project id in the OER
        long ASSET_ID = 569;    // must be a valid asset id in the OER
        // ---------------------------------------
        // Read project extractions
        Extraction[] projectExtractions = repository
            .extractionReadProjectExtractions(authToken, PROJECT_ID, true);
        // ---------------------------------------
        // Read asset extractions
        Extraction[] assetExtractions = repository
            .extractionReadAssetExtractions(authToken, PROJECT_ID, ASSET_ID, true);
        // ---------------------------------------
        // Read user extractions
        Extraction[] userExtractions = repository
            .extractionReadUserExtractions(authToken, true);
        // ---------------------------------------
        // Read related assets
        Asset[] assets = repository.extractionReadRelatedAssets(authToken,
            ASSET_ID);
      } catch (OpenAPIException lEx) {
        System.out.println("ServerCode = " + lEx.getServerErrorCode());
        System.out.println("Message    = " + lEx.getMessage());
        System.out.println("StackTrace:");
        lEx.printStackTrace();
      } catch (RemoteException lEx) {
        lEx.printStackTrace();
      } catch (ServiceException lEx) {
        lEx.printStackTrace();
      } catch (MalformedURLException lEx) {
        lEx.printStackTrace();
      }
    }
}
```

21.2.3 Use Case: Update an Extraction

Description

An extraction record is updated when the state of the asset is changed or when the consumer of the asset completes an asset survey. A state change or survey completion can be separate transactions or performed in tandem.

To update an extraction the following conditions must be met:

- The project must be open.
- The asset must be registered and active.
- The extraction must be active.
- The survey taken must be active.

These conditions are checked by the appropriate methods in which they are used. Exceptions are thrown if the conditions are not met.

Sample Code is as follows:

```
package com.flashline.sample.extractionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Answer;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Categorization;
import com.flashline.registry.openapi.entity.Choice;
import com.flashline.registry.openapi.entity.ChoiceList;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.ExtractionDownload;
import com.flashline.registry.openapi.entity.IExtraction;
import com.flashline.registry.openapi.entity.Question;
import com.flashline.registry.openapi.entity.SurveyTaken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ExtractUpdate {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      long PROJECT_ID = 50000; // must be a valid project id in the OER
      long ASSET_ID = 569; // must be a valid asset id in the OER
      // ---------------------------------------
      // Create a new extraction
      long[] lAssetIDs = { ASSET_ID };
      ExtractionDownload[] extractionDownloads =
 repository.extractionCreate(authToken, PROJECT_ID, lAssetIDs);
      Extraction[] assetExtractions = repository
          .extractionReadAssetExtractions(authToken, PROJECT_ID, ASSET_ID, true);
      ////////////////////////////////////////////////////////////////////////
      // this assumes that there is at least 1 extraction and the first one is
      // used
      ////////////////////////////////////////////////////////////////////////
      IExtraction iExtraction = repository
          .extractionReadExtractionStates(authToken);
      Extraction extraction = repository.extractionRead(authToken,
          assetExtractions[0].getID());
      ////////////////////////////////////////////////////////////////////////
      // can set the status of the extraction to 'Deployed', 'Rejected', or 'In
      // Process'.
      ////////////////////////////////////////////////////////////////////////
      assetExtractions[0].setStatus("In Process");
      extraction = repository.extractionTentativelyAccept(authToken,
```

```
            extraction);
          SurveyTaken surveyTaken = repository.surveyTakenRead(authToken,
            extraction);
          extraction = repository.extractionUpdateSurvey(authToken,
            extraction, surveyTaken);
          extraction.setStatus(iExtraction.getInProcess());
          surveyTaken = repository.surveyTakenRead(authToken, extraction);
          extraction = repository.extractionUpdateSurvey(authToken,
            extraction, surveyTaken);
          Categorization[] rejectionReasons = repository
            .extractionReadRejectionReasons(authToken);
          surveyTaken = repository.surveyTakenRead(authToken, extraction);
          extraction = repository.extractionUpdateSurvey(authToken,
            extraction, surveyTaken);
          surveyTaken = repository.surveyTakenRead(authToken, extraction);
          Question[] questions = repository.surveyReadQuestions(authToken);
          ChoiceList choiceList = null;
          Choice[] choices = null;
          Answer[] answers = new Answer[4];
          for (int i = 0; i < answers.length; i++) {
            answers[i] = new Answer();
          }
          answers[0].setQuestionId(questions[0].getId());
          choiceList = questions[0].getChoiceList();
          choices = choiceList.getChoices();
          answers[0].setChoiceId(choices[0].getId());
          answers[0].setValue(choices[0].getValue());
          answers[1].setQuestionId(questions[1].getId());
          answers[1].setChoiceId(0);
          answers[1].setValue("100");
          answers[2].setQuestionId(questions[2].getId());
          answers[2].setChoiceId(0);
          answers[2].setValue("200");
          answers[3].setQuestionId(questions[3].getId());
          choiceList = questions[3].getChoiceList();
          choices = choiceList.getChoices();
          answers[3].setChoiceId(choices[3].getId());
          answers[3].setValue(choices[3].getValue());
          surveyTaken.setAnswers(answers);
          surveyTaken = repository.surveyTakenUpdate(authToken, surveyTaken);
          // ----------------------------------
          // revert extractions
          repository.extractionResetDatabase();
        } catch (OpenAPIException lEx) {
          System.out.println("ServerCode = " + lEx.getServerErrorCode());
          System.out.println("Message    = " + lEx.getMessage());
          System.out.println("StackTrace:");
          lEx.printStackTrace();
        } catch (RemoteException lEx) {
          lEx.printStackTrace();
        } catch (ServiceException lEx) {
          lEx.printStackTrace();
        } catch (MalformedURLException lEx) {
          lEx.printStackTrace();
        }
      }
    }
```

Pitfalls

Bear in mind that a state change or a survey taken is a two-step process. The first step is to change the state or take the survey. The second step is to update the extraction status using the extractionUpdateStatus method. A state change or survey taken can be separate transactions or performed in tandem. If performed in tandem only a single extractionUpdateStatus method call is required.

The extractionUpdateStatus method requires a SurveyTaken. This is true regardless of whether a survey was taken at the time the extractionUpdateStatus method is called (as in a state change, for example). This survey is retrieved using the surveyTakenRead method. If a survey has not been taken for this extraction, then one is created by the surveyTakenRead method.

The current survey in Oracle Enterprise Repository consists of four questions. When a survey is taken an array is created storing answers for the four questions. Each answer must contain three pieces of information to be valid:

- The value (the user response to the question)
- The question ID
- The choice ID

Questions 2 and 3 are single answer questions, so the choice ID here is always set to 0. Questions 1 and 4, however, are multiple-choice. The multiple choices are retrieved using the surveyReadChoiceList method.

Methods to Avoid:

The following objects are used in the Extraction and Survey subsystems:

- Extraction
- ExtractionDownload
- FileInfo
- SurveyTaken
- Question
- ChoiceList
- Choice
- Answer

The use of any of the get methods within these objects is acceptable. All the remaining methods - especially the set methods - should be avoided. The events provided by these remaining methods are covered by the methods in the Extraction and Survey subsystems.

22

Localization of REX Clients

This chapter provides an overview about localization of REX clients and describes the use cases for localization.

This chapter contains the following sections:

- Section 22.1, "Overview"
- Section 22.2, "Use Cases"

22.1 Overview

Statuses are passed back to a REX client as either an OpenAPIException or AuditMsg object. OpenAPIException objects are used for exceptions, whereas, AuditMsg objects are used for processes that run asynchronously. Both of these objects return a text error message to the REX client.

The interface of both objects has been expanded to include an error code and a list of message arguments so that REX clients can display error or status messages in another language. Clients can continue to use the standard error messages or they can ignore the message and use the error code and the message arguments to construct their own error message.

For example, if you want to localize an application that uses REX, you would first get the properties file listing all the possible error messages. The messages look something like this:

```
ERR_9008 = Error updating project with ID = [{0}].
```

Then you must translate all the messages as necessary:

```
ERR_9008 = Errorway updatingay ojectpray ithway IDway = [{0}].
```

If the client code tries to modify a project with ID=123, and that modification fails, then your end-users get an exception with this error message:

```
Error updating project with ID = [123].
```

If you want to display that error in a local language (such as, Pig Latin), you would take the error code, 9008, and look it up in your translated file to get the string "Errorway updatingay ojectpray ithway IDway = [{0}]." Then you would use the message arguments to replace the tokens. In this case, there is only one string, "123", so you should be able to find one message argument.

You can then construct a custom error message for your end-users:

```
Errorway updatingay ojectpray ithway IDway = [123].
```

22.2 Use Cases

This section describes the use cases for localization of REX clients. It contains the following topics:

- Section 22.2.1, "Use Case: Creating localized messages from REX Exceptions"
- Section 22.2.2, "Use Case: Creating localized messages from REX Audit Messages"

22.2.1 Use Case: Creating localized messages from REX Exceptions

Description

- From the OpenAPIException get the server error code and the message arguments
- Get the resource bundle for the OpenAPIExceptions appropriate for the client locale
- Get the string associated with the error code and replace the token with the message arguments

OpenAPIException Sample code is as follows:

```
package com.flashline.sample.localization;
import java.net.URL;
import java.text.MessageFormat;
import java.util.ResourceBundle;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class SyncTest {
    private static final int INVALID_PROJECT_ID = 8672609;
    public static void main(String[] args) throws Exception {
      URL lURL = new
 URL("http://localhost:9080/registry/services/FlashlineRegistry");
      FlashlineRegistry reg = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      AuthToken token = reg.authTokenCreate("admin", "n0pa55w0rd");
      try {
        Project project = reg.projectRead(token,INVALID_PROJECT_ID);
      } catch (OpenAPIException ex) {
        String msg =
 createMessage(ex.getServerErrorCode(),ex.getMessageArguments());
        System.out.println(msg);
      }
    }
    private static String createMessage(int pServerErrorCode, Object[] pArgs) {
      ResourceBundle mResourceBundle =
 ResourceBundle.getBundle("com.flashline.sample.localization.sync_error
_messages");
      return MessageFormat.format(mResourceBundle.getString("ERR
_"+pServerErrorCode), pArgs);
    }
}
```

22.2.2 Use Case: Creating localized messages from REX Audit Messages

Description

- From the AuditMsg and the ImpExpJob get the server error code and the message arguments from the AuditMsg
- Get the resource bundle for audit messsages appropriate for the client locale
- Get the string associated with the error code and replace the token with the message arguments

AuditMsg Sample Code is as follows:

```
package com.flashline.sample.localization;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.text.MessageFormat;
import java.util.ResourceBundle;
import javax.activation.DataHandler;
import javax.xml.rpc.ServiceException;
import org.apache.axis.client.Stub;
import org.apache.soap.util.mime.ByteArrayDataSource;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.ImpExpJob;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AsyncTest {
  public void run() throws MalformedURLException, ServiceException,
 OpenAPIException, RemoteException{
    URL lURL = new
 URL("http://localhost:9080/registry/services/FlashlineRegistry");
    FlashlineRegistry reg = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
    AuthToken token = reg.authTokenCreate("admin", "n0pa55w0rd");
    try {
      File lFile = new
 File("samples/com/flashline/sample/localization/asynctest.zip");
      //Import the file and save to db
      InputStream lIS = new FileInputStream(lFile);
      ByteArrayDataSource lDataSource = new ByteArrayDataSource(lIS,
 "application/x-zip-compressed");
      DataHandler lDH = new DataHandler(lDataSource);
      // add the attachment
      ((Stub)reg).addAttachment(lDH);
      ImpExpJob lJob = reg.importExecute(token, "flashline", null, "Import Assets
 Test", null);
      boolean lPassed = false;
      for(int i=0; i<1000; i++){
        lJob = reg.importStatus(token, lJob);
        System.out.println("Import Job ["+lJob.getID()+"] - State:
 "+lJob.getState());
```

```
        String msg =
createMessage(lJob.getAuditMsg().getSummaryID(),lJob.getAuditMsg().getSummaryArgs
());
        System.out.println(msg);
        if( lJob.getState().equals("completed")){
          lPassed = true;
          break;
        }
        try {
          Thread.sleep(1000);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
      if (lPassed){
         System.out.println("Import Completed");
      }
    } catch (OpenAPIException ex) {
      String msg =
 createMessage(ex.getServerErrorCode(),ex.getMessageArguments());
      System.out.println(msg);
    } catch (FileNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }
  /**
   * @param args
   * @throws ServiceException
   * @throws RemoteException
   * @throws MalformedURLException
   * @throws OpenAPIException
   */
  public static void main(String[] args) throws OpenAPIException,
 MalformedURLException, RemoteException, ServiceException {
    AsyncTest test = new AsyncTest();
    test.run();
  }
  private static String createMessage(int pServerErrorCode, Object[] pArgs) {
    ResourceBundle mResourceBundle =
 ResourceBundle.getBundle("com.flashline.sample.localization.async_error
_messages");
    return MessageFormat.format(mResourceBundle.getString("ERR
_"+pServerErrorCode), pArgs);
  }
  private String readZip(String pFileName) throws IOException {
    int    lNumRead = 0;
    char[] lBuf = new char[2048];
    StringBuffer lQuery = new StringBuffer();
    InputStreamReader lReader = new
 InputStreamReader(getClass().getResourceAsStream(pFileName));
    while( (lNumRead=lReader.read(lBuf)) != -1){
      lQuery.append(lBuf, 0, lNumRead);
    }
    return lQuery.toString();
  }
}
```

23

Notification API

This chapter provides an overview of Notification API and describes the use cases using this API.

This chapter contains the following sections:

- Section 23.1, "Overview"
- Section 23.2, "Use Cases"

23.1 Overview

The Notification Subsystem provides a Web Services-based mechanism that is used to create Oracle Enterprise Repository notifications.

23.2 Use Cases

This section describes the use cases using the Notification API. It contains the following topics:

- Section 23.2.1, "Use Case: Read Notification Substitution List"
- Section 23.2.2, "Use Case: Create a Notification"

23.2.1 Use Case: Read Notification Substitution List

Description

To create a new read notification substitution list.

Sample code is as follows:

23.2.2 Use Case: Create a Notification

Description

Create a Oracle Enterprise Repository notification.

Sample code is as follows:

```
package com.flashline.sample.artifactstoreapi;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
```

```
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.ArtifactStoreBean;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.ArtifactStoreCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ArtifactStores {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
  ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
      // ---------------------------------------
      // query for an artifact store
      ArtifactStoreCriteria lArtifactStoreCriteria = null;
      ArtifactStoreBean[] lArtifactStoreBeans = null;
      ArtifactStoreBean lArtifactStoreBean = null;
      lArtifactStoreCriteria = new ArtifactStoreCriteria();
      lArtifactStoreCriteria.setHostCriteria("existing-artifact-store.com");
      lArtifactStoreCriteria.setBasepathCriteria("/");
      lArtifactStoreBeans = repository.artifactStoreQuery(authToken,
 lArtifactStoreCriteria, false);
      // create a missing artifact store if missing and based on the criteria
      lArtifactStoreCriteria = new ArtifactStoreCriteria();
      lArtifactStoreCriteria.setHostCriteria("missing-artifact-store.com");
      lArtifactStoreCriteria.setBasepathCriteria("/");
      // a new artifact store is created
      lArtifactStoreBeans = repository.artifactStoreQuery(authToken,
 lArtifactStoreCriteria, true);
      lArtifactStoreBean = lArtifactStoreBeans[0];
    } catch(Exception e) {
      throw new RuntimeException(e.getMessage());
    }
  }
}
```

24

Policy API

This chapter provides an overview of Policy API and describes the use cases using this API.

This chapter contains the following sections:

- Section 24.1, "Overview"
- Section 24.2, "Use Cases"

24.1 Overview

REX now supports the following functions against Policies

- Query Policy:
  - Status of the Policy (pass/fail) on an Asset
  - Status of the collection of Policies on an Asset
  - Obtain XML from the Policy Assertion Technical Description Field
  - Assets that the Policy is applied too
- Viewer
  - Maintain list of individual Policy Assertions on a Policy
  - Set status of individual Policy Assertions for an Asset.
  - Apply and remove Policy from assets

Additional Import(s) Required (Some may not be used in all examples.)

```
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.PolicyAssertion;
import com.flashline.registry.openapi.entity.PolicyAssertionResult;
```

Note:

- Policies in Oracle Enterprise Repository are a specific type of asset, based on the Policy Type. Refer to the Asset API use cases for information related to the creation, modification and removal of a Policy.

Definitions

- **Assertions**

An assertion is a policy statement added to a policy asset.

- **AssertionResult**

  When a Policy has been applied to an asset, each assertion within the policy can be evaluated for the asset. The Assertion Result is pass, fail or unknown for any asset and assertion pair.

Methods

There are four new methods available with the FlashlineRegistry service

- `assetReadAppliedPolicies()`
- `assetUpdateAppliedPolicies()`
- `assetEvaluateAgainstPolicy()`
- `assetEvaluateAgainstAllPolicies()`

24.2 Use Cases

This section describes the use cases using the Policy API. It contains the following topics:

- Section 24.2.1, "Use Case: Create a Policy"
- Section 24.2.2, "Use Case: Get All Policies"
- Section 24.2.3, "Use Case: Get/Set Policy Assertions"
- Section 24.2.4, "Use Case: Get Policies That Have Been Applied To An Asset"
- Section 24.2.5, "Use Case: Set Which Policies Are Applied To An Asset"
- Section 24.2.6, "Use Case: Evaluate Asset Compliance"

24.2.1 Use Case: Create a Policy

Description

To create a new policy, create a new asset based on the Policy Type (102).

Sample code is as follows:

```
package com.flashline.sample.policies;
import java.net.URL;
import java.rmi.RemoteException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.PolicyAssertion;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreatePolicySample {
  private static final String POLICY_TYPE_NAME_PREFIX = "Policies-Test Policy
 Type";
  private static final long ASSET_POLICY_ARCHETYPE = 102;
  private static final String POLICY_NAME_PREFIX = "Policies-Test Policy";
  private static final String POLICY_VERSION = "1.0";
  private static FlashlineRegistry mRepository = null;
  private static AssetType mPolicyAssetType = null;
  private AuthToken mAuthToken = null;
```

```
 public CreatePolicySample(String[] pArgs) {
   try {
     ////////////////////////////////////////////////////////
     // Connect to Oracle Enterprise Repository
     ////////////////////////////////////////////////////////
     URL lURL = null;
     lURL = new URL(pArgs[0]);
     mRepository = new
FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
     // ////////////////////////////
     // Authenticate with OER
     // ////////////////////////////
     mAuthToken  = mRepository.authTokenCreate(pArgs[1], pArgs[2]);
     mPolicyAssetType = createPolicyAssetType();
   } catch(Exception e) {
   }
 }
 public static void main(String[] pArgs) {
   try {
     CreatePolicySample lCreatePolicySample = new CreatePolicySample(pArgs);
     // -------------------------
     // create a new policy object
     Asset lPolicy = lCreatePolicySample.createPolicy();
   } catch(Exception e) {
     e.printStackTrace();
   }
 }
 /**
  * Creates an asset policy with a unique name
  */
 private Asset createPolicy() throws RemoteException {
   String lPolicyName = POLICY_NAME_PREFIX + " " + System.currentTimeMillis();
   // -------------------------
   // create a policy (an asset whose assettype's archtype is "102" (policy)
   Asset lPolicy = mRepository.assetCreate(mAuthToken, lPolicyName, POLICY
_VERSION, mPolicyAssetType.getID());
   lPolicy.setCustomData("<custom-data></custom-data>");
   // -------------------------
   // set some polcy assertions
   lPolicy.setPolicyAssertions(generateSampleAssertions());
   return mRepository.assetUpdate(mAuthToken, lPolicy);
 }
 /**
  * Returns several sample policy assertions for use in testing.
  * Located in a function to be shared between test calls.
  *
  * @return Array of policy assertions
  */
 private PolicyAssertion[] generateSampleAssertions() {
   PolicyAssertion[] lPolicyAssertions = new PolicyAssertion[3];
   String[] lPolicyAssertionNames = {"First", "Second", "Third"};
   for (int i=0; i<lPolicyAssertionNames.length; i++) {
     String lPolicyAssertionName = "My " + lPolicyAssertionNames[i] + "
Assertion";
     lPolicyAssertions[i] = new PolicyAssertion();
     lPolicyAssertions[i].setName(lPolicyAssertionName);
     lPolicyAssertions[i].setDescription(lPolicyAssertionName + " Description");
     lPolicyAssertions[i].setTechnicalDefinition(lPolicyAssertionName + "
Technical Definition");
   }
```

```
    return lPolicyAssertions;
  }
  /**
   * Creates an asset policy asset type with a unique name
   */
  private AssetType createPolicyAssetType() throws RemoteException {
    String lPolicyTypeName = POLICY_TYPE_NAME_PREFIX + " " +
 System.currentTimeMillis();
    // ---------------------------
    // create a new asset type
    AssetType lPolicyType = mRepository.assetTypeCreate(mAuthToken,
 lPolicyTypeName);
    // ---------------------------
    // update the asset type to be a policy asset type by settings the archtype =
 102
    lPolicyType.setArcheTypeIDs(new long[] {ASSET_POLICY_ARCHETYPE});
    return mRepository.assetTypeUpdate(mAuthToken, lPolicyType);
  }
}
```

24.2.2 Use Case: Get All Policies

Description

To get all policies, find all assets whose assetype's archetype is a policy archetype (102).

Sample code is as follows:

```
package com.flashline.sample.policies;
import java.net.URL;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.AssetTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class FindPoliciesSample {
  private static FlashlineRegistry mRepository = null;
  private static AuthToken mAuthToken = null;
  public FindPoliciesSample(String[] pArgs) {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      mRepository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      mAuthToken  = mRepository.authTokenCreate(pArgs[1], pArgs[2]);
    } catch(Exception e) {
```

```
    }
  }
  public static void main(String[] pArgs) {
    try {
      FindPoliciesSample lFindPoliciesSample = new FindPoliciesSample(pArgs);
      AssetType[] lPolicyAssetTypes = null;
      Asset[] lPolicies = null;
      AssetTypeCriteria lAssetTypeCriteria = null;
      AssetCriteria lAssetCritera = null;
      List lListPolicies = new LinkedList();
      // ----------------------
      // search for all asset types that have the policy (102) archetype
      lAssetTypeCriteria = new AssetTypeCriteria();
      lAssetTypeCriteria.setArcheTypeCriteria("102");
      lPolicyAssetTypes = mRepository.assetTypeQuery(mAuthToken,
 lAssetTypeCriteria);
      for(int i=0; i<lPolicyAssetTypes.length; i++) {
        // ----------------------
        // for each policy assettype, search for all assets that are of policy
 assettype
        lAssetCritera = new AssetCriteria();
        lAssetCritera.setAssetTypeCriteria(lPolicyAssetTypes[i].getID());
        lPolicies = mRepository.assetQuery(mAuthToken, lAssetCritera);
        // ----------------------
        // add polices to list
        lListPolicies.addAll(Arrays.asList(lPolicies));
      }
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

24.2.3 Use Case: Get/Set Policy Assertions

Description

To get policy assertions, call `getPolicyAssertions`. To set policy assertions, call `setPolicyAssertions`, then update the policy.

Sample Code is as follows:

```
package com.flashline.sample.policies;
import java.net.URL;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.PolicyAssertion;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.AssetTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class GetSetPolicyAssertionsSample {
  private static FlashlineRegistry mRepository = null;
  private static AuthToken mAuthToken = null;
```

```
 public GetSetPolicyAssertionsSample(String[] pArgs) {
   try {
     ////////////////////////////////////////////////////////
     // Connect to Oracle Enterprise Repository
     ////////////////////////////////////////////////////////
     URL lURL = null;
     lURL = new URL(pArgs[0]);
     mRepository = new
FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
     // ////////////////////////////
     // Authenticate with OER
     // ////////////////////////////
     mAuthToken  = mRepository.authTokenCreate(pArgs[1], pArgs[2]);
   } catch(Exception e) {
   }
 }
 public static void main(String[] pArgs) {
   try {
     GetSetPolicyAssertionsSample lGetSetPolicyAssertionsSample = new
GetSetPolicyAssertionsSample(pArgs);
     AssetType[] lPolicyAssetTypes = null;
     Asset[] lPolicies = null;
     AssetTypeCriteria lAssetTypeCriteria = null;
     AssetCriteria lAssetCritera = null;
     List lListPolicies = new LinkedList();
     // ----------------------
     // search for all asset types that have the policy (102) archetype
     lAssetTypeCriteria = new AssetTypeCriteria();
     lAssetTypeCriteria.setArcheTypeCriteria("102");
     lPolicyAssetTypes = mRepository.assetTypeQuery(mAuthToken,
lAssetTypeCriteria);
     for(int i=0; i<lPolicyAssetTypes.length; i++) {
       // ----------------------
       // for each policy assettype, search for all assets that are of policy
assettype
       lAssetCritera = new AssetCriteria();
       lAssetCritera.setAssetTypeCriteria(lPolicyAssetTypes[i].getID());
       lPolicies = mRepository.assetQuery(mAuthToken, lAssetCritera);
       // ----------------------
       // add polices to list
       lListPolicies.addAll(Arrays.asList(lPolicies));
     }
     if(lListPolicies.size() > 0) {
       // ----------------------
       // get the first policy
       Asset lPolicy = (Asset)lListPolicies.get(0);
       // ----------------------
       // get the policy assertions
       PolicyAssertion[] lPolicyAssetions = lPolicy.getPolicyAssertions();
       // ----------------------
       // print out the policy assertions
       for(int i=0; i<lPolicyAssetions.length; i++) {
         lPolicyAssetions[i].toString();
       }
       // ----------------------
       // set different policy assertions
       lPolicy.setPolicyAssertions(generateNewAssertions());
       // ----------------------
       // update the asset with new assertions
       mRepository.assetUpdate(mAuthToken, lPolicy);
```

```
      } else {
        System.out.println("No policies were found in OER.");
      }
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
  /**
   * Returns several sample policy assertions for use in testing.
   * Located in a function to be shared between test calls.
   *
   * @return Array of policy assertions
   */
  private static PolicyAssertion[] generateNewAssertions() {
    PolicyAssertion[] lPolicyAssertions = new PolicyAssertion[3];
    String[] lPolicyAssertionNames = {"NEW-First", "NEW-Second", "NEW-Third"};
    for (int i=0; i<lPolicyAssertionNames.length; i++) {
      String lPolicyAssertionName = "My " + lPolicyAssertionNames[i] + "
 Assertion";
      lPolicyAssertions[i] = new PolicyAssertion();
      lPolicyAssertions[i].setName(lPolicyAssertionName);
      lPolicyAssertions[i].setDescription(lPolicyAssertionName + " Description");
      lPolicyAssertions[i].setTechnicalDefinition(lPolicyAssertionName + "
 Technical Definition");
    }
    return lPolicyAssertions;
  }
}
```

24.2.4 Use Case: Get Policies That Have Been Applied To An Asset

Description

Call `assetReadAppliedPolicies` to obtain policies applied to an asset.

Sample Code is as follows:

```
package com.flashline.sample.policies;
import java.net.URL;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.PolicyAssertion;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.AssetTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class GetAppliedPoliciesSample {
  private static FlashlineRegistry mRepository = null;
  private static AuthToken mAuthToken = null;
  public GetAppliedPoliciesSample(String[] pArgs) {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
```

```
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      mRepository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      mAuthToken  = mRepository.authTokenCreate(pArgs[1], pArgs[2]);
    } catch(Exception e) {
    }
  }
  public static void main(String[] pArgs) {
    try {
      GetAppliedPoliciesSample lGetAppliedPoliciesSample = new
 GetAppliedPoliciesSample(pArgs);
      long lAssetId = 50000;
      // ---------------
      // read the policed appled to asset 50000
      Asset[] lAppliedPolicies = mRepository.assetReadAppliedPolicies(mAuthToken,
 lAssetId);
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

24.2.5 Use Case: Set Which Policies Are Applied To An Asset

Description

Call `assetUpdateAppliedPolicies` to update policies that have been applied to an asset.

Sample Code is as follows:

```
package com.flashline.sample.policies;
import java.net.URL;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AssetType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.PolicyAssertion;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.query.AssetTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ApplyPoliciesSample {
  private static FlashlineRegistry mRepository = null;
  private static AuthToken mAuthToken = null;
  public ApplyPoliciesSample(String pArgs[]) {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
```

```
      mRepository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      mAuthToken  = mRepository.authTokenCreate(pArgs[1], pArgs[2]);
    } catch(Exception e) {
    }
  }
  public static void main(String[] pArgs) {
    try {
      ApplyPoliciesSample lApplyPoliciesSample = new ApplyPoliciesSample(pArgs);
      long lAssetId = 50000;
      long[] lPolicyIds = {50000, 50001, 50002};
      mRepository.assetUpdateAppliedPolicies(mAuthToken, lAssetId, lPolicyIds);
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

24.2.6 Use Case: Evaluate Asset Compliance

Description

Use `assetEvaluateAgainstPolicy` to determine an asset's compliance with a specified policy. Use `assetEvalauteAgainstAllPolicies` to determine an asset's compliance against all applied policies.

Sample Code is as follows:

```
package com.flashline.sample.policies;
import java.net.URL;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class PolicyEvaluationSample {
  private static FlashlineRegistry mRepository = null;
  private static AuthToken mAuthToken = null;
  public PolicyEvaluationSample(String[] pArgs) {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      mRepository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      mAuthToken  = mRepository.authTokenCreate(pArgs[1], pArgs[2]);
    } catch(Exception e) {
    }
  }
  public static void main(String[] pArgs) {
    try {
      PolicyEvaluationSample lPolicyEvalSamp = new PolicyEvaluationSample(pArgs);
```

```
      long lAssetId = 50000;
      long lPolicyId = 50001;
      String lEvaluationResult = null;
      // --------------------
      // evaluate asset id 50000 against policy id 50001
      // the return is one of the following values "pass", "fail", "unknown"
      lEvaluationResult = mRepository.assetEvaluateAgainstPolicy(mAuthToken,
 lAssetId, lPolicyId);
      // --------------------
      // evaluate asset id 50000 against all polices applied to the asset
      // the return is one of the following values "pass", "fail", "unknown"
      lEvaluationResult = mRepository.assetEvaluateAgainstAllPolicies(mAuthToken,
 lAssetId);
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

25

Projects API

This chapter provides an overview of Projects API and describes the use cases using this API.

This chapter contains the following sections:

- Section 25.1, "Overview"
- Section 25.2, "Use Cases"

25.1 Overview

This section covers projects, providing information covering create, read, update, query, and validate. Several entities are attached to Projects: related projects, users, consumed assets, and produced assets. The addition and removal of these entities is also covered in this section.

Additional Import(s) Required (Some may not be used in all examples.)

```
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.query.ProjectCriteria;
import com.flashline.registry.openapi.entity.KeyValuePair;
import com.flashline.registry.openapi.entity.Results;
import java.text.SimpleDateFormat;
import com.flashline.registry.openapi.decision.ExtractionReassignmentDecision;
import com.flashline.registry.openapi.entity.ProjectEntities;
```

25.2 Use Cases

This section describes the use cases using the Projects API. It contains the following topics:

- Section 25.2.1, "Use Case: Create a New Project"
- Section 25.2.2, "Use Case: Read a Project"
- Section 25.2.3, "Use Case: Validate a Project"
- Section 25.2.4, "Use Case: Update a Project"
- Section 25.2.5, "Use Case: Update a Project's Produced Assets"
- Section 25.2.6, "Use Case: Remove Produced Assets from a Project"
- Section 25.2.7, "Use Case: Update a Project's Asset Usage"
- Section 25.2.8, "Use Case: Closing a Project with Hidden Assets"

- Section 25.2.9, "Use Case: Add Users and Related Projects to a Project"
- Section 25.2.10, "Use Case: Remove Related Projects and Users from a Project"
- Section 25.2.11, "Use Case: Update a Project's Extractions - Reassign Extractions to a Different User on the Same or a Different Project"

25.2.1 Use Case: Create a New Project

Description

This method creates a project, assigns users, and assigns related projects.

Rules for projects:

- The project must have an assigned project leader.
- A project's name must be unique and cannot be null.
- A project must be assigned to a department.
- A project's estimated hours must be a whole number, 0 or greater.

Sample code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewProject {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      Project lProject = null;
      ProjectEntities lProjectEntities = null;
      String[] lLeaderIds = null;
      //
------------------------------------------------------------------------------
--
      lProject = new Project();
      lProjectEntities = new ProjectEntities();
      //
```

```
-------------------------------------------------------------------------------
-
      // set the name of project
      lProject.setName("NEW_PROJECT_NAME");
      //
-------------------------------------------------------------------------------
--
      // set the name of the project's department
      lProject.setDepartmentID(50000); // a department with id 50000 must
                                       // already exist
      //
-------------------------------------------------------------------------------
--
      // set the userids of the project leaders
      lLeaderIds = new String[] { "99" };
      lProjectEntities.setLeaderIDs(lLeaderIds);
      //
-------------------------------------------------------------------------------
--
      repository.projectCreate(authToken, lProject, lProjectEntities);
    } catch (OpenAPIException oapie) {
      System.out.println("\t --- ServerCode = " + oapie.getServerErrorCode());
      System.out.println("\t --- Message    = " + oapie.getMessage());
    } catch (Exception e) {
      System.out.println("\t --- ErrorMessage = " + e.getMessage());
    }
  }
}
```

25.2.2 Use Case: Read a Project

Description

Searches for a project and reads its extractions, produced assets, users, and related projects.

Sample code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.query.ProjectCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ReadProject {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
```

```
        // Connect to Oracle Enterprise Repository
        ////////////////////////////////////////////////////////
        URL lURL = null;
        lURL = new URL(pArgs[0]);
        FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
            .getFlashlineRegistry(lURL);
        // ////////////////////////////
        // Authenticate with OER
        // ////////////////////////////
        AuthToken authToken = repository.authTokenCreate(pArgs[1],
            pArgs[2]);
        // ---------------------------------
        // Read a project
        ProjectCriteria projectCriteria = new ProjectCriteria();
        projectCriteria.setNameCriteria("Project A");
        Project[] projects = repository.projectQuery(authToken,
            projectCriteria);
        if (projects.length > 0) {
          try {
            Project projectRead = (Project) projects[0];
            Extraction[] lExtractions = repository.projectReadExtractions(
                authToken, projectRead);
            Asset[] lAssets = repository.projectReadProducedAssets(
                authToken, projectRead);
            Project[] childProjects = repository.projectReadChildProjects(
                authToken, projectRead);
            Project[] parentProjects = repository
                .projectReadParentProjects(authToken, projectRead);
            RegistryUser[] members = repository.projectReadMembers(
                authToken, projectRead);
            RegistryUser[] leaders = repository.projectReadLeaders(
                authToken, projectRead);
          } catch (OpenAPIException ex) {
            ex.printStackTrace();
          }
        } else {
          System.out.println("No projects found");
        }
      } catch (OpenAPIException lEx) {
        System.out.println("ServerCode = " + lEx.getServerErrorCode());
        System.out.println("Message    = " + lEx.getMessage());
        System.out.println("StackTrace:");
        lEx.printStackTrace();
      } catch (RemoteException lEx) {
        lEx.printStackTrace();
      } catch (ServiceException lEx) {
        lEx.printStackTrace();
      } catch (MalformedURLException lEx) {
        lEx.printStackTrace();
      }
    }
  }
```

25.2.3 Use Case: Validate a Project

Description

Validating a project allows the user to catch any validation errors before a project save is attempted.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Department;
import com.flashline.registry.openapi.entity.KeyValuePair;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.Results;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ValidateProject {
  public static void main(String pArgs[]) throws OpenAPIException,
      RemoteException, ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      Project lProject = new Project();
      Results lResults = new Results();
      String[] lLeaders = { "100" };
      Department department = repository.departmentRead(authToken, "Department
 Name");
      ProjectEntities lProjectEntities = new ProjectEntities();
      // ----------------------------------
      // set the project data
      lProjectEntities.setLeaderIDs(lLeaders);
      lProject.setName("Project Name");
      lProject.setDepartmentName("DEPARTMENT_NAME");
      // ----------------------------------
      // Validate a project
      lResults = repository.projectValidate(authToken, lProject,
 lProjectEntities);
      KeyValuePair[] lPairs = lResults.getErrors();
      for (int i = 0; i < lPairs.length; i++) {
        KeyValuePair lPair = lPairs[i];
        System.out.println(lPair.getValue());
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
```

```
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

25.2.4 Use Case: Update a Project

Description

Update the information and data associated with a specific project.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Department;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProject {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      Project lProject = new Project();
      Department department = new Department();
      ProjectEntities lProjectEntities = new ProjectEntities();
      // ----------------------------------
      // creating a new temporary project for sample
      Project lSampleProject = createProject(repository, authToken);
      // ----------------------------------
      // read an existing project
      try {
        lProject = repository.projectRead(authToken, lSampleProject.getID());
      } catch (OpenAPIException ex) {
        throw ex;
```

```
      }
      // ----------------------------------
      // change project data
      lProject.setName("Update "+lProject.getName());
      lProject.setDescription("Updated Description");
      try {
        department = repository.departmentRead(authToken,
            "Different Department");
        if (department==null) {
          System.out.println("dept is null");
          department = repository.departmentCreate(authToken, "Different
Department",
              "Different Department description...");
        }
      } catch (OpenAPIException ex) {
        throw ex;
      }
      lProject.setDepartmentID(department.getID());
      lProject.setAddByDefault(true);
      lProject.setEstimatedHours(50);
      java.util.Calendar lCal = new java.util.GregorianCalendar();
      SimpleDateFormat sdf = new SimpleDateFormat("M/d/yy");
      lCal.setTime(sdf.parse("1/1/04"));
      lProject.setStartDate(lCal);
      // ----------------------------------
      // Update the project
      lProject = (Project) repository.projectUpdate(authToken,
          lProject, lProjectEntities);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    } catch (Exception e) {
    }
  }
  protected static Project createProject(FlashlineRegistry repository, AuthToken
authToken) throws OpenAPIException, RemoteException {
    Project lProject = new Project();
    ProjectEntities lProjectEntities = new ProjectEntities();
    lProject.setName("Project "+Calendar.getInstance().getTimeInMillis());
    lProject.setDepartmentID(50000); // a department with id 50000 must
    String[] lLeaderIds = new String[] { "99" };
    lProjectEntities.setLeaderIDs(lLeaderIds);
    lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
    return lProject;
  }
}
```

25.2.5 Use Case: Update a Project's Produced Assets

Description

Allows the user perform a single database transaction to set the produced assets of a project.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.APIValidationException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProjectProducedAssets {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      ProjectEntities lProjectEntities = new ProjectEntities();
      Asset lSampleAsset1 = createAsset(repository, authToken);
      Asset lSampleAsset2 = createAsset(repository, authToken);
      String[] assetIds = { ""+lSampleAsset1.getID(), ""+lSampleAsset2.getID() };
      try {
        // ----------------------------------
        // read an existing project
        Project projectRead = repository.projectRead(authToken, 50000);
        // ----------------------------------
        // set the produced asset ids
        lProjectEntities.setAssetIDs(assetIds);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead,
            lProjectEntities);
      } catch (APIValidationException ex) {
        ex.printStackTrace();
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
```

```
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static Asset createAsset(FlashlineRegistry repository, AuthToken
 authToken)
      throws OpenAPIException, RemoteException {
    Asset myAsset = repository.assetCreate(authToken,
        "My Produced Asset", ""+Calendar.getInstance().getTimeInMillis(), 144);
    return myAsset;
  }
}
```

25.2.6 Use Case: Remove Produced Assets from a Project

Description

Remove produced assets from a project.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.APIValidationException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class RemoveProducedAssetsFromProject {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      ProjectEntities lProjectEntities = new ProjectEntities();
```

```
      String[] assetIds = { "569", "589" };
      try {
        // ----------------------------------
        // read an existing project
        Project projectRead = repository.projectRead(authToken, 50000);
        // ----------------------------------
        // set the remove assets ids
        lProjectEntities.setRemovedAssetIDs(assetIds);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead,
            lProjectEntities);
      } catch (APIValidationException ex) {
        ex.printStackTrace();
      }
      // ----------------------------------
      // revert extractions
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

As an alternative, produced assets may be removed by specifying the assets that are to remain on the project.

Sample Code:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class RemoveProducedAssetsFromProject2 {
  public static void main(String pArgs[])
      throws OpenAPIException, RemoteException, ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
```

```
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ----------------------------------
      // An alternate way of removing produced assets is to specify which assets
      // you wish to remain on the project.
      String[] assetIDs = { "569" };
      ProjectEntities lEntities = new ProjectEntities();
      Project projectRead = new Project();
      try {
        // ----------------------------------
        // read an existing project
        projectRead = repository.projectRead(authToken, 50000);
        // ----------------------------------
        // set the entities of the produced assets
        lEntities.setAssetIDs(assetIDs);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead, lEntities);
      } catch (OpenAPIException ex) {
        ex.printStackTrace();
      }
      // ----------------------------------
      // revert extractions
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

25.2.7 Use Case: Update a Project's Asset Usage

Description

Allows the user to reject extractions that are associated with a project.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.List;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectAsset;
```

```
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProjectExtractions {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ----------------------------------
      // Update a project's extractions
      ProjectEntities lProjectEntities = null;
      try {
        // ----------------------------------
        // read an existing project
        Project projectRead = repository.projectRead(authToken, 50000);
        // ----------------------------------
        // get an extraction or create one
        long lExtractionID = 0;
        ProjectAsset[] lProjectAssets = projectRead.getAssets();
        if (lProjectAssets!=null && lProjectAssets.length>0) {
          lProjectAssets[0].getStatus();
          lExtractionID = lProjectAssets[0].getID();
        } else {
          lProjectEntities = new ProjectEntities();
          lExtractionID = repository.assetRead(authToken, 569).getID();
          String[] lAssetIDs = { ""+lExtractionID };
          lProjectEntities.setAssetIDs(lAssetIDs);
          repository.projectUpdate(authToken, projectRead, lProjectEntities);
        }
        // ----------------------------------
        // set the rejected assets ids
        String[] rejectedIds = null;
        projectRead = repository.projectRead(authToken, 50000); // reload modified
 project
        Extraction[] lExtractions = repository.projectReadExtractions(authToken,
 projectRead);
        rejectedIds = new String[lExtractions.length];
        for (int i=0; lExtractions!=null && i<lExtractions.length; i++) {
          rejectedIds[i] = ""+lExtractions[i].getID();
        }
        lProjectEntities = new ProjectEntities();
        lProjectEntities.setRejectedIDs(rejectedIds);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead, lProjectEntities);
      } catch (OpenAPIException ex) {
        ex.printStackTrace();
```

```
      }
      // ----------------------------------
      // revert extractions
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

25.2.8 Use Case: Closing a Project with Hidden Assets

Description

When closing a project, the project lead is required to update the usage status of assets consumed in the project that have not already been designated as `DEPLOYED` or `REJECTED`.

However, certain Advanced Role Based Access Control (RBAC) settings in AquaLogic Enterprise Repository may prevent the project lead from seeing all assets consumed by the project.

If the project is closed, any hidden assets not already rejected are automatically designated as `DEPLOYED`.

When using AquaLogic Enterprise Repository, the project lead in this situation is notified that the project contains hidden assets, and is provided with the opportunity to contact users who have the necessary access to update the usage status of the hidden assets and to complete an asset value survey. Once the project lead is confident that the appropriate users have taken the necessary action, he/she can close the project.

The following example demonstrates a programmatic FLEX mechanism for handling the status update of assets that are hidden from the project lead at project closure.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.List;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectAsset;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
```

```
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProjectExtractions {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ----------------------------------
      // Update a project's extractions
      ProjectEntities lProjectEntities = null;
      try {
        // ----------------------------------
        // read an existing project
        Project projectRead = repository.projectRead(authToken, 50000);
        // ----------------------------------
        // get an extraction or create one
        long lExtractionID = 0;
        ProjectAsset[] lProjectAssets = projectRead.getAssets();
        if (lProjectAssets!=null && lProjectAssets.length>0) {
          lProjectAssets[0].getStatus();
          lExtractionID = lProjectAssets[0].getID();
        } else {
          lProjectEntities = new ProjectEntities();
          lExtractionID = repository.assetRead(authToken, 569).getID();
          String[] lAssetIDs = { ""+lExtractionID };
          lProjectEntities.setAssetIDs(lAssetIDs);
          repository.projectUpdate(authToken, projectRead, lProjectEntities);
        }
        // ----------------------------------
        // set the rejected assets ids
        String[] rejectedIds = null;
        projectRead = repository.projectRead(authToken, 50000); // reload modified
 project
        Extraction[] lExtractions = repository.projectReadExtractions(authToken,
 projectRead);
        rejectedIds = new String[lExtractions.length];
        for (int i=0; lExtractions!=null && i<lExtractions.length; i++) {
          rejectedIds[i] = ""+lExtractions[i].getID();
        }
        lProjectEntities = new ProjectEntities();
        lProjectEntities.setRejectedIDs(rejectedIds);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead, lProjectEntities);
      } catch (OpenAPIException ex) {
        ex.printStackTrace();
      }
      // ----------------------------------
      // revert extractions
```

```
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

25.2.9 Use Case: Add Users and Related Projects to a Project

Description

The process of adding users to project is similar to the process of adding related projects.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.APIValidationException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AddUsersAndRelatedProjectsToProject {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ---------------------------------
      // Add users and related projects to a project
```

```
     Project projectRead = new Project();
     String[] newLeaderIDs = { "99" };
     ProjectEntities lEntities = new ProjectEntities();
     try {
       // ----------------------------------
       // read an existing project
       projectRead = repository.projectRead(authToken, 50000);
       // ----------------------------------
       // create two new projects
       Project lParentProject = createNewProject(repository, authToken, "My
Parent Project");
       Project lChildProject = createNewProject(repository, authToken, "My Child
Project");
       String[] newParentIDs = { ""+lParentProject.getID() };
       String[] newChildIDs = { ""+lChildProject.getID() };
       // ----------------------------------
       // create two new users
       RegistryUser lUserOne = createNewUser(repository, authToken, "one");
       RegistryUser lUserTwo = createNewUser(repository, authToken, "two");
       String[] newMemberIDs = { ""+lUserOne.getID(), ""+lUserTwo.getID() };
       // ----------------------------------
       // set the added leader ids
       lEntities.setAddedLeaderIDs(newLeaderIDs);
       // ----------------------------------
       // set the added member ids
       lEntities.setAddedMemberIDs(newMemberIDs);
       // ----------------------------------
       // set the added children project ids
       lEntities.setAddedChildIDs(newChildIDs);
       // ----------------------------------
       // set the added parent project ids
       lEntities.setAddedParentIDs(newParentIDs);
       // ----------------------------------
       // update the project
       repository.projectUpdate(authToken, projectRead, lEntities);
     } catch (OpenAPIException ex) {
       throw ex;
     }
   } catch (OpenAPIException lEx) {
     System.out.println("ServerCode = " + lEx.getServerErrorCode());
     System.out.println("Message    = " + lEx.getMessage());
     System.out.println("StackTrace:");
     lEx.printStackTrace();
   } catch (RemoteException lEx) {
     lEx.printStackTrace();
   } catch (ServiceException lEx) {
     lEx.printStackTrace();
   } catch (MalformedURLException lEx) {
     lEx.printStackTrace();
   }
 }
 protected static Project createNewProject(FlashlineRegistry repository,
AuthToken authToken, String pName)
     throws APIValidationException, RemoteException {
   Project lProject = new Project();
   ProjectEntities lProjectEntities = new ProjectEntities();
   lProject.setName(pName+" "+Calendar.getInstance().getTimeInMillis()); // force
uniqueness
   lProject.setDepartmentID(50000); // a department with id 50000 must already
exist
```

```
    String[] lLeaderIds = new String[] { "99" };
    lProjectEntities.setLeaderIDs(lLeaderIds);
    lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
    return lProject;
  }
  protected static RegistryUser createNewUser(FlashlineRegistry repository,
 AuthToken authToken, String pUserName)
      throws APIValidationException, RemoteException {
    String lUserName = pUserName + Calendar.getInstance().getTimeInMillis(); //
 force uniqueness
    RegistryUser lRegistryUser = repository.userCreate(authToken, lUserName,
 "First", pUserName,
        pUserName+"@example.com", pUserName, false, false, false);
    return lRegistryUser;
  }
}
```

The following example presents an alternate way of adding users and related projects. In this example, the added users/projects are the ONLY users/projects assigned to the project. Any users/projects not included in the String Array of IDs are removed from the project. This option combines adding and removing users into one step.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.APIValidationException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class AddUsersAndRelatedProjectsToProject2 {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ---------------------------------
      // The following example presents an alternate way of adding users and
      // related projects.
      // In this example the added users/projects are the ONLY
```

```
      // users/projects assigned to the project.
      // Any users/projects not included in the String Array of IDs are
      // removed from the project.
      // This option combines adding and removing users into one step.
      Project projectRead = new Project();
      String[] newLeaderIDs = { "50003" };
      ProjectEntities lEntities = new ProjectEntities();
      try {
        // ----------------------------------
        // read an existing project
        projectRead = repository.projectRead(authToken, 50000);
        // ----------------------------------
        // create two new projects
        Project lParentProject = createNewProject(repository, authToken, "My
Parent Project");
        Project lChildProject = createNewProject(repository, authToken, "My Child
Project");
        String[] newParentIDs = { ""+lParentProject.getID() };
        String[] newChildIDs = { ""+lChildProject.getID() };
        // ----------------------------------
        // create two new users
        RegistryUser lUserOne = createNewUser(repository, authToken, "one");
        RegistryUser lUserTwo = createNewUser(repository, authToken, "two");
        String[] newMemberIDs = { ""+lUserOne.getID(), ""+lUserTwo.getID() };
        // ----------------------------------
        // set the leader ids
        lEntities.setLeaderIDs(newLeaderIDs);
        // ----------------------------------
        // set the member ids
        lEntities.setMemberIDs(newMemberIDs);
        // ----------------------------------
        // set the children project ids
        lEntities.setChildIDs(newChildIDs);
        // ----------------------------------
        // set the parent project ids
        lEntities.setParentIDs(newParentIDs);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead, lEntities);
      } catch (OpenAPIException ex) {
        throw ex;
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static Project createNewProject(FlashlineRegistry repository,
AuthToken authToken, String pName)
      throws APIValidationException, RemoteException {
    Project lProject = new Project();
    ProjectEntities lProjectEntities = new ProjectEntities();
```

```
   lProject.setName(pName+" "+Calendar.getInstance().getTimeInMillis()); // force
 uniqueness
   lProject.setDepartmentID(50000); // a department with id 50000 must already
 exist
   String[] lLeaderIds = new String[] { "99" };
   lProjectEntities.setLeaderIDs(lLeaderIds);
   lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
   return lProject;
  }
  protected static RegistryUser createNewUser(FlashlineRegistry repository,
 AuthToken authToken, String pUserName)
      throws APIValidationException, RemoteException {
   String lUserName = pUserName + Calendar.getInstance().getTimeInMillis(); //
 force uniqueness
   RegistryUser lRegistryUser = repository.userCreate(authToken, lUserName,
 "First", pUserName,
       pUserName+"@example.com", pUserName, false, false, false);
   return lRegistryUser;
  }
}
```

25.2.10 Use Case: Remove Related Projects and Users from a Project

Description:

The process of removing users from a project is similar to the process of removing related projects.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.decision.ExtractionReassignmentDecision;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class RemoveRelatedProjectsAndUsersFromProject {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
```

```
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      long lParentProjectID = createProject(repository, authToken).getID();
      long lChildProject1ID = createProject(repository, authToken).getID();
      long lChildProject2ID = createProject(repository, authToken).getID();
      long lUser1ID = createUser(repository, authToken).getID();
      long lUser2ID = createUser(repository, authToken).getID();
      long lUser3ID = createUser(repository, authToken).getID();
      long lUser4ID = createUser(repository, authToken).getID();
      long lUser5ID = createUser(repository, authToken).getID();
      long lUser6ID = createUser(repository, authToken).getID();
      // ----------------------------------
      // Remove related projects and users from a project
      Project projectRead = new Project();
      String[] removedParentProjectIDs = { ""+lParentProjectID };
      String[] removedChildProjectIDs = { ""+lChildProject1ID, ""+lChildProject2ID
};
      String[] removedLeaderIDs = { ""+lUser1ID };
      String[] removedMemberIDs = { ""+lUser2ID };
      ProjectEntities lEntities = new ProjectEntities();
      try {
        projectRead = repository.projectRead(authToken, 50000);
      } catch (OpenAPIException ex) {
        throw ex;
      }
      try {
        // ----------------------------------
        // set the removed parent project ids
        lEntities.setRemovedParentIDs(removedParentProjectIDs);
        // ----------------------------------
        // set the removed children project ids
        lEntities.setRemovedChildIDs(removedChildProjectIDs);
        // ----------------------------------
        // set the remove leader ids
        lEntities.setRemovedLeaderIDs(removedLeaderIDs);
        // ----------------------------------
        // set the removed member ids
        lEntities.setRemovedMemberIDs(removedMemberIDs);
        // ----------------------------------
        // set the exraction reassignment decisions
        ExtractionReassignmentDecision[] decisions = new
ExtractionReassignmentDecision[2];
        ExtractionReassignmentDecision decision = new
ExtractionReassignmentDecision();
        decision.setUserID(lUser3ID);
        decision.setReassignUserID(lUser4ID);
        decisions[0] = decision;
        decision = new ExtractionReassignmentDecision();
        decision.setUserID(lUser5ID);
        decision.setReassignUserID(lUser6ID);
        decisions[1] = decision;
        // ----------------------------------
        // set the userid for the reassigned extracions
        lEntities.setReassignIDs(decisions);
        // ----------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead, lEntities);
      } catch (OpenAPIException ex) {
        throw ex;
```

```
      }
      // ----------------------------------
      // revert extractions
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static Project createProject(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    Project lProject = new Project();
    ProjectEntities lProjectEntities = new ProjectEntities();
    lProject.setName("Project "+Calendar.getInstance().getTimeInMillis());
    lProject.setDepartmentID(50000); // a department with id 50000 must
    String[] lLeaderIds = new String[] { "99" };
    lProjectEntities.setLeaderIDs(lLeaderIds);
    lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
    return lProject;
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

As an alternative, the following example tells the system which users/projects to keep, rather than telling it which ones to remove.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.decision.ExtractionReassignmentDecision;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class RemoveRelatedProjectsAndUsersFromProject2 {
  public static void main(String pArgs[])
      throws OpenAPIException, RemoteException, ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
```

```
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ---------------------------------
      // As an alternative, the following example tells the system which
      // users/projects to keep,
      // rather than telling it which ones to remove.
      Project projectRead = new Project();
      String[] lParentProjectIDs = { "50003" };
      String[] lChildProjectIDs = { "50002", "50001" };
      String[] lLeaderIDs = { "50001" };
      String[] lMemberIDs = { "50005" };
      ProjectEntities lEntities = new ProjectEntities();
      try {
        projectRead = repository.projectRead(authToken, 50000);
      } catch (OpenAPIException ex) {
        throw ex;
      }
      try {
        // ---------------------------------
        // set the parent project ids
        lEntities.setParentIDs(lParentProjectIDs);
        // ---------------------------------
        // set the children project ids
        lEntities.setChildIDs(lChildProjectIDs);
        // ---------------------------------
        // set the leader ids
        lEntities.setLeaderIDs(lLeaderIDs);
        // ---------------------------------
        // set the member ids
        lEntities.setMemberIDs(lMemberIDs);
        // ---------------------------------
        // set the extraction reassignment decisions
        ExtractionReassignmentDecision[] decisions = new
ExtractionReassignmentDecision[2];
        ExtractionReassignmentDecision decision = new
ExtractionReassignmentDecision();
        decision.setUserID(50011);
        decision.setReassignUserID(50001);
        decisions[0] = decision;
        decision = new ExtractionReassignmentDecision();
        decision.setUserID(50012);
        decision.setReassignUserID(50005);
        decisions[1] = decision;
        // ---------------------------------
        // set the userid for the reassigned extracions
        lEntities.setReassignIDs(decisions);
        // ---------------------------------
        // update the project
        repository.projectUpdate(authToken, projectRead, lEntities);
      } catch (OpenAPIException ex) {
        throw ex;
      }
      // ---------------------------------
```

```
      // revert extractions
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

25.2.11 Use Case: Update a Project's Extractions - Reassign Extractions to a Different User on the Same or a Different Project

Description

Extractions can be reassigned from one user to another. The user receiving the reassigned extractions can be on the same or a different project.

Sample Code as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.decision.ExtractionReassignmentDecision;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProjectExtractionswithReassign {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
```

```
      Project lProject = null;
      long currentProjectID = 50000; // an existing project with id 50000 must
                                     // exist
      long someProjectID = 0; // an existing project with id 50001 must
                              // exist where re-extractions are being
                              // assigned to
      long extractionID = 50002; // the id of the extraction being reassigned
      long reassignUserID = 0; // the id of the user being assigned to this
                               // extraction
      // ----------------------------------
      // Update a project's extractions - reassign extractions to a different
      // user on the same or different project
      // ----------------------------------
      // read a project, get a sample project
      lProject = repository.projectRead(authToken, currentProjectID);
      someProjectID = createProject(repository, authToken).getID();
      // ----------------------------------
      // get a member of the project to reassign
      Project lReassignProject = repository.projectRead(authToken, someProjectID);
      String[] memberIDs = repository.projectReadMemberIDs(authToken,
lReassignProject);
      if (memberIDs!=null && memberIDs.length>0) {
        reassignUserID = Long.parseLong(memberIDs[0]);
      }
      // ----------------------------------
      // if no members exist, create a user and add them
      if (reassignUserID==0) {
        ProjectEntities lProjectEntities = new ProjectEntities();
        reassignUserID = createUser(repository, authToken).getID();
        String[] newMemberIDs = { ""+reassignUserID };
        lProjectEntities.setAddedMemberIDs(newMemberIDs);
        repository.projectUpdate(authToken, lReassignProject, lProjectEntities);
      }
      // ----------------------------------
      // set the extraction reassignment decision
      ExtractionReassignmentDecision[] lDecisions = new
ExtractionReassignmentDecision[1];
      ExtractionReassignmentDecision lDecision = new
ExtractionReassignmentDecision();
      // ----------------------------------
      // set the reassigned project id
      lDecision.setProjectID(someProjectID);
      // ----------------------------------
      // specify which extraction (by id)
      lDecision.setExtractionID(extractionID);
      // ----------------------------------
      // set the reassigned user id
      lDecision.setReassignUserID(reassignUserID);
      lDecisions[0] = lDecision;
      // ----------------------------------
      // reassign project extractions
      repository.projectReassignExtractions(authToken, lProject,
          lDecisions);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
```

```
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static Project createProject(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    Project lProject = new Project();
    ProjectEntities lProjectEntities = new ProjectEntities();
    lProject.setName("Project "+Calendar.getInstance().getTimeInMillis());
    lProject.setDepartmentID(50000); // a department with id 50000 must
    String[] lLeaderIds = new String[] { "99" };
    lProjectEntities.setLeaderIDs(lLeaderIds);
    lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
    return lProject;
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

25.2.12 Use Case: Update a Project's User - Reassign User and His/Her Extractions to Another Project

Description

Reassign a user and his/her extractions to another project.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.decision.ExtractionReassignmentDecision;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProjectExtractionswithReassign {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
```

```
        FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
            .getFlashlineRegistry(lURL);
        // ////////////////////////////
        // Authenticate with OER
        // ////////////////////////////
        AuthToken authToken = repository.authTokenCreate(pArgs[1],
            pArgs[2]);
        Project lProject = null;
        long currentProjectID = 50000; // an existing project with id 50000 must
                                       // exist
        long someProjectID = 0; // an existing project with id 50001 must
                                // exist where re-extractions are being
                                // assigned to
        long extractionID = 50002; // the id of the extraction being reassigned
        long reassignUserID = 0; // the id of the user being assigned to this
                                 // extraction
        // ----------------------------------
        // Update a project's extractions - reassign extractions to a different
        // user on the same or different project
        // ----------------------------------
        // read a project, get a sample project
        lProject = repository.projectRead(authToken, currentProjectID);
        someProjectID = createProject(repository, authToken).getID();
        // ----------------------------------
        // get a member of the project to reassign
        Project lReassignProject = repository.projectRead(authToken, someProjectID);
        String[] memberIDs = repository.projectReadMemberIDs(authToken,
lReassignProject);
        if (memberIDs!=null && memberIDs.length>0) {
          reassignUserID = Long.parseLong(memberIDs[0]);
        }
        // ----------------------------------
        // if no members exist, create a user and add them
        if (reassignUserID==0) {
          ProjectEntities lProjectEntities = new ProjectEntities();
          reassignUserID = createUser(repository, authToken).getID();
          String[] newMemberIDs = { ""+reassignUserID };
          lProjectEntities.setAddedMemberIDs(newMemberIDs);
          repository.projectUpdate(authToken, lReassignProject, lProjectEntities);
        }
        // ----------------------------------
        // set the extraction reassignment decision
        ExtractionReassignmentDecision[] lDecisions = new
ExtractionReassignmentDecision[1];
        ExtractionReassignmentDecision lDecision = new
ExtractionReassignmentDecision();
        // ----------------------------------
        // set the reassigned project id
        lDecision.setProjectID(someProjectID);
        // ----------------------------------
        // specify which extraction (by id)
        lDecision.setExtractionID(extractionID);
        // ----------------------------------
        // set the reassigned user id
        lDecision.setReassignUserID(reassignUserID);
        lDecisions[0] = lDecision;
        // ----------------------------------
        // reassign project extractions
        repository.projectReassignExtractions(authToken, lProject,
            lDecisions);
```

```
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static Project createProject(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    Project lProject = new Project();
    ProjectEntities lProjectEntities = new ProjectEntities();
    lProject.setName("Project "+Calendar.getInstance().getTimeInMillis());
    lProject.setDepartmentID(50000); // a department with id 50000 must
    String[] lLeaderIds = new String[] { "99" };
    lProjectEntities.setLeaderIDs(lLeaderIds);
    lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
    return lProject;
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

25.2.13 Use Case: Update a Project's User - Reassign User Only (Not the User's Extractions) to Another Project

Description

Users can be reassigned from one project to another. If the user is to be moved without his/her extractions, the extractions must first be reassigned to another project member before the user is reassigned.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.decision.ExtractionReassignmentDecision;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.ProjectUserType;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
```

```
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateProjectUserWithReassign2 {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
      Project lProject = null;
      Project someProject = null; // the id the other project
      Project reassignProject = null; // the id of the project being reassigned
      long currentProjectID = 50000; // the id of the current project
      long extractionID = 0; // the id of the extraction
      long extractionReassignUserID = 0; // the id of the user being
                                              // reassigned
      long projectReassignUserID = 0; // the id of the user being reassigned
      // ----------------------------------
      // Update a project's user - reassign only the user (not their
      // extractions) to another project
      // ----------------------------------
      // read a project
      lProject = repository.projectRead(authToken, currentProjectID);
      // ----------------------------------
      // create some projects
      someProject = createProject(repository, authToken);
      reassignProject = createProject(repository, authToken);
      // ----------------------------------
      // get a member of the project to reassign
      String[] memberIDs = repository.projectReadMemberIDs(authToken, lProject);
      if (memberIDs!=null && memberIDs.length>0) {
        extractionReassignUserID = Long.parseLong(memberIDs[0]);
        if (memberIDs.length>1) {
          projectReassignUserID = Long.parseLong(memberIDs[0]);
        }
      }
      // ----------------------------------
      // if no members exist, create users and add them
      if (extractionReassignUserID==0) {
        ProjectEntities lProjectEntities = new ProjectEntities();
        extractionReassignUserID = createUser(repository, authToken).getID();
        lProjectEntities.setAddedMemberIDs(new String[]{
 ""+extractionReassignUserID });
        repository.projectUpdate(authToken, lProject, lProjectEntities);
      }
      if (projectReassignUserID==0) {
        ProjectEntities lProjectEntities = new ProjectEntities();
        projectReassignUserID = createUser(repository, authToken).getID();
        lProjectEntities.setAddedMemberIDs(new String[]{ ""+projectReassignUserID
 });
        repository.projectUpdate(authToken, lProject, lProjectEntities);
```

```
     }
     // get extraction for user or create one
     Extraction[] lAssetExtractions =
repository.projectReadExtractions(authToken, lProject);
     if (lAssetExtractions!=null && lAssetExtractions.length>0) {
       extractionID = lAssetExtractions[0].getID();
     }
     if (extractionID==0) {
       // create new extraction
       ProjectEntities lProjectEntities = new ProjectEntities();
       Asset lAsset = repository.assetRead(authToken, 569);
       lProjectEntities.setAssetIDs(new String[]{ ""+lAsset.getID() });
       repository.projectUpdate(authToken, lProject, lProjectEntities);
     }
     // ----------------------------------
     // add users to reassign project (if they aren't already)
     {
       Project lReassignProject = repository.projectRead(authToken,
reassignProject.getID());
       boolean isMemberExtraction = false, isMemberProject = false;
       String[] reassignMemberIDs = repository.projectReadMemberIDs(authToken,
lReassignProject);
       for (int i=0; reassignMemberIDs!=null && i<reassignMemberIDs.length; i++)
{
         if
(Long.parseLong(reassignMemberIDs[i].trim())==extractionReassignUserID)
isMemberExtraction = true;
         if (Long.parseLong(reassignMemberIDs[i].trim())==projectReassignUserID)
isMemberProject = true;
       }
       if (!isMemberExtraction) {
         ProjectEntities lProjectEntities = new ProjectEntities();
         lProjectEntities.setAddedMemberIDs(new String[]{
""+extractionReassignUserID });
         repository.projectUpdate(authToken, lReassignProject, lProjectEntities);
       }
       if (!isMemberProject) {
         ProjectEntities lProjectEntities = new ProjectEntities();
         lProjectEntities.setAddedMemberIDs(new String[]{
""+projectReassignUserID });
         repository.projectUpdate(authToken, lReassignProject, lProjectEntities);
       }
     }
     // ----------------------------------
     // add users to some project (if they aren't already)
     {
       Project lSomeProject = repository.projectRead(authToken,
someProject.getID());
       boolean isMemberExtraction = false, isMemberProject = false;
       String[] SomeMemberIDs = repository.projectReadMemberIDs(authToken,
lSomeProject);
       for (int i=0; SomeMemberIDs!=null && i<SomeMemberIDs.length; i++) {
         if (Long.parseLong(SomeMemberIDs[i].trim())==extractionReassignUserID)
isMemberExtraction = true;
         if (Long.parseLong(SomeMemberIDs[i].trim())==projectReassignUserID)
isMemberProject = true;
       }
       if (!isMemberExtraction) {
         ProjectEntities lProjectEntities = new ProjectEntities();
         lProjectEntities.setAddedMemberIDs(new String[]{
```

```
""+extractionReassignUserID });
        repository.projectUpdate(authToken, lSomeProject, lProjectEntities);
       }
       if (!isMemberProject) {
        ProjectEntities lProjectEntities = new ProjectEntities();
        lProjectEntities.setAddedMemberIDs(new String[]{
""+projectReassignUserID });
        repository.projectUpdate(authToken, lSomeProject, lProjectEntities);
       }
     }
     // ---------------------------------
     // set extraction reassignment decision
     ExtractionReassignmentDecision[] lDecisions = new
ExtractionReassignmentDecision[1];
     ExtractionReassignmentDecision lDecision = new
ExtractionReassignmentDecision();
     // set the reassign decision's project id
     lDecision.setProjectID(someProject.getID());
     // set the reassign decision's extraction id
     lDecision.setExtractionID(extractionID);
     // set the reassign decision's reassigned user ids
     lDecision.setReassignUserID(extractionReassignUserID);
     lDecisions[0] = lDecision;
     // reassign project extractions
     repository.projectReassignExtractions(authToken, lProject,
         lDecisions);
     // verify reassignment
     lProject = repository.projectRead(authToken, currentProjectID);
     ProjectUserType userType = repository
         .projectReadUserTypes(authToken);
     lDecisions = new ExtractionReassignmentDecision[1];
     lDecision = new ExtractionReassignmentDecision();
     lDecision.setProjectID(reassignProject.getID());
     lDecision.setReassignUserID(projectReassignUserID);
     lDecision.setReassignType(userType.getUserTypeLeader());
     lDecisions[0] = lDecision;
     repository.projectReassignUsers(authToken, lProject, lDecisions);
     // ---------------------------------
     // revert extractions
     repository.extractionResetDatabase();
   } catch (OpenAPIException lEx) {
     System.out.println("ServerCode = " + lEx.getServerErrorCode());
     System.out.println("Message    = " + lEx.getMessage());
     System.out.println("StackTrace:");
     lEx.printStackTrace();
   } catch (RemoteException lEx) {
     lEx.printStackTrace();
   } catch (ServiceException lEx) {
     lEx.printStackTrace();
   } catch (MalformedURLException lEx) {
     lEx.printStackTrace();
   }
 }
 protected static Project createProject(FlashlineRegistry repository, AuthToken
authToken) throws OpenAPIException, RemoteException {
   Project lProject = new Project();
   ProjectEntities lProjectEntities = new ProjectEntities();
   lProject.setName("Project "+Calendar.getInstance().getTimeInMillis());
   lProject.setDepartmentID(50000); // a department with id 50000 must
   String[] lLeaderIds = new String[] { "99" };
```

```
    lProjectEntities.setLeaderIDs(lLeaderIds);
    lProject = repository.projectCreate(authToken, lProject, lProjectEntities);
    return lProject;
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

25.2.14 Use Case: Read the Value-Provided for a Project and Asset

Description

Reads the value-provided detail for a project and asset.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Answer;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Choice;
import com.flashline.registry.openapi.entity.ChoiceList;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.ExtractionDownload;
import com.flashline.registry.openapi.entity.ProjectAssetValue;
import com.flashline.registry.openapi.entity.Question;
import com.flashline.registry.openapi.entity.SurveyTaken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ReadValueProvidedForProjectAndAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ---------------------------------
      // Read the value provided for a project and asset
      long projectid = 50000; // the id of the project
      long assetid = 569; // the id of the asset
```

```
     // ----------------------------------
     // Make sure the project has an extraction
     long[] lAssetIDs = { assetid };
     ExtractionDownload[] extractionDownload =
repository.extractionCreate(authToken, projectid, lAssetIDs);
     Extraction extraction =
repository.extractionReadByProjectAndAsset(authToken, projectid, assetid);
     // ----------------------------------
     // take survey and update
     SurveyTaken surveyTaken = takeSurvey(repository, authToken, extraction);
     surveyTaken = repository.surveyTakenUpdate(authToken, surveyTaken);
     extraction = repository.extractionUpdateSurvey(authToken, extraction,
surveyTaken);
     // ----------------------------------
     // read project asset values
     ProjectAssetValue[] projectAssetValues = repository
         .projectAssetValueRead(authToken, projectid, assetid);
     if (projectAssetValues != null) {
       for (int i = 0; i < projectAssetValues.length; i++) {
         ProjectAssetValue projectAssetValue = projectAssetValues[i];
         projectAssetValue.getUserInfo().getUserName();
         projectAssetValue.getExtractionDate();
         projectAssetValue.getExtractionStatus();
         projectAssetValue.getPredictedValue();
         projectAssetValue.isPredictedValueSelected();
         projectAssetValue.getConsumerFoundValue();
         projectAssetValue.getConsumerUsage();
         projectAssetValue.getConsumerValue();
         projectAssetValue.isConsumerValueSelected();
         projectAssetValue.getProjectLeadUsage();
         projectAssetValue.getProjectLeadValue();
         projectAssetValue.isProjectLeadValueSelected();
         projectAssetValue.getAssetUsage();
         projectAssetValue.getAssetValue();
         projectAssetValue.getAssetValueSource();
       }
     }
     // ----------------------------------
     // revert extractions
     repository.extractionResetDatabase();
   } catch (OpenAPIException lEx) {
     System.out.println("ServerCode = " + lEx.getServerErrorCode());
     System.out.println("Message    = " + lEx.getMessage());
     System.out.println("StackTrace:");
     lEx.printStackTrace();
   } catch (RemoteException lEx) {
     lEx.printStackTrace();
   } catch (ServiceException lEx) {
     lEx.printStackTrace();
   } catch (MalformedURLException lEx) {
     lEx.printStackTrace();
   }
 }
 protected static SurveyTaken takeSurvey(FlashlineRegistry repository, AuthToken
authToken, Extraction extraction)
     throws OpenAPIException, RemoteException {
   // ----------------------------------
   // take survey
   SurveyTaken surveyTaken = repository.surveyTakenRead(authToken, extraction);
   Question[] questions = repository.surveyReadQuestions(authToken);
```

```
    ChoiceList choiceList = null;
    Choice[] choices = null;
    Answer[] answers = new Answer[4];
    for(int i=0; i<answers.length; i++){
      answers[i] = new Answer();
    }
    // ---------------------------------
    //Sort questions
    Question[] sortedQuestions = new Question[4];
    for (int i=0;i<questions.length;i++) {
      if (questions[i].getId()==100) {
        sortedQuestions[0] = questions[i];
      }
      if (questions[i].getId()==101) {
        sortedQuestions[1] = questions[i];
      }
      if (questions[i].getId()==102) {
        sortedQuestions[2] = questions[i];
      }
      if (questions[i].getId()==103) {
        sortedQuestions[3] = questions[i];
      }
    }
    answers[0].setQuestionId(sortedQuestions[0].getId());
    choiceList = sortedQuestions[0].getChoiceList();
    choices = choiceList.getChoices();
    answers[0].setChoiceId(choices[0].getId());
    answers[0].setValue(choices[0].getValue());
    answers[1].setQuestionId(sortedQuestions[1].getId());
    answers[1].setChoiceId(0);
    answers[1].setValue("100");
    answers[2].setQuestionId(sortedQuestions[2].getId());
    answers[2].setChoiceId(0);
    answers[2].setValue("200");
    answers[3].setQuestionId(sortedQuestions[3].getId());
    choiceList = sortedQuestions[3].getChoiceList();
    choices = choiceList.getChoices();
    answers[3].setChoiceId(choices[3].getId());
    answers[3].setValue(choices[3].getValue());
    surveyTaken.setAnswers(answers);
    return surveyTaken;
  }
}
```

25.2.15 Use Case: Update the Value Provided for a Project and Asset - Use Predicted Value

Description

Uses the predicted value to update the value-provided for a project and asset.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
```

```
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Answer;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Choice;
import com.flashline.registry.openapi.entity.ChoiceList;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.ExtractionDownload;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectAssetValue;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.Question;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.entity.SurveyTaken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateValueProvidedForProjectAndUserWithPredictedValue {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(
          pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
      // ----------------------------------
      // Update the value provided for a project and asset - use predicted value
      long userid = repository.userReadByAuthToken(authToken).getID(); // the id
 of the user
      long projectid = 50000; // the id of the project
      long assetid = 569; // the id of the asset
      repository.testExtractionResetDatabaseForProject(projectid); // for sample
 *only*
      Project lProject = repository.projectRead(authToken, projectid);
      // ----------------------------------
      // if no user id exists, create a user
      if (userid==0) {
        userid = createUser(repository, authToken).getID();
      }
      ProjectEntities lProjectEntities = new ProjectEntities();
      lProjectEntities.setAddedLeaderIDs(new String[]{ ""+userid });
      repository.projectUpdate(authToken, lProject, lProjectEntities);
      // ----------------------------------
      // Get a RegistryUser for a user
      RegistryUser user = repository.userRead(authToken, userid);
      // ----------------------------------
      // Make sure the project has an extraction
      long[] lAssetIDs = { assetid };
      ExtractionDownload[] extractionDownload =
 repository.extractionCreate(authToken, projectid, lAssetIDs);
      Extraction extraction =
 repository.extractionReadByProjectAndAsset(authToken, projectid, assetid);
```

```
      // ----------------------------------
      // take survey
      SurveyTaken surveyTaken = repository.surveyTakenRead(authToken, extraction);
      Question[] questions = repository.surveyReadQuestions(authToken);
      ChoiceList choiceList = null;
      Choice[] choices = null;
      Answer[] answers = new Answer[4];
      for(int i=0; i<answers.length; i++){
        answers[i] = new Answer();
      }
      // ----------------------------------
      //Sort questions
      Question[] sortedQuestions = new Question[4];
      for (int i=0;i<questions.length;i++) {
        if (questions[i].getId()==100) {
          sortedQuestions[0] = questions[i];
        }
        if (questions[i].getId()==101) {
          sortedQuestions[1] = questions[i];
        }
        if (questions[i].getId()==102) {
          sortedQuestions[2] = questions[i];
        }
        if (questions[i].getId()==103) {
          sortedQuestions[3] = questions[i];
        }
      }
      answers[0].setQuestionId(sortedQuestions[0].getId());
      choiceList = sortedQuestions[0].getChoiceList();
      choices = choiceList.getChoices();
      answers[0].setChoiceId(choices[0].getId());
      answers[0].setValue(choices[0].getValue());
      answers[1].setQuestionId(sortedQuestions[1].getId());
      answers[1].setChoiceId(0);
      answers[1].setValue("100");
      answers[2].setQuestionId(sortedQuestions[2].getId());
      answers[2].setChoiceId(0);
      answers[2].setValue("200");
      answers[3].setQuestionId(sortedQuestions[3].getId());
      choiceList = sortedQuestions[3].getChoiceList();
      choices = choiceList.getChoices();
      answers[3].setChoiceId(choices[3].getId());
      answers[3].setValue(choices[3].getValue());
      surveyTaken.setAnswers(answers);
      // ----------------------------------
      // update survey
      surveyTaken = repository.surveyTakenUpdate(authToken, surveyTaken);
      extraction = repository.extractionUpdateSurvey(authToken, extraction,
surveyTaken);
      // ----------------------------------
      // Get a ProjectAssetValue for a project asset and user.
      ProjectAssetValue projectAssetValue = repository
          .projectAssetValueReadForUser(authToken, projectid, assetid, user);
      if (projectAssetValue != null) {
        // ----------------------------------
        // update the project asset value
        projectAssetValue = repository.projectAssetValueUpdate(
            authToken, projectAssetValue, "predicted_selected");
      }
      // ----------------------------------
```

```
      // revert extractions
      repository.extractionResetDatabase();
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
     String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
     return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

25.2.16 Use Case: Update the Value Provided for a Project and Asset - Use Consumer Value

Description

Uses the consumer value to update the value-provided for a project and asset.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Answer;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Choice;
import com.flashline.registry.openapi.entity.ChoiceList;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.ExtractionDownload;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectAssetValue;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.Question;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.entity.SurveyTaken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateValueProvidedForProjectAndUserWithConsumerValue {
  public static void main(String[] pArgs) {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
```

```
     ////////////////////////////////////////////////////////
     URL lURL = null;
     lURL = new URL(
         pArgs[0]);
     FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
         .getFlashlineRegistry(lURL);
     // ///////////////////////////////
     // Authenticate with OER
     // ///////////////////////////////
     AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
     // ----------------------------------
     // Update the value-provided for a project and asset - use consumer value
     long userID = repository.userReadByAuthToken(authToken).getID(); // ID of
the user being read
     long projectID = 50000; // ID of project being updated
     long assetID = 569; // ID of asset
     repository.testExtractionResetDatabaseForProject(projectID); // for sample
*only*
     Project lProject = repository.projectRead(authToken, projectID);
     // ----------------------------------
     // if no user id exists, create a user
     if (userID==0) {
       userID = createUser(repository, authToken).getID();
     }
     ProjectEntities lProjectEntities = new ProjectEntities();
     lProjectEntities.setAddedLeaderIDs(new String[]{ ""+userID });
     repository.projectUpdate(authToken, lProject, lProjectEntities);
     // ----------------------------------
     // Get the user
     RegistryUser user = repository.userRead(authToken, userID);
     // ----------------------------------
     // Make sure the project has an extraction
     long[] lAssetIDs = { assetID };
     ExtractionDownload[] extractionDownload =
repository.extractionCreate(authToken, projectID, lAssetIDs);
     Extraction extraction =
repository.extractionReadByProjectAndAsset(authToken, projectID, assetID);
     // ----------------------------------
     // take survey
     SurveyTaken surveyTaken = repository.surveyTakenRead(authToken, extraction);
     Question[] questions = repository.surveyReadQuestions(authToken);
     ChoiceList choiceList = null;
     Choice[] choices = null;
     Answer[] answers = new Answer[4];
     for(int i=0; i<answers.length; i++){
       answers[i] = new Answer();
     }
     // ----------------------------------
     //Sort questions
     Question[] sortedQuestions = new Question[4];
     for (int i=0;i<questions.length;i++) {
       if (questions[i].getId()==100) {
         sortedQuestions[0] = questions[i];
       }
       if (questions[i].getId()==101) {
         sortedQuestions[1] = questions[i];
       }
       if (questions[i].getId()==102) {
         sortedQuestions[2] = questions[i];
       }
```

```
        if (questions[i].getId()==103) {
          sortedQuestions[3] = questions[i];
        }
      }
      answers[0].setQuestionId(sortedQuestions[0].getId());
      choiceList = sortedQuestions[0].getChoiceList();
      choices = choiceList.getChoices();
      answers[0].setChoiceId(choices[0].getId());
      answers[0].setValue(choices[0].getValue());
      answers[1].setQuestionId(sortedQuestions[1].getId());
      answers[1].setChoiceId(0);
      answers[1].setValue("100");
      answers[2].setQuestionId(sortedQuestions[2].getId());
      answers[2].setChoiceId(0);
      answers[2].setValue("200");
      answers[3].setQuestionId(sortedQuestions[3].getId());
      choiceList = sortedQuestions[3].getChoiceList();
      choices = choiceList.getChoices();
      answers[3].setChoiceId(choices[3].getId());
      answers[3].setValue(choices[3].getValue());
      surveyTaken.setAnswers(answers);
      // ----------------------------------
      // update survey
      surveyTaken = repository.surveyTakenUpdate(authToken, surveyTaken);
      extraction = repository.extractionUpdateSurvey(authToken, extraction,
 surveyTaken);
      // ----------------------------------
      // Get a ProjectAssetValue for a project asset and user.
      ProjectAssetValue projectAssetValue = repository
          .projectAssetValueReadForUser(authToken, projectID, assetID, user);
      if (projectAssetValue != null) {
        // If a ProjectAssetValue does not exist for this project, asset, and
        // user combination a null value is returned.
        ProjectAssetValue projectAssetValueSelection = new ProjectAssetValue();
        projectAssetValue = repository.projectAssetValueUpdate(
            authToken, projectAssetValue, "consumer_selected");
      }
    } catch (OpenAPIException oapie) {
      System.out.println("ServerCode = " + oapie.getServerErrorCode());
      System.out.println("Message    = " + oapie.getMessage());
      System.out.println("StackTrace:");
      oapie.printStackTrace();
    } catch (RemoteException re) {
      re.printStackTrace();
    } catch (ServiceException se) {
      se.printStackTrace();
    } catch (MalformedURLException mue) {
      mue.printStackTrace();
    }
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

25.2.17 Use Case: Update the Value-Provided for a Project and Asset - Use Project Lead Value

Description

Uses the project lead value to update the value-provided for a project and asset.

Sample Code is as follows:

```
package com.flashline.sample.projectapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Answer;
import com.flashline.registry.openapi.entity.AssetUsageType;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Choice;
import com.flashline.registry.openapi.entity.ChoiceList;
import com.flashline.registry.openapi.entity.Extraction;
import com.flashline.registry.openapi.entity.ExtractionDownload;
import com.flashline.registry.openapi.entity.Project;
import com.flashline.registry.openapi.entity.ProjectAssetValue;
import com.flashline.registry.openapi.entity.ProjectEntities;
import com.flashline.registry.openapi.entity.Question;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.entity.SurveyTaken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class UpdateValueProvidedForProjectAndUserWithLeadValue {
  public static void main(String[] pArgs) {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(
          pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////
      // Authenticate with OER
      // ////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
      // ----------------------------------
      // Update the value provided for a project and asset - use project lead
 value
      long userID = repository.userReadByAuthToken(authToken).getID(); // ID of
 the user being read
      long projectID = 50000; // ID of project being updated
      long assetID = 569; // ID of asset
      float newValue = 50.0f; // Project asset value
      repository.testExtractionResetDatabaseForProject(projectID); // for sample
 *only*
      Project lProject = repository.projectRead(authToken, projectID);
      // ----------------------------------
      // if no user id exists, create a user
```

```
      if (userID==0) {
        userID = createUser(repository, authToken).getID();
      }
      ProjectEntities lProjectEntities = new ProjectEntities();
      lProjectEntities.setAddedLeaderIDs(new String[]{ ""+userID });
      repository.projectUpdate(authToken, lProject, lProjectEntities);
      // ----------------------------------
      // Get a RegistryUser for a user.
      RegistryUser user = repository.userRead(authToken, userID);
      // ----------------------------------
      // Make sure the project has an extraction
      long[] lAssetIDs = { assetID };
      ExtractionDownload[] extractionDownload =
repository.extractionCreate(authToken, projectID, lAssetIDs);
      Extraction extraction =
repository.extractionReadByProjectAndAsset(authToken, projectID, assetID);
      // ----------------------------------
      // take survey
      SurveyTaken surveyTaken = repository.surveyTakenRead(authToken, extraction);
      Question[] questions = repository.surveyReadQuestions(authToken);
      ChoiceList choiceList = null;
      Choice[] choices = null;
      Answer[] answers = new Answer[4];
      for(int i=0; i<answers.length; i++){
        answers[i] = new Answer();
      }
      // ----------------------------------
      //Sort questions
      Question[] sortedQuestions = new Question[4];
      for (int i=0;i<questions.length;i++) {
        if (questions[i].getId()==100) {
          sortedQuestions[0] = questions[i];
        }
        if (questions[i].getId()==101) {
          sortedQuestions[1] = questions[i];
        }
        if (questions[i].getId()==102) {
          sortedQuestions[2] = questions[i];
        }
        if (questions[i].getId()==103) {
          sortedQuestions[3] = questions[i];
        }
      }
      answers[0].setQuestionId(sortedQuestions[0].getId());
      choiceList = sortedQuestions[0].getChoiceList();
      choices = choiceList.getChoices();
      answers[0].setChoiceId(choices[0].getId());
      answers[0].setValue(choices[0].getValue());
      answers[1].setQuestionId(sortedQuestions[1].getId());
      answers[1].setChoiceId(0);
      answers[1].setValue("100");
      answers[2].setQuestionId(sortedQuestions[2].getId());
      answers[2].setChoiceId(0);
      answers[2].setValue("200");
      answers[3].setQuestionId(sortedQuestions[3].getId());
      choiceList = sortedQuestions[3].getChoiceList();
      choices = choiceList.getChoices();
      answers[3].setChoiceId(choices[3].getId());
      answers[3].setValue(choices[3].getValue());
      surveyTaken.setAnswers(answers);
```

```
      // ----------------------------------
      // update survey
      surveyTaken = repository.surveyTakenUpdate(authToken, surveyTaken);
      extraction = repository.extractionUpdateSurvey(authToken, extraction,
 surveyTaken);
      // ----------------------------------
      // Get a ~ProjectAssetValue for a project asset and user.
      ProjectAssetValue projectAssetValue = repository
          .projectAssetValueReadForUser(authToken, projectID, assetID, user);
      if (projectAssetValue != null) {
        // A null value is returned if no If a ProjectAssetValue does not exists
        // for this project, asset, and user combination.
        // ----------------------------------
        // Get an ~AssetUsageType array.
        AssetUsageType[] usageTypes = repository
            .projectAssetValueReadTypes(authToken);
        projectAssetValue.setProjectLeadUsage(usageTypes[1].getName());
// Set the projectAssetValue to a AssetUsageType value.
        projectAssetValue.setProjectLeadValue(newValue); // Set to a new value.
        ProjectAssetValue projectAssetValueSelection = new ProjectAssetValue();
        projectAssetValue = repository.projectAssetValueUpdate(
            authToken, projectAssetValue, "predicted_selected");
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
  protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken
 authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, lUserName, "First", "User",
 "user@example.com", "user", false, false, false);
  }
}
```

26

Relationship Types API

This chapter provides an overview of Relationship Types API and describes the use cases using this API.

This chapter contains the following sections:

- Section 26.1, "Overview"
- Section 26.2, "Use Cases"

26.1 Overview

The Relationship Type defines the structure of a relationship that is used to associate two assets.

Asset Subsystems

When creating or editing assets, Relationship Types are used to define or modify the relationships that exist between assets.

26.2 Use Cases

This section describes the use cases using the Relationship Types API. It contains the following topics:

- Section 26.2.1, "Use Case: Create a new relationship type"
- Section 26.2.2, "Use Case: Modify related assets"
- Section 26.2.3, "Use Case: Query related assets"

26.2.1 Use Case: Create a new relationship type

Description

Creating a new type of relationship to be used between assets.

Sample code is as follows:

```
package com.flashline.sample.relationshiptypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RelationshipType;
```

```
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateNewRelationshipType {
        public static void main(String pArgs[]) throws java.rmi.RemoteException,
 OpenAPIException {
                    try{
         ////////////////////////////////////////////////////////
         // Connect to Oracle Enterprise Repository
         ////////////////////////////////////////////////////////
                    URL lURL = null;
                    lURL = new URL(pArgs[0]);
                    FlashlineRegistry repository =new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
         // ////////////////////////////
         // Authenticate with OER
         // ////////////////////////////
                    AuthToken authToken = repository.authTokenCreate(pArgs[1],
 pArgs[2]);
                    // ---------------------------------------
                    // create the new relationship type
                    String newRelationshipTypeName
 ="My-NewRelationshipTypeName";  //Relationship Type name must contain only alpha
 characters or hyphens
                    RelationshipType newRelationshipType =
 repository.relationshipTypeCreate(authToken, newRelationshipTypeName);
                    System.out.println("The new relationshipType id =
 "+newRelationshipType.getID()+" ");
         // ---------------------------------------
         // set the direction definition and the display text describing the
 relationship type
         //// Two-way = "BIDIRECTIONAL"
         //// Two-way, order matters = "ORDERED-BIDIRECTIONAL"
         //// One-way = "UNIDIRECTIONAL"
         newRelationshipType.setDirection("ORDERED-BIDIRECTIONAL");
         newRelationshipType.setDisplayPrimary("Contained In"); // Source Asset -
 'Contained In' - Target Asset
         newRelationshipType.setDisplaySecondary("Contains");   // Target Asset -
 'Contains' - Source Asset
         newRelationshipType = repository.relationshipTypeUpdate(authToken,
 newRelationshipType);
         // ---------------------------------------
         // delete the new relationship type
         repository.relationshipTypeDelete(authToken,
 newRelationshipType.getID());
                   }catch(OpenAPIException  lEx) {
                     System.out.println("ServerCode = "+
 lEx.getServerErrorCode());
                     System.out.println("Message    = "+ lEx.getMessage());
                     System.out.println("StackTrace:");
                     lEx.printStackTrace();
                   } catch (RemoteException lEx) {
                     lEx.printStackTrace();
                   } catch (ServiceException lEx) {
                     lEx.printStackTrace();
                   } catch (MalformedURLException lEx) {
                     lEx.printStackTrace();
                   }
                 }
}
```

26.2.2 Use Case: Modify related assets

Description

A target asset is related to other assets using My RelationshipType. Using this same relationship type, establish a relationship to an additional asset.

Sample code is as follows:

```
package com.flashline.sample.relationshiptypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RelationshipType;
import com.flashline.registry.openapi.query.RelationshipTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class FindRelationshipTypeAndUseInAsset {
         public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException, ServiceException {
                    try{
          ////////////////////////////////////////////////////////
          // Connect to Oracle Enterprise Repository
          ////////////////////////////////////////////////////////
                      URL lURL = null;
                      lURL = new URL(pArgs[0]);
                      FlashlineRegistry repository =new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
          // ///////////////////////////////
          // Authenticate with OER
          // ///////////////////////////////
                      AuthToken authToken = repository.authTokenCreate(pArgs[1],
 pArgs[2]);
          Asset myAsset = repository.assetRead(authToken, 563);
                      //MY_OTHER_ASSET_ID should be an integer and should be the
 id of an asset in the repository
                      RelationshipType[] allRelationshipTypes =
 getAllRelationshipTypes(repository, authToken);
                      for (int i = 0; i < allRelationshipTypes.length; i++) {
                        if
 (allRelationshipTypes[i].getName().equals("MyRelationshipType2")) {
                          //This is the relationship type, modify the assets that
 are related
                          // using it
                          RelationshipType myRelationshipType =
 allRelationshipTypes[i];
                         Asset otherAsset = repository.assetRead(authToken, 569);
 //569= MY_OTHER_ASSET_ID
                          //MY_OTHER_ASSET_ID should be an integer and should be
 the id of an asset in the repository
                          //add this asset to the list of related assets
                          long[] oldSecondaryIDs =
 myRelationshipType.getSecondaryIDs();
```

```
                        long[] newSecondaryIDs = new long[oldSecondaryIDs.length
+ 1];
                        for (int j = 0; j < oldSecondaryIDs.length; j++) {
                          newSecondaryIDs[j] = oldSecondaryIDs[j];
                        }
                        newSecondaryIDs[newSecondaryIDs.length - 1] =
otherAsset.getID();
                        myRelationshipType.setSecondaryIDs(newSecondaryIDs);
                      }
                    }
                    myAsset.setRelationshipTypes(allRelationshipTypes);
                    repository.assetUpdate(authToken, myAsset);
                  }catch(OpenAPIException  lEx) {
                    System.out.println("ServerCode = "+
lEx.getServerErrorCode());
                    System.out.println("Message    = "+ lEx.getMessage());
                    System.out.println("StackTrace:");
                    lEx.printStackTrace();
                  } catch (RemoteException lEx) {
                    lEx.printStackTrace();
                  } catch (ServiceException lEx) {
                    lEx.printStackTrace();
                  } catch (MalformedURLException lEx) {
                    lEx.printStackTrace();
                  }
                }
  /**
    * This method returns every relationship type in the repository
    * @param repository
    * @param authToken
    * @return
    * @throws RemoteException
    */
   public static RelationshipType[] getAllRelationshipTypes(FlashlineRegistry
repository, AuthToken authToken) throws RemoteException {
     //Create an empty relationship type criteria object
     RelationshipTypeCriteria criteria = new RelationshipTypeCriteria();
     criteria.setNameCriteria("");
     RelationshipType[] allRelationshipTypes =
repository.relationshipTypeQuery(authToken, criteria);
     return allRelationshipTypes;
   }
}
```

Note: Methods to Avoid - SetPromptNotifySecondary()

26.2.3 Use Case: Query related assets

Description

Querying for related asset types.

Sample Code is as follows:

```
package com.flashline.sample.relationshiptypeapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
```

```
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RelationshipType;
import com.flashline.registry.openapi.query.RelationshipTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class FindRelationshipTypeAndUseInAsset {
        public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException, ServiceException {
                    try{
         ////////////////////////////////////////////////////////
         // Connect to Oracle Enterprise Repository
         ////////////////////////////////////////////////////////
                     URL lURL = null;
                     lURL = new URL(pArgs[0]);
                     FlashlineRegistry repository =new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
         // ///////////////////////////////
         // Authenticate with OER
         // ///////////////////////////////
                     AuthToken authToken = repository.authTokenCreate(pArgs[1],
 pArgs[2]);
         Asset myAsset = repository.assetRead(authToken, 563);
                     //MY_OTHER_ASSET_ID should be an integer and should be the
 id of an asset in the repository
                     RelationshipType[] allRelationshipTypes =
 getAllRelationshipTypes(repository, authToken);
                     for (int i = 0; i < allRelationshipTypes.length; i++) {
                       if
 (allRelationshipTypes[i].getName().equals("MyRelationshipType2")) {
                         //This is the relationship type, modify the assets that
 are related
                         // using it
                         RelationshipType myRelationshipType =
 allRelationshipTypes[i];
                        Asset otherAsset = repository.assetRead(authToken, 569);
 //569= MY_OTHER_ASSET_ID
                         //MY_OTHER_ASSET_ID should be an integer and should be
 the id of an asset in the repository
                         //add this asset to the list of related assets
                         long[] oldSecondaryIDs =
 myRelationshipType.getSecondaryIDs();
                         long[] newSecondaryIDs = new long[oldSecondaryIDs.length
 + 1];
                         for (int j = 0; j < oldSecondaryIDs.length; j++) {
                           newSecondaryIDs[j] = oldSecondaryIDs[j];
                         }
                         newSecondaryIDs[newSecondaryIDs.length - 1] =
 otherAsset.getID();
                         myRelationshipType.setSecondaryIDs(newSecondaryIDs);
                       }
                     }
                     myAsset.setRelationshipTypes(allRelationshipTypes);
                     repository.assetUpdate(authToken, myAsset);
                    }catch(OpenAPIException  lEx) {
                     System.out.println("ServerCode = "+
 lEx.getServerErrorCode());
```

```
                    System.out.println("Message    = "+ lEx.getMessage());
                    System.out.println("StackTrace:");
                    lEx.printStackTrace();
                  } catch (RemoteException lEx) {
                    lEx.printStackTrace();
                  } catch (ServiceException lEx) {
                    lEx.printStackTrace();
                  } catch (MalformedURLException lEx) {
                    lEx.printStackTrace();
                  }
                }
   /**
     * This method returns every relationship type in the repository
     * @param repository
     * @param authToken
     * @return
     * @throws RemoteException
     */
    public static RelationshipType[] getAllRelationshipTypes(FlashlineRegistry
 repository, AuthToken authToken) throws RemoteException {
      //Create an empty relationship type criteria object
      RelationshipTypeCriteria criteria = new RelationshipTypeCriteria();
      criteria.setNameCriteria("");
      RelationshipType[] allRelationshipTypes =
 repository.relationshipTypeQuery(authToken, criteria);
      return allRelationshipTypes;
    }
}
```

Example of the RelationshipTypeQuery

```
try
{
   RelationshipTypeCriteria rCriteria = new RelationshipTypeCriteria();
   RelationshipType[] allRelationshipTypes =
 FlashlineRegistry.relationshipQuery(lAuthToken, rCriteria);
}
catch (OpenAPIException e)
{
     e.printStackTrace();
}
catch (RemoteException re)
{
    re.printStackTrace();
}
```

27

Role API

This chapter provides an overview of Role API and describes the use cases using this API.

This chapter contains the following sections:

- Section 27.1, "Overview"
- Section 27.2, "Use Cases"

27.1 Overview

The Role Subsystem provides a Web Services-based mechanism that is used to create, read, update, query, and otherwise manipulate Oracle Enterprise Repository Roles.

Related Subsystems

For more information, see Chapter 30, "User API".

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.Role;
import com.flashline.registry.openapi.query.RoleCriteria;
```

27.2 Use Cases

This section describes the use cases using the Role API. It contains the following topics:

- Section 27.2.1, "Use Case: Manipulate Roles"

27.2.1 Use Case: Manipulate Roles

Description

- Create a new role
- Read a role
- Update a role
- Delete a role
- Query for roles

Sample code is as follows:

```
package com.flashline.sample.roleapi;
```

```
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Role;
import com.flashline.registry.openapi.query.RoleCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class Roles {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ----------------------------------------
      // Create a new role
      String lName = "new_role_type_name";
      String lDesc = "new_role_type_desc";
      boolean lDeafultForNewUser = true;
      Role lRole = null;
      lRole = repository.roleCreate(authToken, lName, lDesc,
          lDeafultForNewUser);
      // ----------------------------------------
      // Read a Role
      String lRoleName = lName;
      Role lRole2 = null;
      lRole2 = repository.roleRead(authToken, lRoleName);
      // ----------------------------------------
      // Update a Role
      String lRoleName3 = lName;
      Role lRole3 = null;
      lRole3 = repository.roleRead(authToken, lRoleName3);
      lRole3.setName("user_modified");
      lRole3.setStatus(20);
      lRole3 = repository.roleUpdate(authToken, lRole);
      // ----------------------------------------
      // Delete a Role
      String lRoleName4 = "user_modified"; // role name must exist in OER
      Role lRole4 = repository.roleRead(authToken, lRoleName4);
      if (lRole4==null) {
        lRole4 = repository.roleRead(authToken, lName);
      }
      if (lRole4!=null) {
        try {
          repository.roleDelete(authToken, lRole4.getName());
```

```
        } catch (OpenAPIException e) {
          e.printStackTrace();
        }
      }
      // ----------------------------------------
      // This method is used to query for roles.
      Role[] lRoles = null;
      RoleCriteria lRoleCriteria = null;
      lRoleCriteria = new RoleCriteria();
      lRoleCriteria.setNameCriteria("user");
      lRoles = repository.roleQuery(authToken, lRoleCriteria);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

28

Subscriptions API

This chapter provides an overview of Subscriptions API and describes the use cases using this API.

This chapter contains the following sections:

- Section 28.1, "Overview"
- Section 28.2, "Use Cases"

28.1 Overview

The subscriptions API provides a mechanism for users to manage the assets to which a user is subscribed. Subscription, in this context, refers specifically to email subscriptions. Subscriptions created through this API are the equivalent of users clicking the Subscribe button on the asset detail page. Once a user subscribes to an asset, they are notified through email of events occurring on the asset. For a list of events for which subscribed users are notified, see the "Email Templates" section in *Oracle Fusion Middleware User Guide for Oracle Enterprise Repository*.

Using the subscriptions API of REX, developers can create, delete, and inspect subscriptions to lists of assets. The operations are always performed for the user identified in the authentication token passed as an argument to the various subscription methods.

28.2 Use Cases

This section describes the use cases using the Subscriptions API. It contains the following topics:

- Section 28.2.1, "Use Case: Create Subscription to Assets"
- Section 28.2.2, "Use Case: Delete Subscription to Assets"
- Section 28.2.3, "Use Case: Read Subscriptions for Assets"
- Section 28.2.4, "Use Case: Read Users Subscribed to an Asset"

28.2.1 Use Case: Create Subscription to Assets

Description

- Authenticate with REX.
- Read a list of asset summaries through a query.
- Subscribe to the matched assets.

Sample code is as follows:

```
package com.flashline.sample.subscriptionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetSummary;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class CreateSubscription {
  public static void main(String pArgs[]) throws OpenAPIException,
      RemoteException, ServiceException {
    try {
      // ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      // ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////////////////////////////////
      // Login to OER
      // ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);
      // ////////////////////////////////////////////////////////
      // find the assets to which to subscribe
      // ////////////////////////////////////////////////////////
      AssetCriteria criteria = new AssetCriteria();
      criteria.setNameCriteria("%");
      AssetSummary[] lAssetSummaries = repository.assetQuerySummary(authToken,
          criteria);
      // ////////////////////////////////////////////////////////
      // Iterate through assets, pulling out the ids and adding
      // to the array of longs
      // ////////////////////////////////////////////////////////
      long[] lAssetIDs = new long[lAssetSummaries.length];
      for (int i = 0; i < lAssetSummaries.length; i++) {
        lAssetIDs[i] = lAssetSummaries[i].getID();
      }
      // ////////////////////////////////////////////////////////
      // Create the subscriptions. The value of "false" for the
      // parameter pFailOnAnyError, causes the operation to NOT
      // fail for any asset to which the user does not have VIEW
      // privileges, or for which the asset is not subscribable.
      //
      // If this value is not "false", the operation throws
      // an exception if any asset in the array of asset IDs is
      // not subscribable or viewable by the user, and NONE of the
      // subscriptions are recorded in the repository.
      // ////////////////////////////////////////////////////////
      repository.subscriptionCreate(authToken, lAssetIDs, false);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
```

```
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

28.2.2 Use Case: Delete Subscription to Assets

Description

- Authenticate with REX.
- Read a list of asset summaries through a query.
- Delete any subscriptions that may exist for the matched assets.

Sample Code as follows:

```
package com.flashline.sample.subscriptionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetSummary;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class DeleteSubscription {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////////////////////////////
      // find the assets for which to delete subscriptions
      ////////////////////////////////////////////////////////////
      AssetCriteria criteria = new AssetCriteria();
      criteria.setNameCriteria("%");
      AssetSummary[] lAssetSummaries = repository.assetQuerySummary(authToken,
 criteria);
      ////////////////////////////////////////////////////////////
      // Iterate through assets, pulling out the ids and adding
```

```
      // to the array of longs
      ////////////////////////////////////////////////////////
      long[] lAssetIDs = new long[lAssetSummaries.length];
      for (int i = 0; i < lAssetSummaries.length; i++) {
        lAssetIDs[i] = lAssetSummaries[i].getID();
      }
      ////////////////////////////////////////////////////////
      // Delete the subscriptions on the list of assets.
      ////////////////////////////////////////////////////////
      repository.subscriptionDelete(authToken, lAssetIDs);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

28.2.3 Use Case: Read Subscriptions for Assets

Description

- Authenticate with REX.
- Read the list of subscribed assets for the authenticated user.

Sample Code is as follows:

```
package com.flashline.sample.subscriptionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ReadSubscriptions {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////
```

```
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////////////////////////
      // Read all of the assets to which the user is subscribed.
      ////////////////////////////////////////////////////////
      long[] lSubscribedAssets =
 repository.subscriptionReadSubscribedAssets(authToken);
      ////////////////////////////////////////////////////////
      // Print out the assets to which the user is subscribed
      ////////////////////////////////////////////////////////
      Asset[] lAssets = repository.assetReadArray(authToken, lSubscribedAssets);
      System.out.println("Subscribed Assets for user "+pArgs[1]);
      for(int i=0; i<lAssets.length; i++){
        System.out.println("  -> "+lAssets[i].getLongName());
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

28.2.4 Use Case: Read Users Subscribed to an Asset

Description

- Authenticate with REX.
- Read the list of users subscribed to a particular asset.

Sample Code is as follows:

```
package com.flashline.sample.subscriptionapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AssetSummary;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.query.AssetCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class ReadSubscribersToAsset {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
```

```
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      ////////////////////////////////////////////////////////
      // Login to OER
      ////////////////////////////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(
          pArgs[1],pArgs[2]);
      ////////////////////////////////////////////////////////
      // Assume that this query returns some number of assets...
      ////////////////////////////////////////////////////////
      AssetCriteria lCriteria = new AssetCriteria();
      lCriteria.setNameCriteria("%");
      AssetSummary[] lAssetSummaries = repository.assetQuerySummary(authToken,
lCriteria);
      ////////////////////////////////////////////////////////
      // Read the users that are subscribed to the first asset
      ////////////////////////////////////////////////////////
      RegistryUser[] lSubscribedUsers =
        repository.subscriptionReadUsersSubscribedToAsset(authToken,
lAssetSummaries[0].getID());
      for (int i=0; i<lSubscribedUsers.length; i++){
        System.out.println("Subscribed Users:
"+lSubscribedUsers[i].getUserName());
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

29

System Settings API

This chapter provides an overview of System Settings API and describes the use cases using this API.

This chapter contains the following sections:

- Section 29.1, "Overview"
- Section 29.2, "Use Cases"

29.1 Overview

Within the Oracle Enterprise Repository's System Settings section administrators can configure the basic operations and enable/disable specific features. The System Settings API provides a mechanism to query these system settings.

> **Note:** Users are allowed only to query the system settings for values, the system settings cannot be set or modified through REX.

To query the system settings, the following package import(s) are required:

```
import com.flashline.registry.openapi.entity.SettingValue;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.query.SystemSettingsCriteria;
```

Reserved Methods

The `systemSettingsAddBundle` method is reserved for future use and is not intended for general use.

29.2 Use Cases

This section describes the use cases using the System Settings API. It contains the following topics:

- Section 29.2.1, "Use Case: Query for System Settings"

29.2.1 Use Case: Query for System Settings

Description

Query for system settings in REX.

Sample code is as follows:

```
package com.flashline.sample.systemsettingsapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.SettingValue;
import com.flashline.registry.openapi.query.SystemSettingsCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class SystemSettings {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ///////////////////////////////
      // Authenticate with OER
      // ///////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      // ///////////////////////////////
      // Set Application Token on AuthToken object.  This is supplied by OER
      //authToken.setApplicationToken("TokenString");
      // ////////////////////////////////////
      //Read all available system settings
      //Create an empty Criteria object.  No criteria returns all settings.
      SystemSettingsCriteria lCriteria = new SystemSettingsCriteria();
      lCriteria.setNameCriteria("enterprise.defaults.displayname.field");
      SettingValue[] lValues = repository.systemSettingsQuery(authToken,
 lCriteria);
      for (int i=0;i<lValues.length;i++) {
        SettingValue lValue = lValues[i];
        System.out.println("Setting Name: " + lValue.getDescriptor().getName());
        System.out.println("Setting Value: " + lValue.getValue());
      }
      // ////////////////////////////////////
      //Read a specific setting
      lCriteria.setNameCriteria("cmee.server.companyname");
      lValues = repository.systemSettingsQuery(authToken, lCriteria);
      for (int i=0;i<lValues.length;i++) {
        SettingValue lValue = lValues[i];
        System.out.println("Setting Name: " + lValue.getDescriptor().getName());
        System.out.println("Setting Value: " + lValue.getValue());
      }
      // ////////////////////////////////////
      //Read a specific section
      lCriteria.setSectionCriteria("general");
      lValues = repository.systemSettingsQuery(authToken, lCriteria);
      for (int i=0;i<lValues.length;i++) {
        SettingValue lValue = lValues[i];
        System.out.println("Setting Name: " + lValue.getDescriptor().getName());
```

```
        System.out.println("Setting Value: " + lValue.getValue());
      }
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

30

User API

This chapter provides an overview of User API and describes the use cases using this API.

This chapter contains the following sections:

- Section 30.1, "Overview"
- Section 30.2, "Use Cases"

30.1 Overview

The User Subsystem provides a Web Services-based mechanism that is used to create, read, update, query, and otherwise manipulate Oracle Enterprise Repository User accounts.

Related Subsystem

For more information, see Chapter 27, "Role API".

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.query.UserCriteria;
```

30.2 Use Cases

This section describes the use cases using the User API. It contains the following topics:

- Section 30.2.1, "Use Case: Manipulating Users"

30.2.1 Use Case: Manipulating Users

Description

- Create a new user.
- Retrieve an existing user.
- Update a user.
- Deactivate a user.
- Query for users.

Sample code is as follows:

```
package com.flashline.sample.userapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Calendar;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.query.UserCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class Users {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
      // ----------------------------------------
      // Create a new user
      String lUserName = "testUserCreate
_"+Calendar.getInstance().getTimeInMillis();
      String lFirstName = "testUserCreate_FirstName";
      String lLastName = "testUserCreate_LastName";
      String lEmail = lUserName+"@example.com";
      String lPassword = "testUserCreate_Password";
      boolean lMustChangePassword = false;
      boolean lPasswordNeverExpires = false;
      boolean lAssignDefafultRoles = true;
      RegistryUser RbacRegistrySecUser = repository.userCreate(
          authToken, lUserName, lFirstName, lLastName, lEmail, lPassword,
          lMustChangePassword, lPasswordNeverExpires, lAssignDefafultRoles);
      // ----------------------------------------
      // Read a User
      long lId = 50000; // user id must exist in OER
      RegistryUser lUser1 = repository.userRead(authToken,
          lId);
      // ----------------------------------------
      // Update a User
      lUser1.setActiveStatus(10);
      lUser1.setUserName("xxx");
      lUser1.setPhoneNumber("412-521-4914");
      lUser1.setMustChangePassword(true);
      lUser1.setPasswordNeverExpires(false);
      lUser1.setPassword("changed_password");
      lUser1.setEmail("newaddress@bea.com");
      try {
```

```
        lUser1 = repository.userUpdate(authToken,
            lUser1);
      } catch (OpenAPIException e) {
        e.printStackTrace();
      }
      // ----------------------------------------
      // Deactivate a User
      RegistryUser lUser2 = null;
      try {
        lUser2 = repository.userDeactivate(authToken, lId);
      } catch (OpenAPIException e) {
        e.printStackTrace();
      }
      // ----------------------------------------
      // Query for Users
      RegistryUser lUsers[] = null;
      UserCriteria lUserCriteria = null;
      lUserCriteria = new UserCriteria();
      lUserCriteria.setNameCriteria("testname");
      lUsers = repository.userQuery(authToken,
          lUserCriteria);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

31

Vendor API

This chapter provides an overview of Vendor API and describes the use cases using this API.

This chapter contains the following sections:

- Section 31.1, "Overview"
- Section 31.2, "Use Cases"

31.1 Overview

Vendors are the original source of assets, and are responsible for their support. Vendors are identified by a single name string.

Validation - When saving a Vendor, Oracle Enterprise Repository currently validates that:

- The vendor name has to be less than 250 characters
- The Vendor name is unique

Related Subsystem

There is a one to many relationship between assets and vendors (i.e. multiple assets can be linked to the same vendor, but an asset can only have one vendor). When creating or editing assets the Vendor ID metadata element linking the Vendor to the asset can also be modified.

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.Vendor;
import com.flashline.registry.openapi.query.VendorCriteria;
```

31.2 Use Cases

This section describes the use cases using the Vendor API. It contains the following topics:

- Section 31.2.1, "Use Case: Manipulating Vendors"

31.2.1 Use Case: Manipulating Vendors

Description

- Adding a new Vendor to Oracle Enterprise Repository.

- Assigning an existing Vendor to an asset.

Sample code is as follows:

```
package com.flashline.sample.vendorapi;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Vendor;
import com.flashline.registry.openapi.query.VendorCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import
 com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;
public class Vendors {
  public static void main(String pArgs[]) throws OpenAPIException,
 RemoteException,
      ServiceException {
    try {
      ////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new
 FlashlineRegistryServiceLocator().getFlashlineRegistry(lURL);
      // ////////////////////////////
      // Authenticate with OER
      // ////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);
      // ----------------------------------------
      // Create a new vendor
      String newVendorName = "My Vendor";
      Vendor newVendor = repository.vendorCreate(authToken, newVendorName);
      System.out.println("The new vendor id =\"" + newVendor.getID() + "\"");
      // ----------------------------------------
      // Find a vendor and update an asset to use it
      VendorCriteria criteria = new VendorCriteria();
      criteria.setNameCriteria(newVendorName);
      Vendor[] vendors = repository.vendorQuery(authToken, criteria);
      long myVendorID = vendors[0].getID();
      long MY_ASSET_ID = 569;
      Asset myAsset = repository.assetRead(authToken, MY_ASSET_ID);
      // MY_ASSET_ID must be the asset id of an asset in the repository
      myAsset.setVendorID(myVendorID);
      repository.assetUpdate(authToken, myAsset);
      // ----------------------------------------
      // clean up
      myAsset.setVendorID(0);
      repository.vendorDelete(authToken, newVendor.getID());
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
```

```
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }
}
```

Index

A

AcceptableValueLists API, 14-1
*see*Overview
*see*Use Cases
Accessing the Repository Assets Pane, 10-14
Accessing the Repository Assets View, 10-14
Add a File to an Asset Using a PVCS Repository, 4-2
Add the Harvest Artifact Store to an Asset, 3-2
Adding a File to an Asset Using the ClearQuest Artifact Store, 2-12
Adding a File to an Asset Using the File Store's Artifact Store, 2-8
Adding ClearQuest, 2-11
Adding Serena Changeman Plug-ins, 4-1
Adding the File Stores Feature to Oracle Enterprise Repository, 2-6
Advanced Configuration, 5-8
Amberpoint, 6-1
Artifact Store, 1-1
*see*Overview
Artifacts Creation, 5-5
ArtifactStore API, 13-1
*see*Overview
*see*Use Cases
Asset API, 15-1
*see*Overview
*see*Sample Code
*see*Use Cases
Asset Consumption Process, 7-3
Asset Production Process, 7-1
AssetType API, 16-1
*see*Overview
*see*Use Cases
Assign an Oracle Enterprise Repository Project to a .NET Solution, 9-22
Assign IDE Users to Oracle Enterprise Repository Projects, 8-1
Assign Users to an Oracle Enterprise Repository Project, 9-13
Associating JDeveloper Application with Oracle Enterprise Repository, 10-3
Atomicity of Method Calls, 12-4
Authentication and Authorization, 12-8
Automated Usage Detection, 7-5
Automatic Usage Detection, 7-6, 10-10, 10-16

B

Basic Concepts, 12-6
Best Practices, 7-1
BPEL PM, 5-5

C

Categorization Types and Categorizations API, 17-1
*see*Overview
*see*Use Cases
ClearCase Integration, 2-1
ClearCase Web Interface, 2-1
see Overview
see Prerequisites
ClearQuest Integration, 2-11
CMF Entry Type API, 18-1
*see*Overview
*see*Use Cases
Command Line Options for the EM Integration, 5-3
Configure an Artifact Store, 2-3, 3-2
Configure the Connection to Oracle Enterprise Repository, 9-21
Configure the Oracle Enterprise Repository Plug-ins, 9-12, 9-19
*see*Installation
*see*Prerequisites
Configure the Oracle Enterprise Repository Preferences, 9-15
Configuring a ClearQuest Artifact Store, 2-11
Configuring a PVCS Artifact Store, 4-2
Configuring an Artifact Store, 1-1
Configuring an Artifact Store For a File Store, 2-8
Configuring Automatic Usage Detection, 9-23
Configuring Eclipse, 9-8
Configuring Oracle Enterprise Repository, 8-1
Configuring Oracle JDeveloper, 9-1
Configuring the Enterprise Manager Integration Utility, 5-6
Configuring VS .NET, 9-18
Configuring Your IDE, 9-1
Consuming WSDL/Service from Oracle Enterprise Repository, 10-4
Create an asset for the ClearCase file(s), 2-4
Creating a File Store, 2-7
Creating and Configuring an Artifact Store

see
Creating and Configuring Repository and Assets, 2-3
CRUD-Q Naming Convention, 12-3
Custom Access Settings API, 19-1
*see*Overview
*see*Use Cases

D

Department API, 20-1
*see*Overview
*see*Use Cases
Developer-Driven Discovery, 7-5
Download Artifacts, 7-6, 10-3, 10-9, 10-15

E

Enable Automatic Usage Detection, 9-16, 9-23
Enable Harvesting in Eclipse, 9-8
Enable Harvesting in VS .NET, 9-18
Enabling the Assets-in-Progress Properties, 9-14
Enabling UTF-8 Support, 2-2
Encrypting the Configuration File Passwords, 5-9
Endpoint Creation, 5-5
Enterprise Manager Integration Utility, 5-1
*see*High Level Use Cases
*see*Overview
*see*Prerequisites
Establish Compliance Templates, 8-1
Exception Handling, 12-9
Extract the asset and the ClearCase file(s), 2-5
Extraction API, 21-1
*see*Overview
*see*Use Cases

F

File Stores, 2-6
see Adding the File Stores Feature to Oracle Enterprise Repository
see Creating a File Store
see Overview
Fundamental WSDL Data Types, 12-4

G

Getting Started, 12-6
Getting Started - Consuming the WSDL, 12-6

H

Harvest Artifacts, 10-2, 10-7, 10-12
Harvest Files, 7-6
Harvest-HTTP Repository Host Integraion
*see*Installation
*see*Overview
Harvest-HTTP Repository Host Integration, 3-1
Harvesting in Eclipse Environment using "External Program", 9-10
Harvesting in Eclipse Environment using ANT, 9-12
High Level Use Cases, 5-2, 7-5

I

Install the Harvester, 8-1
Installation, 3-1, 9-20
Integrating with Oracle JDeveloper 10g, 9-7
Integrating with Oracle JDeveloper 11g R1, 9-7
Integrating with Oracle JDeveloper 11g R2, 9-1
Integration with Amberpoint, 6-1
Integration with Development Environments, 7-1
*see*Best Practices
*see*High Level Use Cases
*see*Overview
Introduction to REX, 12-1

J

Java JDK, 9-14

K

Known Issues, 5-10

L

Link the ClearCase file to the asset, 2-4
Localization of REX Clients, 22-1
Localization of REX Clients API
*see*Overview
*see*Use Cases
Logging, 5-9

M

Metric Mappings, 5-8, 5-9
Metric Publishing, 5-4
Metrics to Update, 5-5

N

No Inter-call Transaction Support, 12-4
Notification API, 23-1
*see*Overview
*see*Use Cases

O

Obtaining the Enterprise Manager Integration Utility, 5-2
Oracle Enterprise Repository Connectors, 0-1
Oracle Enterprise Repository Projects, 10-14
Oracle Service Bus, 5-4
Overview, 1-1, 2-1, 2-6, 3-1, 5-1, 7-1, 13-1, 14-1, 15-1, 16-1, 17-1, 18-1, 19-1, 20-1, 21-1, 22-1, 23-1, 24-1, 25-1, 26-1, 27-1, 28-1, 29-1, 30-1, 31-1
Overview of SFID, 9-23

P

Policies, 7-2

Policy API, 24-1
*see*Overview
*see*Use Cases
Prerequisites, 2-2, 5-1
Prerequisities, 9-20
Prescription Reuse, 7-3
Prescriptive Reuse, 7-6, 10-6, 10-10
Projects API, 25-1
*see*Overview
*see*Use Cases
Propose/Submit Assets, 7-3

Q

Query Considerations in REX, 12-11

R

Relationship Types API, 26-1
*see*Overview
*see*Use Cases
Repository Extensibility Framework, 12-1
REX, 12-1
REX Architecture, 12-2
Role API, 27-1
Running from Command Line, 5-3

S

Sample Code, 15-3
Scheduling from Enterprise Manager, 5-4
Search Oracle Enterprise Repository, 7-6, 10-2, 10-7, 10-12
Security Considerations, 4-3
Selecting a New Artifact Store, 1-3
Serena ChangeMan Integration, 4-1
Set the artifact store from which to extract files, 2-3
Set up Automatic Usage Detection, 8-2
Setting the Repository and Enterprise Manager Connection Information for the Command-line Utility, 5-6
Setting up Eclipse Environment, 9-8
Setting up Eclipse Environment to use Harvester via ANT, 9-11
SiteMinder, 9-14
Submit Files, 7-6, 10-6, 10-12
Subscriptions API, 28-1
*see*Overview
*see*Use Cases
Subsystems Overview, 12-3
Subsytems and CRUD-Q Convention Relationship, 12-4
Supported Versions for the IDEs, 7-7
System Settings API, 29-1
*see*Overview
*see*Use Cases

T

Target Finders, 5-9

U

UDDIMappings.xml, 6-1
Use Cases, 13-1, 14-1, 15-7, 16-2, 17-2, 18-1, 19-2, 20-1, 21-2, 22-2, 23-1, 24-2, 25-1, 26-1, 27-1, 28-1, 29-1, 30-1, 31-1
Use Cases and Supported Development Environments, 7-7
User API, 30-1
*see*Overview
*see*Use Cases
Using DIME attachments with .NET and the Microsoft Web Services Enhancement (WSE) Kit, 12-12
Using Eclipse, 10-6
*see*Automatic Usage Detection
*see*Download Artifacts
*see*Harvest Artifacts
*see*Prescriptive Reuse
*see*Search Oracle Enterprise Repository
*see*Submit Files
*see*View Asset Details
Using File Stores, 2-9
Using Oracle JDeveloper, 10-1
Using SOAP with Attachments and Java AXIS clients, 12-13
Using the Enterprise Manager Integration Utility, 5-2
Using the IDE to Interact with Oracle Enterprise Repository, 10-1
Using the JDeveloper
*see*Download Artifacts
*see*Harvest Artifacts
*see*Prescriptive Reuse
*see*Search Oracle Enterprise Repository
*see*View Asset Details
Using VS .NET, 10-11
*see*Automatic Usage Detection
*see*Download Artifacts
*see*Harvest Artifacts
*see*Search Oracle Enterprise Repository
*see*Submit Files
*see*View Asset Details

V

Vendor API, 31-1
*see*Use Cases
Versioning Considerations for the Oracle Enterprise Repository REX, 12-5
View Asset Details, 7-6, 10-2, 10-7, 10-13

W

Web Services, 5-5
WebLogic 8.1 Tuxedo Plug-In, 2-2
Webshpere 5.x Apache Plug-In, 2-1

X

XML Parsing, 9-14