Oracle® Fusion Middleware

Developer's Guide for Oracle Enterprise Repository

12*c* Release 1 (12.1.3)

E55955-03

March 2015

Describes how to use the Oracle Enterprise Repository Console and how to install and use the Oracle Enterprise Repository Plug-in for Oracle JDeveloper. This guide also describes how to use the Repository Extensibility Framework.

ORACLE

Oracle Fusion Middleware Developer's Guide for Oracle Enterprise Repository, 12*c* Release 1 (12.1.3)

E55955-03

Contents

4 Using the Repository Extensibility Framework

Preface

Oracle Fusion Middleware Developer's Guide for Oracle Enterprise Repository describes how to use the Oracle Enterprise Repository console and how to connect Oracle JDeveloper to interact with Oracle Enterprise Repository. This guide also describes how to use the Repository Extensibility Framework.

Audience

This document is intended for all Oracle Enterprise Repository users who want to use the Oracle Enterprise Repository console or who want to configure Oracle JDeveloper to easily 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

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Enterprise Repository 12*c* documentation set:

- *Oracle Fusion Middleware Concepts Guide for Oracle Enterprise Repository*
- *Oracle Fusion Middleware Installation Guide for Oracle Enterprise Repository*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Enterprise Repository*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*

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.

What's New in This Guide

This preface introduces the new and changed features of Oracle Enterprise Repository and other significant changes that are described in this guide, and provides pointers to additional information. This document is the new edition of the formerly titled *Oracle Fusion Middleware Integration Guide for Oracle Enterprise Repository*.

For a list of known issues (release notes), see the "Known Issues for Oracle SOA Products and Oracle BPM Products for 12c Release 1 (12.1.3.0.0)" at http://www.oracle.com/technetwork/middleware/soasuite/documentation/releasenotes121300-2124738.html.

New and Changed Features for 12*c* Release 1 (12.1.3)

For Oracle Enterprise Repository 12*c* Release 1 (12.1.3), this guide has been updated to include the following new and changed features:

- Chapter 2, "Configuring Oracle JDeveloper to Support Integration with Oracle Enterprise Repository" describes the installation process for the new Oracle JDeveloper plugin for Oracle Enterprise Repository.
- Chapter 3, "Using Oracle JDeveloper to Interact with Oracle Enterprise Repository" details using the JDeveloper plugin to connect to Oracle Enterprise Repository

Other Significant Changes in this Document for 12*c* Release 1 (12.1.3)

For Oracle Enterprise Repository 12*c* Release 1 (12.1.3), this guide has been updated in the following ways:

- User tasks from the previous version of the *Oracle Fusion Middleware User's Guide for Oracle Enterprise Repository* have been moved to this guide.
- Administration tasks from the previous version of this guide have been moved to the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.

Part I

Using the Oracle Enterprise Repository Console

This part describes using the Oracle Enterprise Repository console to find, use, and evaluate assets.

This part contains the following chapter:

1

Getting Started with the Oracle Enterprise Repository Console

This chapter describes how to use the Oracle Enterprise Repository console.

This chapter includes the following sections:

1.1 Getting Started

This section describes the following processes:

1.1.1 Logging in to Oracle Enterprise Repository

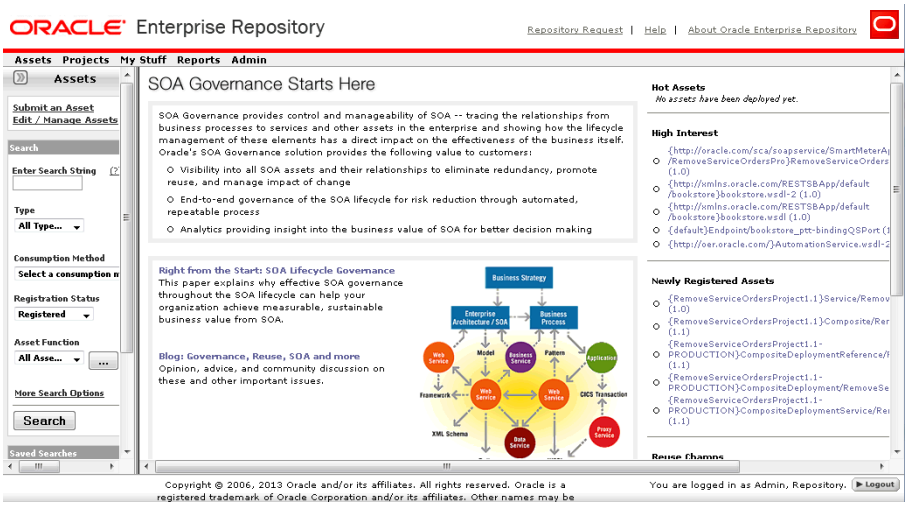
To log in to Oracle Enterprise Repository, perform the following steps:

1. Using a browser, open Oracle Enterprise Repository.
2. On the login screen, enter the appropriate information in the Username and Password boxes.
3. Click **Login**. The Login Status dialog displays a welcome message that includes the date and time of your login.
4. In the Login Stats dialog, click **Close**.

1.1.2 The Default View

After you have logged in, you will see the Oracle Enterprise Repository home page, as shown in Figure 1–1. This page can be customized by the system administrator to display resources, assets, and news relevant to your company.

Figure 1–1 Oracle Enterprise Repository Home Page



The Oracle Enterprise Repository Home Page, showing the Menu Bar, and typical display Assets, Search and Search Criteria and Saved Searches.

****

The Oracle Enterprise Repository Menu Bar

The Oracle Enterprise Repository menu bar runs across the top of the console, and enables you to navigate to the various tools and features within Oracle Enterprise Repository.

The actual links that appear in the menu bar are determined by the permissions granted to your specific role. These links point to corresponding Oracle Enterprise Repository screens, each with specific features and functionality, including:

- **Assets**

  Click to use, submit, edit, and manage assets, as determined by the access settings for your assigned role.

- **Projects**

  Visible and accessible only to users in the appropriate role.

  Click to create, manage, and search for projects, and to assign users to projects.

- **My Stuff**

  Click to view information about assets you have produced or consumed, assets that are assigned to you for review and approval, projects to which you are assigned, your saved searches, or assets to which you have subscribed.

- **Reports**

  Click to view a number of standard reports based on asset usage, value, and compliance. You can create your own custom reports using the reporting tool of your choice. Custom reports can be added to the Reports page.

- **Admin**

  - Visible and accessible only to users in the appropriate role.
  - Enables system configuration after installation, and is used to create users, assign roles, configure access settings, configure system and email settings, and monitor system use.

- **Custom Links**
  - No longer displayed by default.
  - You can configure these additional custom links using the following method:
    a. Open the Oracle Enterprise Repository home page.
    b. Click the **Admin** menu, then **System Settings** option.
    c. Enter **link** in the Search field. The Functional Settings pane is displayed with a list of all the custom links that you can configure.
    d. Click **True** to enable the display of the custom links.
- **Repository Request**

  Click to submit requests to the designated registrar, such as a request for an asset that is unavailable in the registry.
- **Help**

  Provides general information about using Oracle Enterprise Repository.

The Oracle Enterprise Repository Home page can be customized to highlight your company's news and policies. This page includes embedded features that display information about new additions to the repository, the most frequently used assets, and a list of users who have achieved the highest levels of asset reuse. As an alternative, your company can substitute its own home page. In addition, you can create different Home pages that can be displayed based on the user's role. In this way, Business Analysts can see a different set of information than Architects or Development resources. Think of this page as a configurable portal that enables your organization to promote initiatives to all Oracle Enterprise Repository users.

Note: These custom home pages require users to have some exposure to JSP or HTML development as well as the application server's deployment and properties files in order to effectively configure this feature.

1.1.3 Logging Out of Oracle Enterprise Repository

The login indicator, located in the bottom right corner of any Oracle Enterprise Repository page, indicates the user name under which you are currently logged in. A logout link is also provided.

To log out:

1. Click **Logout**. A confirmation message appears.
2. Exit the browser, or click **Log back in**.

1.2 User Role Descriptions

Oracle Enterprise Repository users are defined by roles. Each role has certain permissions that enable users to fulfill specific tasks.

The default roles shipped with Oracle Enterprise Repository are described below. (The Admin page, visible only to users with the appropriate access settings, enables the names of these roles to be changed to suit organizational needs. The Admin page also provides the means to change Access Settings.)

- **User**

  - Anyone with an Oracle Enterprise Repository user name and password is considered to be a user. This role can be assigned as the default role for any new users at the time the user is created. All Oracle Enterprise Repository users can:
    * View news about the company's reuse initiatives
    * Locate, evaluate, use, and download assets
    * View projects
    * Generate reports
    * Submit assets to the registrar

- **Access Administrator**
  - The access administrator creates all Oracle Enterprise Repository users and assigns permissions to them. The access administrator must be familiar with the functions of the Admin page. Typically, access administrators can:
    * Create, view, and edit users and permissions
    * Generate reports

- **Advanced Submitter**
  - The advanced submitter role is typically assigned to asset builders and harvesters.
    * Asset builders focus on building the asset content base. They respond to organization-wide asset needs as well as individual project needs.
    * Harvesters study deployed projects for asset reuse potential and package and submit these assets to the repository.
  - Typically, advanced submitters can:
    * Locate, evaluate, and use and download assets
    * View projects that are associated with assets
    * Generate reports
    * Submit assets to the registrar
    * Edit asset metadata before asset registration

- **Registrar**
  - The registrar is the single point of acceptance or rejection for any asset. There may be more than one person functioning as a repository registrar, depending on the functions addressed. Typically, registrars can:
    * Locate, evaluate, and use or download assets
    * View projects that are associated with assets
    * Generate reports
    * Submit assets to the registrar
    * Edit asset metadata before asset registration
    * Accept assets for the registration process
    * Approve tabs
    * Register assets

        * Apply access settings
        * Edit artifact stores
- **Registrar Administrator**
  - The registrar administrator can use the Type Manager to create and edit asset types, compliance templates, and policy types within the Oracle Enterprise Repository. Typically, registrar administrators can:
    * Locate, evaluate, and use or download assets
    * View projects that are associated with assets
    * Generate reports
    * Submit assets to the registrar
    * Edit asset metadata before asset registration
    * Accept assets for the registration process
    * Approve tabs
    * Register assets
    * Apply access settings
    * Edit artifact stores
- **Project Administrator**
  - Oracle Enterprise Repository tracks asset use at the project level to maintain a history for impact analysis and reporting purposes. Typically, project administrators can:
    * Create, edit, and view projects
    * Generate reports
- **System Administrator**
  - The system administrator configures Oracle Enterprise Repository for use, including, possibly, installation as well as the post-installation configurations. The system administrator typically can:
    * Enable and edit system settings
    * Generate reports

1.3 Exploring the Asset Lifecycle

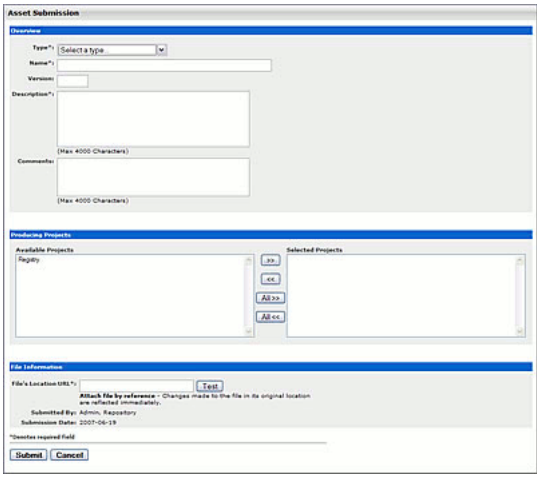
This section describes how you can work with assets in Oracle Enterprise Repository. This section includes the following topics:

1.3.1 Performing a Standard Asset Submission

This procedure is performed on the Assets page.

1. Click **Submit an Asset** in the sidebar. The Asset Submission page is displayed, as shown in Figure 1–2.

Figure 1–2 Asset Submission Page

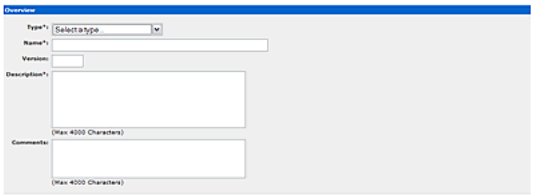


The Asset Submission Page, enabling you to choose and submit an asset by Type, Name, Description, Version, to provide Comments. The second third of the page enables you to choose from available Projects, and the bottom third of the Page enables you to input File Information.

****

2. Select the appropriate Type template to apply to the asset from the list in the Overview section, as shown in Figure 1–3.

Figure 1–3 Overview Section

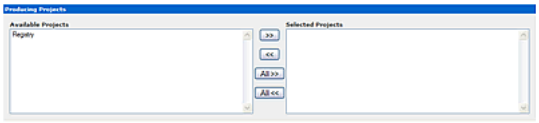


The Overview Section of the Asset Submission Page, showing the list of Type templates which you choose by a Type Selection box, then displays the name, version, description and comments of the Type template in a section for each.

****

3. Enter the appropriate information in the Name, Version, and Description text boxes in the Overview section. (Required)
4. Enter the appropriate information in the Comments section. (Optional)
5. As necessary, designate the project that produced the asset by using the >> and << icons to move the appropriate item from the Available Projects to the Selected Projects columns, as shown in Figure 1–4.

Figure 1–4 Producing Projects Dialog

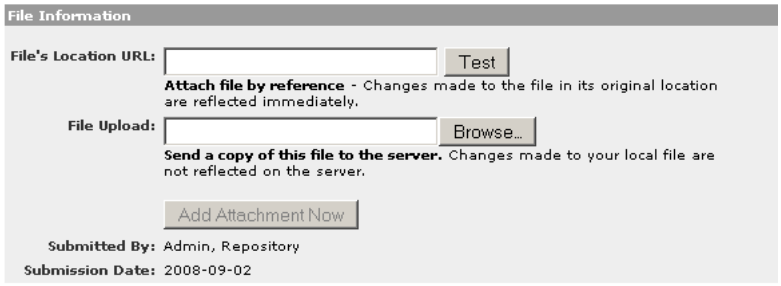


The Producing Projects Dialog on the Asset Submission Page, where you can enter the Location URL of the file containing the asset.

****

6. Enter the URL of the file containing the asset to be submitted in the File's Location URL text box in the File Information section in Figure 1–5.
7. Click **Test** to validate that the file location URL is correct.
8. (Optional) Alternatively, you can browse for the file to send to the server by clicking the **Browse** button that appears at the end of the File Upload field. Then, click the **Add Attachment Now** button.

Figure 1–5 File Information Section



The File Information Section of the Asset Submission page, by the File Location's URL, which shows the File's location URL and a Test button, File Upload text box with a Browse button, and finally the Add Attachment

****

> **Note:** You can disable this requirement by entering `cmee.submission.file.required.enable` into the Admin System Settings Search box and set the value to *False* and click **Save**.

9. When finished, click **Submit**. The Asset Submission confirmation dialog is displayed, confirming that the asset has entered the asset registration queue.
10. Click **Close**.

> **Note:** For information about advanced asset submission and the use of the Asset Editor and the Type Manager, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.

1.3.2 Finding Assets

You can find assets in Oracle Enterprise Repository using various methods. This section includes some of the methods to find assets in Oracle Enterprise Repository:

- Section 1.3.2.1, "Navigating the Asset Tree"
- Section 1.3.2.2, "Performing Standard Searches"
- Section 1.3.2.3, "Using the More Search Options Feature"
- Section 1.3.2.4, "Saving Searches"
- Section 1.3.2.5, "Using EasyLinks"

1.3.2.1 Navigating the Asset Tree

This procedure is performed on the Oracle Enterprise Repository Assets page.

Note: You must set the following system setting to True to enable the Browse Tree:

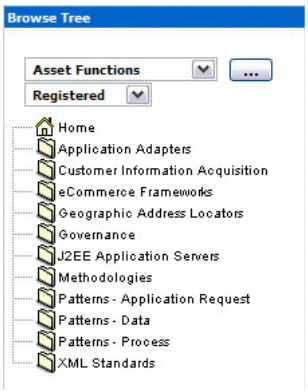
`cmee.jws.pass-all-cookies`

See "System Settings Overview" in *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository* for more information about system settings.

---

1. Click the **Browse** button in the Assets sidebar to open the Browse Tree, as shown in Figure 1–6.

Figure 1–6 Browse Tree

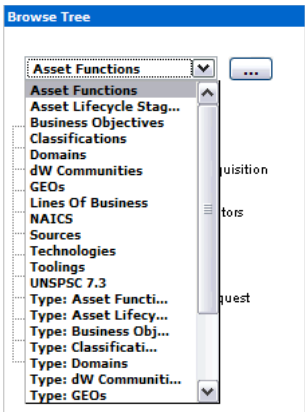


The Browse Tree showing Registered Asset Functions, including Home, Application Adapters, Customer Information Acquisition, eCommerce Frameworks, Geographic Address Location, Governance, J2EE Application Servers, Methodologies, Patterns-Application Request, Patterns-Data, Patterns-Process, XML Standards

****

2. Use the list to filter the results, as shown in Figure 1–7.

Figure 1–7 Browse Tree - Filtering



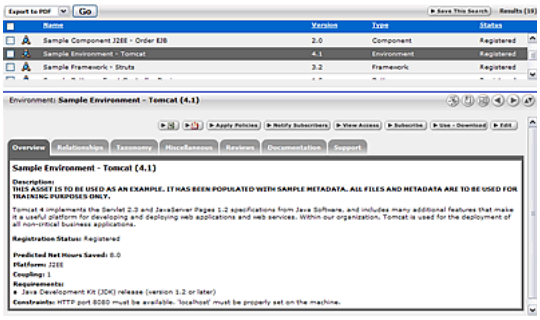
The Browse Tree after filtering, used with the example of Asset Functions

****

3. Click any folder in the tree to display its contents in the main pane.

   The detail of the first item in the results list automatically opens in the lower frame, as shown Figure 1–8.

Figure 1–8 Overview Tab



The Overview Tab, with the lower frame showing the detail of a folder in the tree. The Sample Environment for Tomcat 4.1 is the example.

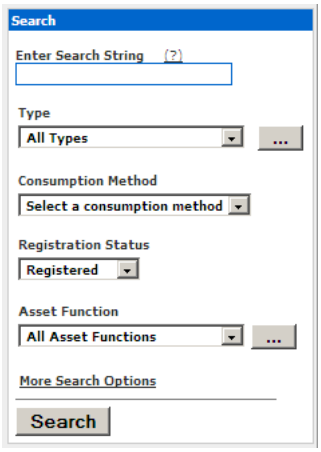
****

1.3.2.2 Performing Standard Searches

This procedure is performed in the Search box in the sidebar of the Assets page.

1. Enter a keyword in the Enter Search String text box.

Figure 1–9 Search Pane

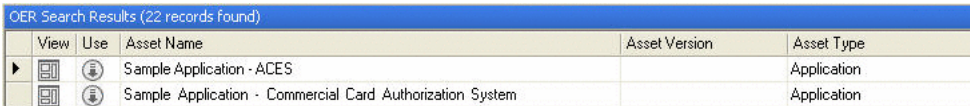


The Search Panel, showing the Search String Text Box where you can enter a Search String, the Type Function List, the Consumption Method Function List, the Registration Status Function List, the Asset Function Function List, plus the option to choose more Search Options.

****

2. Use the Type and Asset Function lists to refine the search.
3. Click **Search**. The search results are listed in the main pane, as shown in Figure 1–10.

Figure 1–10 Search Results



The Search Results listed in the main pane. The example shows Sample Service-Account Detail.

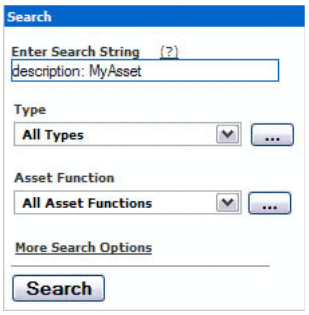
****

4. If necessary, save the search.

1.3.2.2.1 Searching For a Specific Field

This procedure is also performed in the Search box on the Assets page by entering the appropriate search terms in the text box, as shown in Figure 1–11. You can search for specific fields, such as Name, Description, and WSDL. However, you cannot search for custom data fields.

Figure 1–11 Search Pane



Using the Search Pane to search for a specific field. Here, a description field with a search string is specified by entering description followed by a colon and then MyAsset.

****

You can search specific fields, such as Name and Description.

> **Note:** The `cmee.search.specific.field` system setting must be enabled to search specific fields. See "System Settings Overview" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository* for more information about system settings.

- Use the following syntax to perform a search on myAsset in the name field only:

  ```
  name:myAsset
  ```

- Use the following syntax to perform a search on myAsset in the description field only:

  ```
  description:myAsset or desc:myAsset
  ```

- The prefix wsdl: limits the search to the WSDL binding, port type, or port name.

  ```
  Example: wsdl:myService
  ```

Search results appear in the upper section of the main pane.

> **Note:** When changes are made to a custom data field, a log is created indicating that a field has changed. However, the specific details of that change are empty.

1.3.2.3 Using the More Search Options Feature

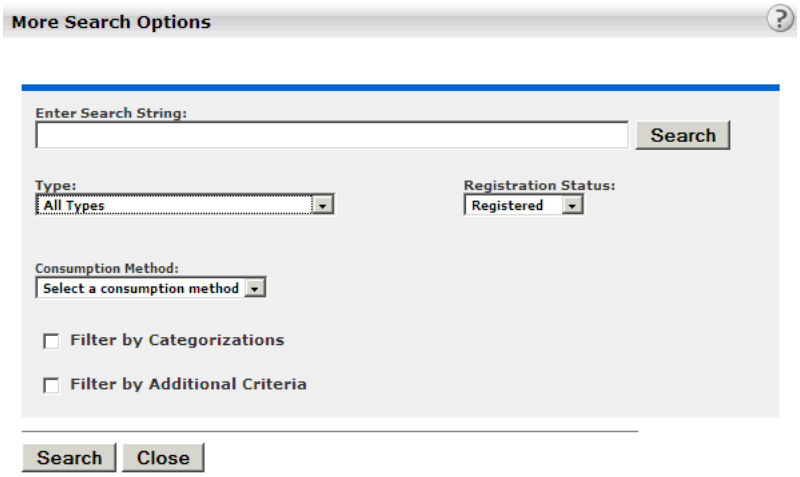
Using the **More Search Options** feature involves a cascading series of selections and features to create a refined set of search parameters. Selections made at each step in the process determines the specific features and functions displayed in the More Search Options dialog.

1.3.2.3.1 Performing a Search Using the More Search Options Feature

This procedure is performed in the Search box in the sidebar of the Assets page.

1. Click the **More Search Options** link. The More Search Options dialog is displayed, as shown in Figure 1–12.

Figure 1–12 More Search Options Dialog

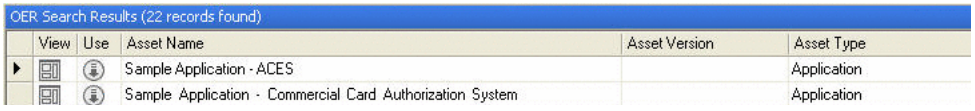


The More Search Options dialog, enabling you to provide a search string, Type, Registration Status, Consumption Method and also Filter by Categorization or Filter by Additional Criteria, either of which you specify by a check box.

****

2. Enter an appropriate search string in the Enter Search String text box.
3. Click **Search** to run the search without further refinement. The search results are listed in the upper frame of the main pane of the Assets page, as shown in Figure 1–13.

Figure 1–13 Search Results



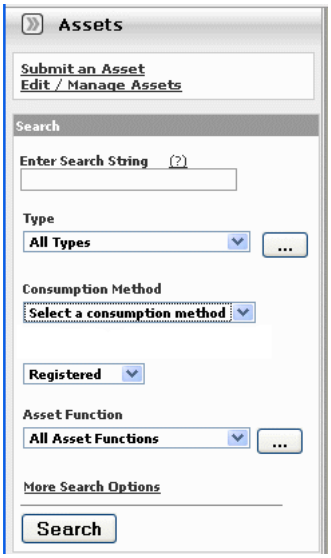
Search Results in upper frame of main pane of Assets page, without search refinement.

****

The More Search Options dialog remains open to allow search refinement. The More Search Options dialog contains the Consumption Method list, as shown in Figure 1–14, which gets displayed only if you install the Harvester Solution Pack.

For more information about using the Consumption Method list, see "Using Consumption Method" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.

Figure 1–14 Assets Pane - Consumption Method



The Assets Pane, showing the Consumption Method.

****

1.3.2.3.2 Filtering More Search Options by Type and/or Registration Status

This procedure is performed in the More Search Options dialog.

> **Note:** Registration Status is available as a search filtering option only if the Oracle Enterprise Repository Assets in Progress option is activated.

1. Select an appropriate type from the Type list.
2. As necessary, select the appropriate registration status from the Registration Status list.
3. Click **Search** to run the search without further filtering. Search results are listed in the upper frame of the main pane of the Assets page.

   The More Search Options dialog remains open to allow search refinement.

1.3.2.3.3 Filtering More Search Options by Categorizations

This procedure is performed in the More Search Options dialog.

1. Select the appropriate option in the Filter by Categorizations section to locate the search string in Any or All of the selected Categorizations, as shown in Figure 1–15.

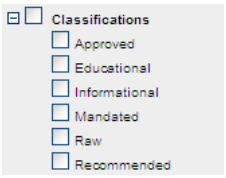
Figure 1–15 Filter by Categorizations

The Filter By Categorizations section, with the Filter by Categorizations section chosen, and the Find results in Any of the categorizations selected below listed. All Categorizations check box is shown, along with Asset Functions, Asset Lifecycle Stages, Business Objectives, Classifications, Domains, Enterprise Architecture Model, Lines of Business, Sources, and Technologies.

****

2. Select the appropriate check box to determine the categorizations to be searched.
3. Click the + next to any categorization to expand the sub-categorization tree to further refine the search, as shown in Figure 1–16.

Figure 1–16 Sub-categorization Tree



The Sub-categorization tree, with the Classifications check box shown and under it Approved, Educational, International, Mandated, Raw and Recommended.

****

4. Click **Search** to run the search without further filtering. The search results are listed in the upper frame of the main pane of the Assets page.

   The More Search Options dialog remains open to allow search refinement.

1.3.2.3.4 Filtering More Search Options by Additional Criteria

This procedure is performed in the More Search Options dialog.

1. Select the **Filter by Additional Criteria** option. The Filter by Additional Criteria section is displayed, as shown in Figure 1–17.

Figure 1–17 More Search Options Dialog - Filter by Additional Criteria

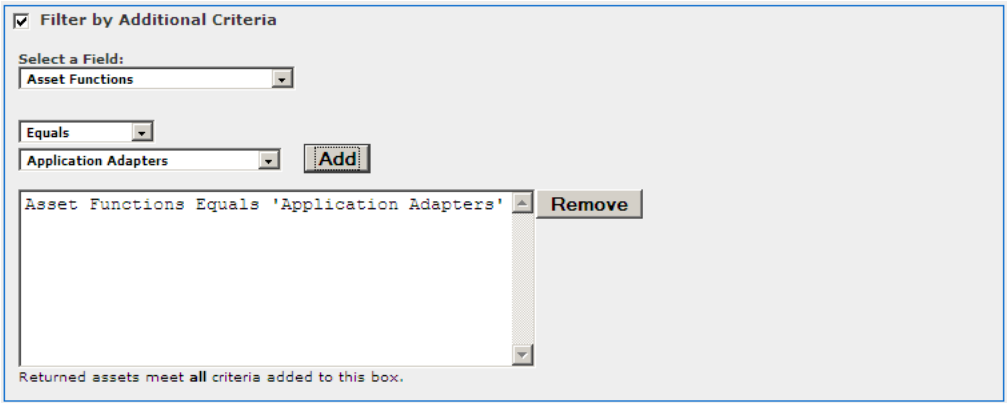
The More Search Options Dialog, providing the option to filter by additional criteria, by selecting a field by which to filter.

****

Note that the available options in the Select a Field list are determined by the selections made in Filtering More Search Options by Type and/or Registration Status, as shown in Figure 1–17.

2. Select one of the options listed in the Select a Field list, as shown in Figure 1–18. This action determines the visibility and contents of subsequent lists in the Filter by Additional Criteria section.

Figure 1–18 Filter by Additional Criteria - Asset Functions



Filter by Additional Criteria, showing the Asset Function Field selected, with Equals and Application Adapter chosen.

****

3. Select the appropriate additional parameter from the available lists. This selection determines whether an additional list or a text box appears to the left of the Add button.
4. As necessary, enter text in the text box, or select an item from the list.
5. Click **Add**. The search parameter appears in the large text box.
6. Add additional parameters as necessary.
7. To remove a parameter:
   - Select (single-click) the parameter.

- Click **Remove**.

8. When the all search parameters have been added, click **Search**.

   Search results appear in the upper frame of the main pane of the Assets page. The More Search Options dialog remains open in the background.

   Note:

   - Search criteria remain intact as long as the More Search Options dialog remains open, even when navigating to other areas in Oracle Enterprise Repository.
   - Click the **More Search Options** link in the sidebar on the Assets page at any time to move the open More Search Options dialog to the foreground, with no change to the criteria established in the current search session.
   - Click **Search** in the More Search Options dialog at any time to refresh the search results and move the search results display to the foreground.

9. If necessary, save the search.

1.3.2.3.5 Searching for Extended Metadata Fields

Certain fields are available for searching on all asset types. These fields are identified by a unique field name, and can be included in criteria for searching for assets of any type. This procedure is performed in the More Search Options dialog.

1. Select the **Filter by Additional Criteria** option.
2. Select an extended metadata field name, such as `import.information` or `internal.import.summary` from the Select a Field list, as shown in Figure 1–19.

   The optional Enter an XPath field is displayed.

Figure 1–19 Filter by Additional Criteria

The More Search Options Dialog, with Filtering by Additional Criteria set, and the import information field selected.

****

3. An Auto Complete feature assists in the entry of XPaths that correspond to the selected extended metadata field.

   For example, an XPath of `/properties/source` searches for assets that were submitted to the Repository from Workspace Studio. And an XPath of `/properties/sync-date` searches for the date that submission from Workspace Studio occurred.

4. When the all search parameters have been added, click **Search**.

5. If necessary, save the search.

1.3.2.4 Saving Searches

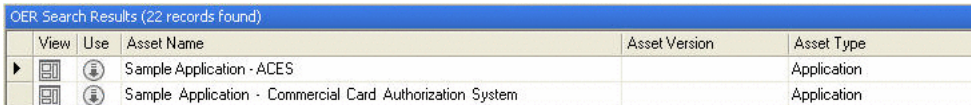
This procedure is performed on the Assets page.

The Saved Searches feature must be activated before searches can be saved.

For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.

1. Perform a standard or More Search Options search. The search results appear in the upper frame of the main pane, as shown in Figure 1–20.

Figure 1–20 Search Results

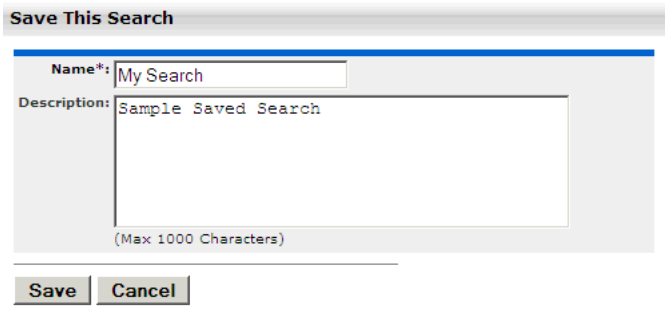


Search Results from a standard or More Search Options search.

****

2. Click **Save This Search**. The Save This Search dialog is displayed, as shown in Figure 1–21.

Figure 1–21 Save This Search Dialog



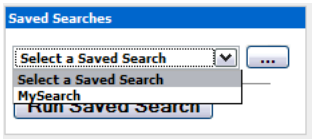
The Save This Search Dialog with the Name of the Search provided and a description of the Search.

****

3. Enter the appropriate information in the Name and Description text boxes.

4. Click **Submit**. The saved search is displayed in the Saved Searches list, as shown in Figure 1–22.

Figure 1–22 Saved Searches Dialog

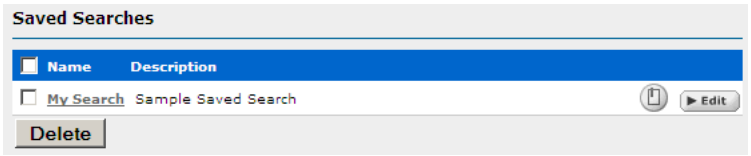


The Saved Searches Dialog list, showing a list of saved searches.

****

The saved searches are also listed in the Saved Searches section in the My Stuff page, as shown in Figure 1–23.

Figure 1–23 Saved Searches Pane in My Stuff Page



The saved searches pane in the My Stuff page, with a name and description of the search listed.

****

1.3.2.4.1 Executing a Saved Search

From the Assets page:

1. Select the search to be executed from the Saved Searches list.
2. Click **Run Saved Search**. The search results are listed in the upper frame of the main pane.

1.3.2.4.2 Managing Saved Searches

These procedures are performed in the Saved Searches section in the My Stuff page.

Executing a Saved Search

1. Click the text link for the search to be executed. The search results appear in the upper frame of the main pane on the Assets page.

Editing a Saved Search

1. Click **Edit** next to the saved search. The Save this Search dialog is displayed.
2. Make changes as appropriate to the text in the Name and Description fields. (Search parameters cannot be edited.)
3. Click **Save**.

Deleting a Saved Search

1. Select the search option that you want to delete. (Select the Name check box to select ALL saved searches.)
2. Click **Delete**. The selected search is deleted.

Linking to a Saved Search

1. Right-click the text link and copy the URL. The browser's dialog is displayed.
2. Use the copied link information to create a direct link to the saved search. The correct syntax is: *http://servername/appname/index.jsp?savedsearchid=SEARCHID*, where SEARCHID is the ID number of the saved search.

Factors Affecting Search Results

- Search results may change over time as assets are added to/removed from Oracle Enterprise Repository.
- Individual search results are subject to user permission settings.
- Activation of the Assets-in-Progress feature may affect the results of existing Saved Searches.

1.3.2.5 Using EasyLinks

The **EasyLink** icon is displayed on assets and saved searches, as shown in Figure 1–24. These icons are standard HTML image links that can copied to the clipboard and pasted into documents or email messages as pointers to specific registry artifacts.

Figure 1–24 EasyLink Icon

The EasyLink icon.

****

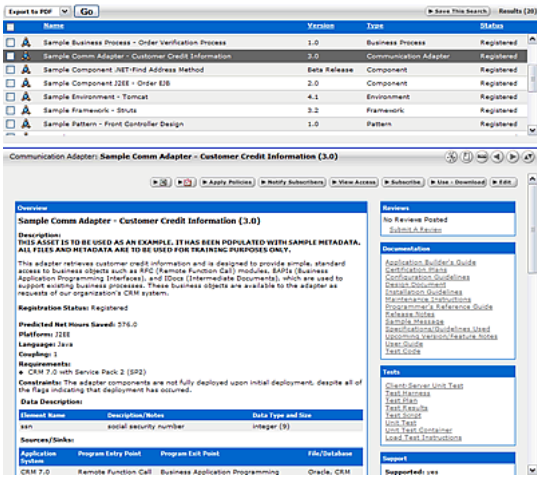
To add a link to this detail page, click and drag the EasyLink into an email or other document to create a link to the currently displayed asset.

1.3.3 Evaluating Assets

The Asset Detail can be viewed in tabular or frame format. The Asset Detail page provides a wide variety of information on the use, functionality, and history of an asset, as shown in Figure 1–25. Typically, asset metadata is provided by the asset producer and is reviewed by the registrars before presentation through Oracle Enterprise Repository. The information that is captured and displayed can be customized by your organization through the Asset Editor and Type Manager.

After performing an asset search, click any asset listed in the search results to open its Asset Detail in the bottom frame.

Figure 1–25 Asset Detail Page



The Asset Detail Page, showing one asset selected and detail on that asset open in the bottom frame.

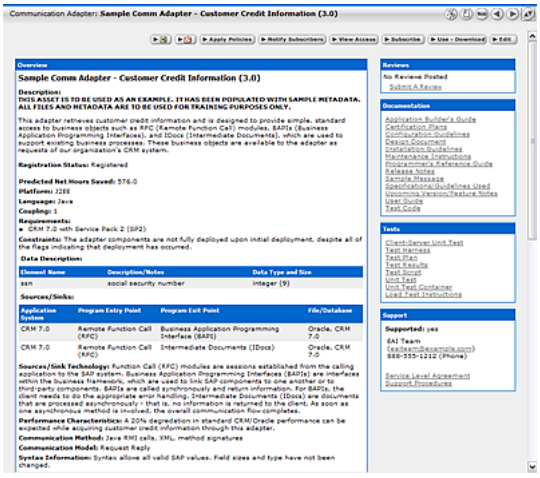
****

Search results on the Asset Search in the Web Console are now limited by default, to improve initial search performance. The number of results is configurable on the Admin | System Properties screen: "`cmee.search.assets.maxresults`". If a search exceeds the maximum number of results, the search results screen shows a result count like "Results (1000 of 2345)", and a link "Show All" that brings back all of the results.

There are navigation icons in the title area of the asset detail. These icons offer a variety of viewing and navigation options to aid in the evaluation of assets.

- Click the **Hide/Display Search Results** icon to expand the Asset Detail, as shown in Figure 1–26. (Click again to restore the search results list in the upper frame.)

Figure 1–26 Asset Detail Page

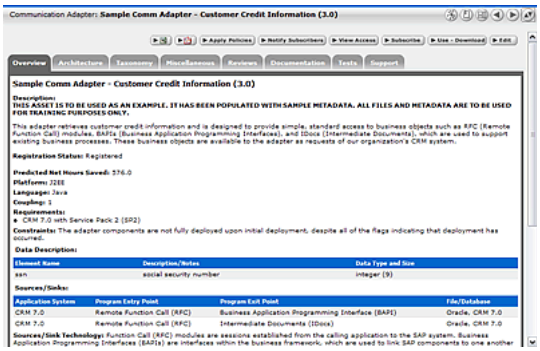


The Asset Detail Page, with the Asset Detail expanded to show more information about the Asset.

****

- Click the **View Details of This Item in Tabs** icon to toggle the Tabbed View mode, as shown in Figure 1–27.

Figure 1–27 Asset Detail Page



The Asset Detail Page in Tabbed View mode, with information from each element in the default asset detail on its own tab.

****

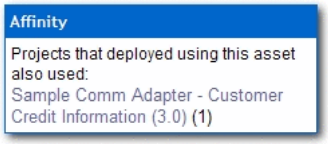
The Tabbed View places the information from each element in the default asset detail on its own tab. When viewing assets of the same type in Tabbed View mode, the selected tab appears for each asset selected from the search results, facilitating easy comparison of asset information. When viewing assets of different types, the selected tab appears only if the assets have the tab in common. Otherwise, the system defaults to the first tab listed in the asset detail.

- Click the **View Next/Previous Item in Search Results** icon to navigate to the next or previous asset in the search results.

Asset Affinity

Within the context of Oracle Enterprise Repository, Affinity is the condition that occurs when two or more assets are deployed in the same project. The Affinity element appears in the asset detail for each asset that meets this condition, and lists other assets that have (deployed) projects in common.

Figure 1–28 Affinity



The Affinity element as it appears in the asset detail for the element.

****

Affinity provides valuable contextual information on asset use.

1.3.3.1 Using the Navigator

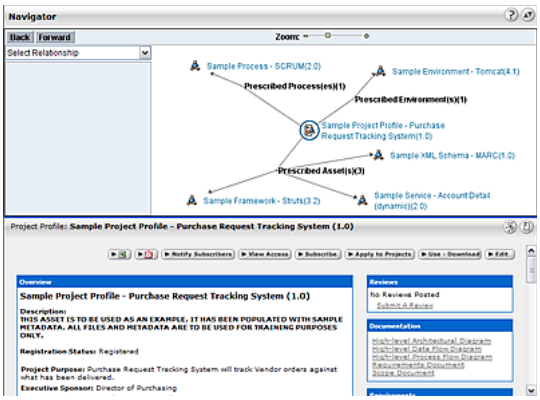
The Navigator provides a dynamic graphical representation of the inter-relationships among assets and projects.

Prerequisite

For information about activating the Oracle Enterprise Repository Navigator, see the "Projects and Navigator" section in the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.

Click the **Launch the Navigator** icon in the detail display of any asset or project. The Navigator page is displayed, as shown in Figure 1–29.

Figure 1–29 Navigator Page



The Navigator Page with the graphical display of the selected asset and related assets shown, and the lower frame showing the asset's detail.

****

The Navigator View

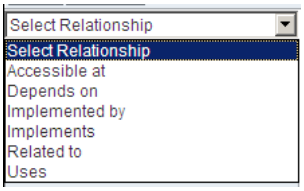
The upper area of the Navigator presents a dynamic, graphical display of the selected asset and all related assets and projects. The selected asset appears in the center of the upper area of the Navigator display, as indicated by the asset icon within a blue circle. Any related assets and producing (the Producing Projects feature must be enabled) and/or consuming projects are displayed as nodes branching off the center. When a Project is centered, all parent and child projects are displayed, as are all assets produced or consumed by the project.

The lower frame of the Navigator presents the asset's detail, similar to the display on the Assets page.

Relationship View

The Select Relationship menu displays a list of relationships relevant to the in-focus asset, as shown in Figure 1–30.

Figure 1–30 Select Relationship Menu



The Select Relationship Menu displaying a list of relationships relevant to the in-focus asset.

****

Click any relationship in the Select Relationship list to view a list of assets or projects bearing the specified relationship to the focus asset. Click any item in that list to move that item to the center of the Navigator display.

Node Display Options

Right-click any node in the display to open a menu of display options. The available options are determined by the type of node and its current display state.

- Enable All Motion

  Enables items in the display to be dragged around the window to aid viewing.

- View Detail
  - Displays the item's detail in the bottom pane.
  - Action can also be toggled by double-clicking the node icon.

If you right-click the any node other the node in the centre, the following options are visible to you:

- Center

  Shifts the focus to the selected item, and displays any available related assets or projects. (Does not change the detail display in the bottom pane.) Action can also be toggled by single-clicking the node icon.

- View Detail
  - Displays the item's detail in the bottom pane.
  - Action can also be toggled by double-clicking the node icon.

Double-clicking any relationship node (Depends on, Producing Projects, and so on) expands or collapses the node.

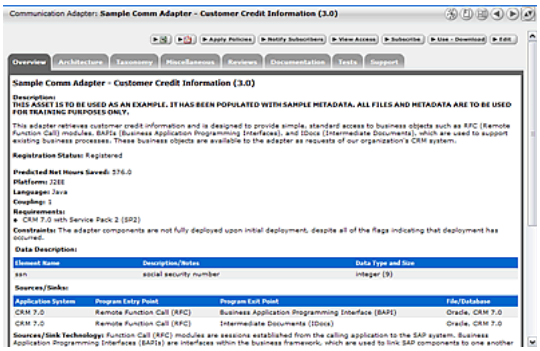
1.3.4 Using/Downloading Assets

Using/Downloading an asset enables users to obtain the payload files associated with an asset. For Service assets, the WSDL file might be included as the asset payload. Using/Downloading an asset also initiates asset usage tracking. You have to select the project on which you are using the asset, and the usage is attributed to that project. Email notifications can be initiated after a configurable period of time that ask the user to indicate whether they are still evaluating the asset, have tentatively accepted it for use, or have rejected it.

Perform the following steps to download an asset and its associated files for evaluation.

1. Use Search or other means to locate the asset to be used.
2. Click **Use / Download** in the asset detail, as shown in Figure 1–31.

Figure 1–31 Asset Details Page

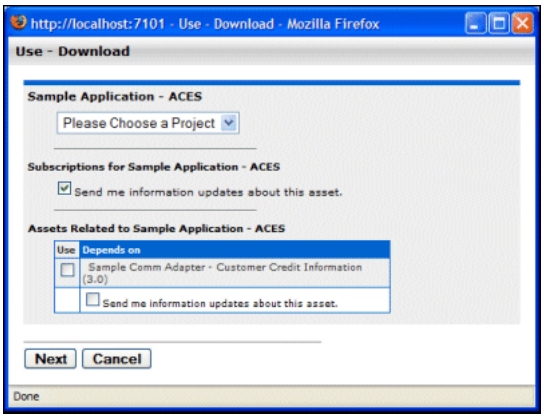


The Asset Details Page with Click/Download clicked.

****

The Use - Download page is displayed, as shown in Figure 1–32.

Figure 1–32 Use - Download Page

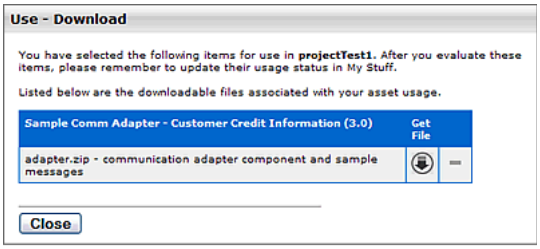


The Use-Download page, showing a chooser for the Sample Application Project.

****

3. Use the Please Choose a Project list to assign the asset to a project (mandatory).
4. Make the appropriate selection in the Subscription check box.
5. As necessary, select any available related assets for inclusion in the download and/or subscription.
6. Click **Next**. A new dialog is displayed to display the downloadable files in the asset payload.

Figure 1–33 Use - Download Page



Downloadable files displayed in the asset payload.

****

7. Click the **Get File** icon. The File Download dialog is displayed.
8. Proceed with the download as necessary.
9. Repeat Steps 7 and 8 as necessary to download other files in the asset payload.
10. When finished, click **Close** to close the Use - Download dialog.
11. After evaluating the asset, be sure to update the asset's status on the My Stuff page.

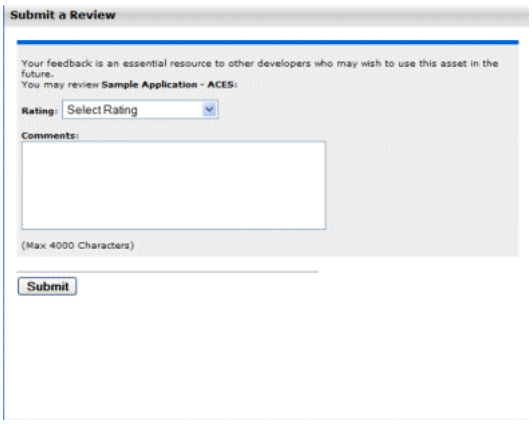
1.3.5 Reviewing Assets

Individuals who have used an asset can provide a written review and a rating for the asset. These reviews are useful to other users, as well as to those who manage the assets. It provides an opportunity for users to share their experiences with others in the community.

Follow these steps to rate an asset and to submit a review.

1. Use Search or other means to locate the asset to be reviewed.
2. Click **Submit a Review** in the Reviews section of the asset detail. The Submit a Review dialog is displayed, as shown in Figure 1–34.

Figure 1–34 Submit a Review



3. Select the appropriate rating from the **Rating** list.
4. Enter the appropriate review information in the **Comments** text box.
5. Click **Submit**. A confirmation message appears.
6. Close the window, or click **click here** to display the review. The review is now listed on the Reviews tab in the asset detail.

1.3.6 Subscribing to Asset Information

Users subscribed to an asset are automatically notified by email if a new version has been created or registered. Manual notification can also be used to communicate any metadata changes to the asset.

1. Use Search or other means to locate an asset.

2. In the Asset window, double-click the asset to display the details of the asset.
3. In the asset details section that appears below the list of assets, click the **Subscribe** option that appears in the extreme right corner. The Subscribe dialog is displayed.
4. Click the **Subscribe** button. The asset subscription dialog with the following message is displayed:

   Send me information updates about this asset.
5. This option is selected by default, click **Close**. The asset is added to the list of asset subscriptions on the My Stuff page.

1.3.7 Exporting Asset Details

The Asset Detail frame on the Assets page provides a wide variety of information on the use, functionality, and history of an asset. The asset details on an asset can be exported in any one of the following forms:

- Section 1.3.7.1, "Exporting Detail to Excel"
- Section 1.3.7.2, "Exporting Detail to PDF"
- Section 1.3.7.3, "Exporting Asset Search Results to ZIP"
- Section 1.3.7.4, "Exporting API Assets to API Catalog"

1.3.7.1 Exporting Detail to Excel

You can export details to an Excel file using the following methods:

- Section 1.3.7.1.1, "Export an Excel File from Search Results"
- Section 1.3.7.1.2, "Export an Excel file from the Detail Page"

> **Note:** The export will fail if a field containing a hyperlink (for instance, the Endpoint URI or WADL URL fields) contains more than 128 characters.

1.3.7.1.1 Export an Excel File from Search Results

This procedure is performed on the Assets page.

1. Perform an asset search by entering keywords in the Enter Search String text box in the Search section. Use the Type and Asset Function list as necessary to refine the search.
2. Select one or more assets from the search results by placing a check in the check box that appears to the left of assets listed in the Search Results pane.
3. Select **Export to Excel** in the list at the top of the pane.

> **Note:** The Export to Excel option is displayed, only when the `cmee.asset.registry.export.excel` property is enabled. This is enabled by default, but if you are migrating, then you might face an issue if this property is not enabled.

4. Click **Go**. Export progress is indicated in the progress window, with the following options:
   - Click **Terminate Job** to end the export.

- Click **View Audit Log** to open a log of the export process.

5. When the export is complete, click **View Excel Spreadsheet** in the progress window. The spreadsheet opens in Microsoft Excel.

   Assets are listed horizontally; Asset Detail elements are indicated by the light-green background.

   When assets of multiple types are exported, each type is assigned to a different worksheet tab. Note the following:

   - The specific configuration of each asset type template determines the asset detail elements that are exported to the spreadsheet.
   - System limitations prevent the export of the Relationships, Keywords, Contacts, Reviews, and Usage elements, and of tables, images, or HTML formatting.
   - Limitations in Excel restrict export to no more than 250 assets of the same type during the same export.

   Note: To use Export Policies To Excel from Asset Search Results Page, perform the following steps:

   1. Log in to https://support.oracle.com.
   2. Click the **Knowledge** tab. The Knowledge Home page is displayed.
   3. In the top-right corner of the screen, select Article ID from the Sources list.
   4. Enter 952047.1 in the space for entering the article ID number, and click the **Browse** icon.

      The Knowledge Browser window is displayed with this article - Unable to Export Policies To Excel from Asset Search Results Page [ID 952047.1].

1.3.7.1.2 Export an Excel file from the Detail Page

This procedure is performed on the Assets page.

1. Locate the appropriate asset.
2. Click the **Excel** icon in the detail page. Export progress is indicated in the progress window.
3. When the export is complete, click **View Excel Spreadsheet** in the progress window to open the spreadsheet in Microsoft Excel.

   Note: The specific configuration of each asset type template determines the asset detail elements that are exported to the spreadsheet.

1.3.7.2 Exporting Detail to PDF

This procedure is performed on the Assets page.

Note:

- Only assets of Types to which an XSLT printing template is associated are available for export to PDF.
- In the event that the asset selected for export is of a Type to which no printing template is associated, you are notified through a dialog that the export cannot proceed. If multiple assets are selected for export, the dialog lists the Types that cannot be exported. You can then: (a) proceed with the export of any assets of Types with associated printing template, or (b) cancel the export.
- Notify the registrar of any assets that cannot be exported to PDF.

1.3.7.2.1 Exporting from the Detail Page

You can export the asset details to PDF from the Asset Detail pane.

1. Use Search or other means to locate the appropriate asset.
2. Click the **PDF** icon in the asset detail.
3. If prompted to do so, select a printing template from the list.

   If the Adobe Acrobat plug-in is installed, the browser opens the PDF. Otherwise, you are prompted to save or open the file.

1.3.7.2.2 Exporting from the Search Results List

1. Perform an asset search by entering keywords in the Enter Search String text box in the Search section.

   Use the Type and Asset Function list as necessary to refine the search.
2. Select one or more assets from the search results by placing a check in the check box that appears to the left of assets listed in the Search Results pane.
3. Select **Export to PDF** from the list above the list of assets.
4. Click **Go**.
5. If the Adobe Acrobat plug-in is installed, the browser opens the PDF. Otherwise, you are prompted to save or open the file.

1.3.7.3 Exporting Asset Search Results to ZIP

When enabled, the Export to ZIP feature allows the export of asset information to a ZIP file, through a menu selection on the asset search results screen.

- The Export to ZIP feature is available only to users with Administrator permissions.
- Import/Export must be enabled. For more information, see the "System Settings Overview" chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.
- The exported file contains all asset metadata in XML format, including relationships.

This procedure is performed on the Assets page.

1. Perform an asset search by entering keywords in the Enter Search String text box in the Search section. Use the Type and Asset Function list as necessary to refine the search.
2. Use the check boxes in the search results list to select the assets to be exported to ZIP.
3. Select **Export to ZIP** from the list.
4. Click **Go**. The Export to ZIP progress dialog is displayed.
5. When the export is complete, click **Download** to download the exported zip file. The File Download dialog is displayed.
6. Select **Open**, **Save**, or **Cancel**, as appropriate.

1.3.7.4 Exporting API Assets to API Catalog

After installing the optional Export to API Catalog feature patch, you can export API assets from Oracle Enterprise Repository directly to a running Oracle API Catalog instance, or you can create a zip file containing API assets that can be imported into API Catalog using the Import/Export tool.

- The Export to API Catalog feature is available only to users with Administrator permissions.
- Import/Export must be enabled in both OER and OAC. For more information, see the "System Settings Overview" chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository* and the "System Settings Overview" chapter in the *Oracle Fusion Middleware Administrator's Guide for Oracle API Catalog*.

See "Installing the Export to API Catalog Feature Patch" in *Oracle Fusion Middleware Installation Guide for Oracle Enterprise Repository* for prerequisites and information about installing the Export to API Catalog feature patch. Section 1.3.7.4.1 describes the API asset metadata that will be imported into API Catalog.

> **Note:** You can export only API assets into API Catalog. Selected assets of other types will not be exported.

1.3.7.4.1 API Asset Metadata Exported to API Catalog The following metadata associated with API Assets is exported to API Catalog:

> **Note:** Metadata not discussed in this section is not imported with the API assets into API Catalog.

Metadata Element	Supported in API Catalog
Name	Yes
Version	Yes
Keywords	Yes
API Status	Set to **Draft** for all exported assets
Description	Yes

Metadata Element	Supported in API Catalog
Documentation URL	Yes. Only the first Documentation item in OER will be displayed in API Catalog. **Note**: Documents stored in an artifact store will not be exported. Only documents with an associated URL will be exported. Note: the Documentation reference may not appear in API Catalog until the API is edited and saved.
Icon	Yes. See Section 1.3.7.4.2 for details.

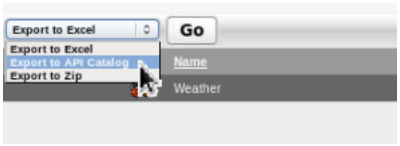
1.3.7.4.2 Exporting Icons with API Assets You can also export an icon for an API asset by completing the following task before export:

1. In the Asset Editor, select the API asset to which you want to add an icon.
2. From the Miscellaneous tab, click **Add** in the File Information section.
3. Enter `icon` into the **Name** field.
4. Enter the URL at which the icon is located into the **URL** field.

1.3.7.4.3 Exporting API Assets to API Catalog Using the Export to API Catalog Feature To export API assets to API Catalog:

> **Note:** API Assets that have already been exported to API Catalog are not overwritten. Exporting these assets again will have no effect. If you want to export updated API assets to API Catalog, you must first remove the assets from API Catalog, as described in "Deleting an Asset" in the *Oracle Fusion Middleware Administrator's Guide for Oracle API Catalog*, and then export the assets again.

1. Perform an asset search by entering keywords in the **Enter Search String** text box in the Search section. Select **API** from the Type list to display only API asset types in the search results.
2. Use the check boxes in the search results list to select the assets to be exported to API Catalog.
3. Select **Export to API Catalog** from the list, as shown in the following figure.



This image displays the Enterprise Repository search results page. One result is displayed. The Export To API Catalog option is selected in the drop down list. The Go button is also displayed.

****

4. Click **Go.**
5. To export the API assets to a running API Catalog Instance:
   a. Select the API Catalog server to which you want to export the selected API assets.

Note: If no API assets are selected, **Export to API Catalog** and **Export to Zip** buttons are not clickable and the following message is displayed: `No assets of the type API have been selected`.

b. Click **Export to API Catalog** to export the API assets directly to a running API Catalog server, as shown in the following figure.

The Select API Catalog Server dialog is displayed. Two API Catalog servers are displayed: Dev and Prod. Prod is selected. The mouse pointer is hovering over the Export to API Catalog button. The Export to Zip button is also displayed.

****

c. Wait until the Import to API Catalog dialog displays Complete, as shown in the following figure.

The Import to API Catalog dialog is displayed. Details about the import operation are displayed. The Status field displays "completed", indicating the operation has completed successfully.

****

d. Confirm that the selected assets now appear on the API Catalog server.

6. To export the API assets into a zip file you can later import into API Catalog using the Import/Export tool:

a. Click **Export to Zip** to export the API assets to a zip file that can later be imported into an API Catalog instance using the Import/Export utility.

Note: If no API assets are selected, **Export to API Catalog** and **Export to Zip** buttons are not clickable and the following message is displayed: `No assets of the type API have been selected`.

b. When the export is complete, click **Download** to download the exported zip file. The File Download dialog is displayed.

c. Select **Open**, **Save**, or **Cancel**, as appropriate.

d. As a user with administrative privileges, import the zip file into API Catalog, as described in "Importing Items into Oracle API Catalog" in *Oracle Fusion Middleware Administrator's Guide for Oracle API Catalog*, to import the API assets into API Catalog.

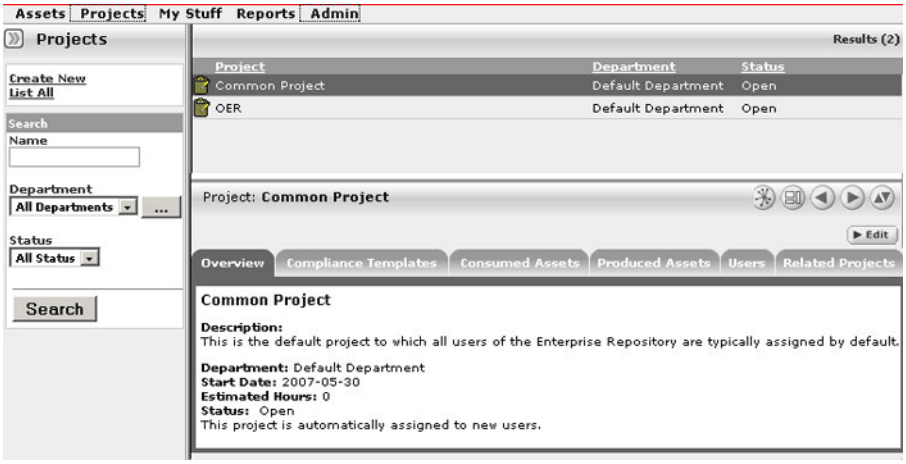
1.4 Projects

Projects are the primary means of gathering metrics in Oracle Enterprise Repository. Oracle Enterprise Repository tracks assets produced by projects, as well as assets consumed by projects. Oracle Enterprise Repository users are assigned to projects, and when a user submits a new asset, they are prompted for the producing project. Likewise, when a user wants to reuse an asset, they are prompted for the project on which the asset is reused. This enables Oracle Enterprise Repository to generate reports on the reuse savings per project. It also enables Oracle Enterprise Repository to report on the savings generated by asset production teams. Projects are also hierarchical, which enables organizations to, for example, establish a program that can spawn many projects.

Projects are also a channel for governance practices. Oracle Enterprise Repository Compliance Templates (usually taking the form of Architecture Blueprints or Project Profiles) can be applied to projects.

The Oracle Enterprise Repository Projects page provides access to tools for creating and managing projects, as shown in Figure 1–35.

Figure 1–35 Oracle Enterprise Repository Projects Page



The Oracle Enterprise Repository Page, showing the list of Projects, Name, Department and Status, with tabs in the midsection of the Page for Overview, Compliance Templates, Consumed Assets, Produced Assets, Users, and Selected Projects.

****

- Section 1.4.1, "Viewing Project Details"
- Section 1.4.2, "Locating a Project"

1.4.1 Viewing Project Details

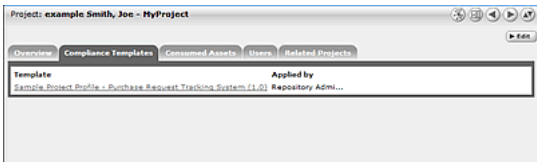
Project information can be displayed as a series of portlets or tabs.

Project information includes a project description, and indicates the assigned department, start date, estimated hours, and project status.

Compliance Templates

Displays the Compliance Template(s) assigned to the project, as shown in Figure 1–36.

Figure 1–36 Compliance Templates Tab



The Compliance Templates Tab displaying the Compliance Template or Templates assigned to the Project.

****

Consumed Assets

Lists any assets used in (or under consideration for use in) the project, as shown in Figure 1–37.

Figure 1–37 Consumed Assets Tab

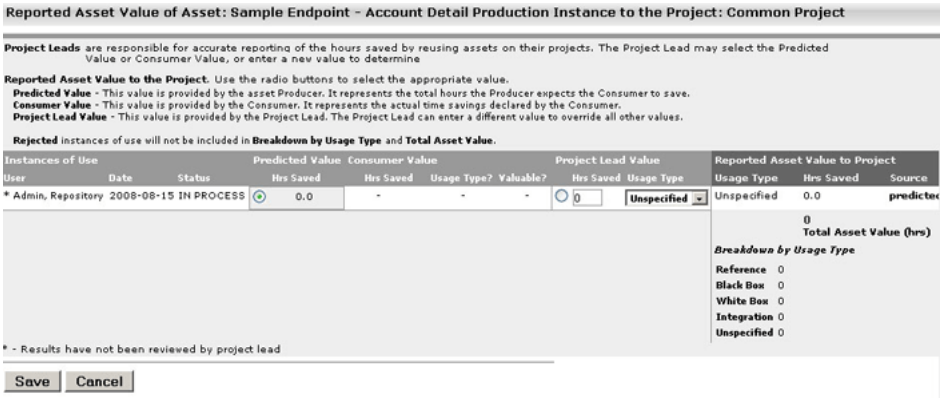
The Consumed Assets Tab, listing any assets used or under construction in the Project by rows, on the right is the Reported Asset Value in hours.

****

Click the **Zoom** icon next to any listed asset to display its Reported Asset Value.

- Project members can view the Asset Usage Detail.
- Project leaders can view/edit the Reported Asset Value.

Figure 1–38 Reported Asset Value of Asset



Clicking the Zoom next to any listed asset to display the assets Reported Asset Value/

****

- Reported Asset Value represents the hours saved by reusing assets on their projects. This value is derived from either the Predicted Value, Consumer Value, or a completely new value determined by the project leader. Only the project leader can set this value.

Produced Assets

Lists any assets produced by the project, as shown in Figure 1–39.

Figure 1–39 Produced Assets Tab

The Produced Assets tab, showing assets produced by the Project.

****

Users

Lists all users associated with the project, and each user's role on the project (leader/members), as shown in Figure 1–40. Project leaders have the ability to assign a Reported Asset Value value to assets consumed by the project.

Figure 1–40 Users Tab

The Users Tab, listing all users associated with the project, and each user's role on the project.

****

Related Projects

When enabled, lists any related projects, and defines the relationships in parent/child terms.

1.4.2 Locating a Project

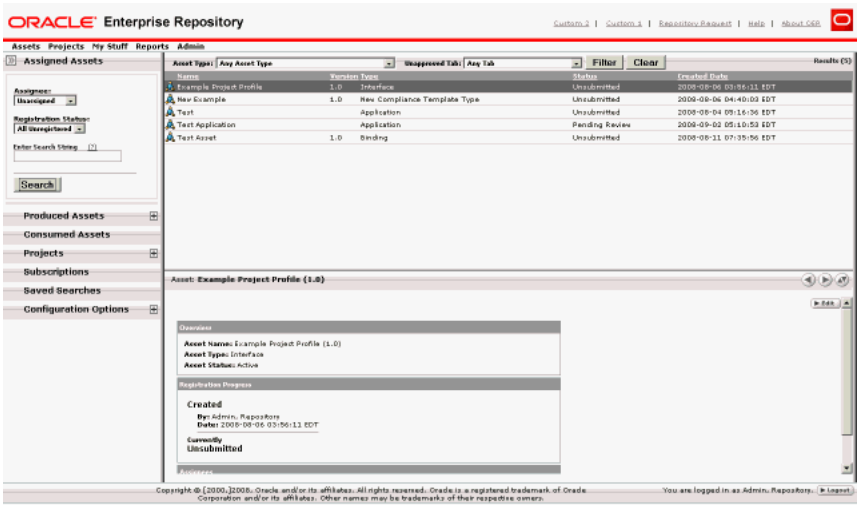
This procedure is performed on the Projects page.

1. Enter a keyword or search term in the Name text box.
2. Use the Department and Status lists as necessary to filter the search.
3. Click **Search**. The search results are listed in the main pane.

1.5 My Stuff

The My Stuff page provides each user with a personalized Oracle Enterprise Repository home page.

Figure 1–41 Oracle Enterprise Repository Assets Home Page



The My Stuff page, showing Assigned Assets, Produced Assets, Consumed Assets, Projects, Subscriptions, Saved Searches, and Configuration Options.

****

The My Stuff page includes several elements, as described below:

- Section 1.5.1, "Assigned Assets"
- Section 1.5.2, "Produced Assets"
- Section 1.5.3, "Consumed Assets"
- Section 1.5.4, "Projects"
- Section 1.5.5, "Subscriptions"
- Section 1.5.6, "Saved Searches"
- Section 1.5.7, "Configuration Options"

1.5.1 Assigned Assets

Lists all the assets assigned to a user for review and processing as part of the asset registration process. You can base your search for assets by selecting Assignee depending on the options below:

- Me - Asset that I am assigned to
- Unassigned - Asset that nobody is assigned to
- Any Assignee - Asset that somebody is assigned to

You can also base your search for assets on the registration status of an asset. The registration status that can assigned to any asset are as follows:

- Registered
- Unregistered
- All Assets

After you select the values in the Assignee and Registration Status fields, enter a search string in the Enter Search String field and click **Search**. The result of your search is displayed on right-side of the screen.

1.5.2 Produced Assets

Lists assets that you submitted.

Click any listed asset to:

- View an asset overview (the asset detail).
- Edit an asset.
- If the asset submission was rejected, then you can also resubmit the asset by clicking the Edit button.

The left pane options available are:

- **Submit an Asset**: Enables you to submit an asset.
- **Edit / Manage Assets**: Enables you to edit or manage an asset that already exists. When you click the Edit / Manage Assets option, the Asset Editor window opens, enabling you to perform the following actions:
  - Configure Acceptable Value Lists
  - Configure Artifact Stores
  - Configure Categorizations
  - Configure Relationships
  - Configure Rejection Reasons
  - Configure Vendors

  The Search My Assets pane enables you to list assets according to Registration status. For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Enterprise Repository*.

1.5.3 Consumed Assets

Lists In-Process assets with a survey status of *Still Evaluating* and the project to which the asset is associated.

Click any listed asset to:

- View an asset overview.
- Submit a review for an asset.
- Update the status of the survey.

1.5.4 Projects

Lists *Closed* and *Open* projects to which the user is assigned. The options available on the left pane are as follows:

- **Create New**: Enables the user to create a new project.
- **List All**: Lists all the projects that are currently available.

Click any listed project to view the following details:

- **Overview**: Describes the general overview of the project that covers Description, the Department it belongs to, other details such as Start Date, Estimated Hours, and Status.
- **Applied Compliance Templates**: Describes the compliance templates currently applied to the project.
- **Consumed Assets**: Describes the assets consumed or downloaded for this project.
- **Produced Assets**: Describes the assets produced for this project.
- **Users**: Describes the users assigned for this project. You can also reassign the users by clicking the Reassign Users / Usage option.
- **Related Projects**: Describes the projects that are currently related to this project.

1.5.5 Subscriptions

Lists all the asset subscriptions for user notification of updates.

Unsubscribe to an Asset

To unsubscribe to an asset, select the asset that you want to unsubscribe and then click the **Unsubscribe** button. A confirmation dialog opens, click **OK**.

Click any listed asset to view the asset details. The names and descriptions of the various tabs available are as follows:

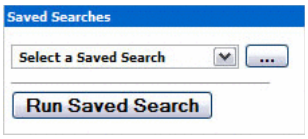
- **Apply Policies**: Enables you to apply or remove policies from this asset.
- **Notify Subscribers**: Enables you to send an email notification to the subscribers of this asset.
- **View Access**: Enables you to view the access settings for an asset.
- **Subscribe**: Enables you to subscribe to this asset to be notified of any updates.
- **Use - Download**: Enables you to use this asset and download any available files.
- **Edit**: Enables you to edit the details of this asset.

1.5.6 Saved Searches

Lists all saved searches. This option enables you to search for a particular search and save it for future use.

Click any listed search to display the search results, as shown in Figure 1–42.

Figure 1–42 Saved Searches Dialog



The Saved Search Dialog, listing Saved Searches, with a button that enables you to run any saved searches.

****

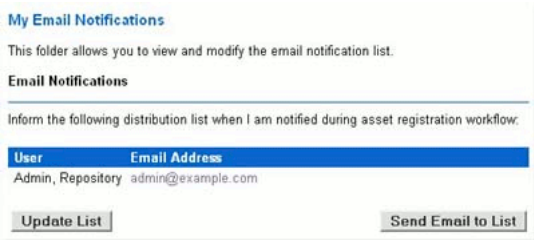
To delete a saved search, you must select the saved search, and then click the **Delete** button.

1.5.7 Configuration Options

Email Notifications

Lists all members of the distribution list.

Figure 1–43 Email Notifications



The Email Notifications Configuration Option\listing and allowing update of all members of a distribution list.

****

- Click the **Update List** button to add or remove list members.
- Click the **Send Email to List** button to open an email that is addressed to the list members.

Change Password

Enables you to change the password to Oracle Enterprise Repository. This has the following fields:

- Old Password
- New Password
- Verify Password

Enter the credentials and click the **Change Password** button.

Part II

Using JDeveloper with Oracle Enterprise Repository

This part describes configuring JDeveloper to connect to Oracle Enterprise Repository and using JDeveloper to interact with Oracle Enterprise Repository.

This part contains the following chapters:

2

Configuring Oracle JDeveloper to Support Integration with Oracle Enterprise Repository

This chapter describes how to configure Oracle JDeveloper to integrate with Oracle Enterprise Repository. First, you must download the Oracle Enterprise Repository plug-in for JDeveloper. Next, you can create a connection to an Oracle Enterprise Repository server. After installing the plug-in and creating a repository connection, see Chapter 3, "Using Oracle JDeveloper to Interact with Oracle Enterprise Repository" for the next steps.

This chapter includes the following sections:

- Install the Oracle Enterprise Repository JDeveloper 12c Plug-in
- Create a New Repository Connection
- Edit an Existing Repository Connection

2.1 Install the Oracle Enterprise Repository JDeveloper 12*c* Plug-in

To install the Oracle Enterprise Repository plug-in for Oracle JDeveloper 12*c*:

1. Install Oracle JDeveloper on your local computer. Starting in 12*c*, you must use the Oracle SOA Suite Quick Start distribution to obtain a version of JDeveloper pre-configured for Oracle SOA Suite. See "Installing Oracle SOA Suite Quick Start for Developers" in *Installing SOA Suite and Business Process Management Suite Quick Start for Developers* for more information.

   > **Note:** Ensure that you are using the Oracle SOA Suite Quick Start for Developers distribution. The Oracle BPM Suite Quick Start for Developers distribution includes a version of JDeveloper that will not work with the plug-in and is not supported for use with Oracle Enterprise Repository.

2. Acquire the file for patch 19721053 (`p19721053_121300_Generic.zip`) from My Oracle Support.
3. Ensure that JDeveloper is closed before continuing.
4. Follow the instructions in the patch's `README.txt` file.
5. Restart JDeveloper.

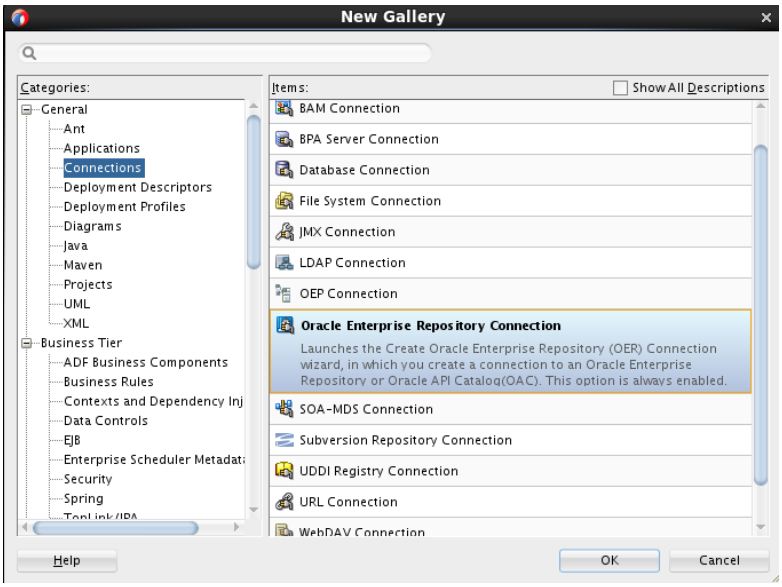
2.2 Create a New Repository Connection

After you have installed the plug-in, you can create a connection to a repository server.

To create a new repository connection:

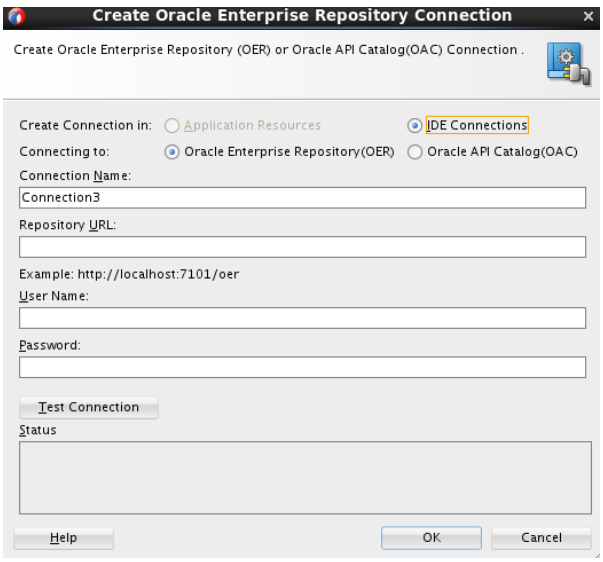
1. Click **File**, **New**. The New Gallery dialog is displayed.
2. Select **General**, **Connections**, and then select **Oracle Enterprise Repository Connection**, as shown in Figure 2–1.

Figure 2–1 New Gallery Dialog



3. Click **OK**. The Create Oracle Enterprise Repository Connection dialog is displayed, as shown in Figure 2–2.

Figure 2–2 Create Oracle Enterprise Repository Connection Dialog

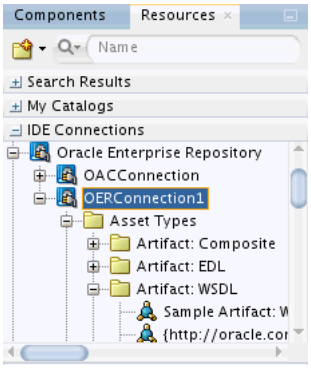


This figure illustrates the Create Oracle Enterprise Repository Connection dialog. This dialog is described in the surrounding text.

****

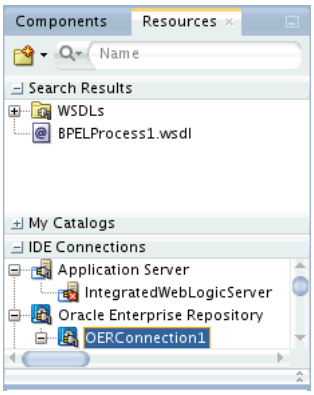
4. Enter the following information:
   - Connecting To: Select **Oracle Enterprise Repository** if you are connecting to OER, or select **Oracle API Catalog** if you are connecting to OAC.
   - Connection Name: A descriptive name for the Oracle Enterprise Repository connection.
   - Repository URL: The 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.
5. Click **Test Connection**. A success message is displayed in the Status pane.
6. Click **OK**.
7. In the Resource Palette, under IDE Connections, expand Oracle Enterprise Repository to see the application server connection that you created, as shown in Figure 2–3.

Figure 2–3 Resource Palette



8. Enter a search criteria to search for assets in the Search field. A list of all the assets is displayed, as shown in Figure 2–4.

Figure 2–4 Search Results



2.3 Edit an Existing Repository Connection

To edit an existing connection between JDeveloper and Oracle Enterprise Repository:

1. In the Resource Palette, under IDE Connections, expand Oracle Enterprise Repository to see the existing repository connections.
2. Right-click on the connection you want to edit, and then click **Properties**.
3. Edit any of the following fields:
   - 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.
4. Click **Test Connection**. A success message is displayed in the Status pane.
5. Click **OK** to save the changes and close the dialog.

3

Using Oracle JDeveloper to Interact with Oracle Enterprise Repository

This chapter describes the consumption processes and the interaction of Oracle JDeveloper with Oracle Enterprise Repository.

This chapter includes the following sections:

- Using Oracle JDeveloper

3.1 Using Oracle JDeveloper

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

You can use Oracle SOA Suite with Oracle Enterprise Repository 12*c*.

This section describes the following topics:

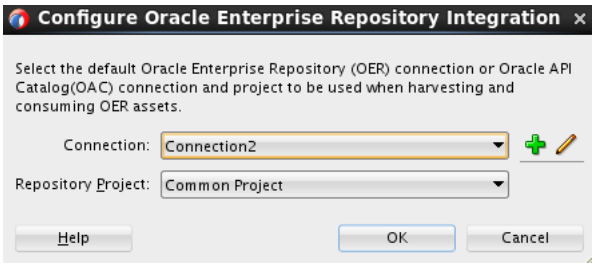
- Section 3.1.1, "Associating a JDeveloper Application with Oracle Enterprise Repository"
- Section 3.1.2, "Search Oracle Enterprise Repository"
- Section 3.1.3, "View Asset Details"
- Section 3.1.4, "Consuming WSDL/Service from Oracle Enterprise Repository"

3.1.1 Associating a JDeveloper Application with Oracle Enterprise Repository

You must associate the JDeveloper application with a default Oracle Enterprise Repository connection to consume assets from Oracle Enterprise Repository using JDeveloper. To associate a JDeveloper application with a default Oracle Enterprise Repository connection:

1. In Oracle JDeveloper, expand the **Tools** menu, and then select **Oracle Enterprise Repository**. The Configure Oracle Enterprise Repository Integration dialog is displayed.
2. Select the following options, as shown in Figure 3–1.
   - 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 3–1 Configure Oracle Enterprise Repository Integration dialog



This figure illustrates the Configure Oracle Enterprise Repository Integration dialog. The dialog displays the Connection and Repository Project lists. The New And Edit icons appear next to the Connection list; these icons allow you to create a new repository connection or edit the connection selected in the Connections list. Help, OK, and Cancel buttons are also displayed.

****

3. 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.

3.1.2 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.

3.1.3 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.

3.1.4 Consuming WSDL/Service from Oracle Enterprise Repository

You can download an asset's artifacts (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).

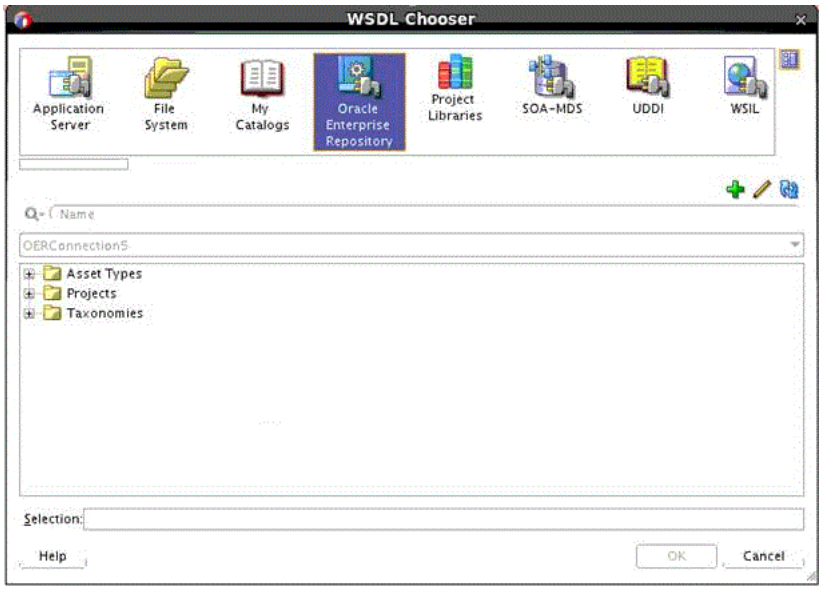
You can consume services, schemas, xslts, and events from Oracle Enterprise Repository.

Note: To track the usage of an asset, you have to first associate a JDeveloper application with Oracle Enterprise Repository project. See Section 3.1.1, "Associating a JDeveloper Application with Oracle Enterprise Repository" for more information.

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 WSDL Chooser dialog is displayed.
4. Select the Oracle Enterprise Repsoitory tab, as shown in Figure 3–2.

Figure 3–2 WSDL Chooser Dialog



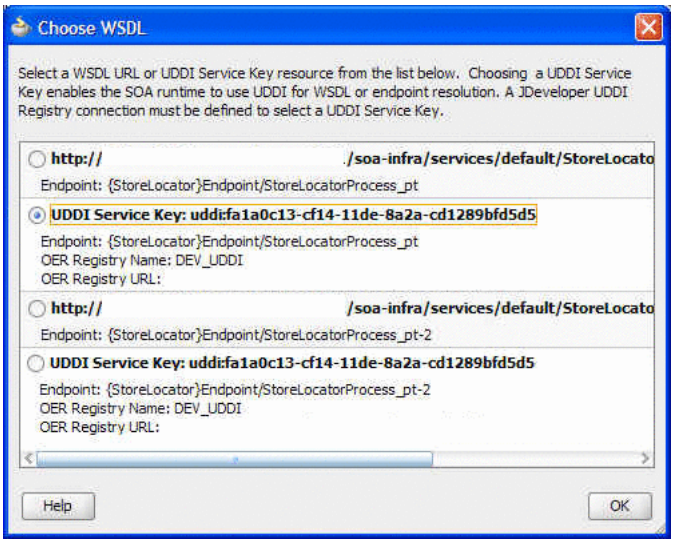
This figure illustrates the WSDL Chooser dialog. The Oracle Enterprise Repsoitory tab is selected.

****

5. Use the navigation tree to find 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 3–3.

Figure 3–3 Choose WSDL Dialog



This figure illustrates the 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 cannot select UDDI keys.

6. Click **OK**.

To consume a WADL 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 REST Service component from the Component Palette to the External References swim lane. The Create REST Binding dialog is displayed.
3. Enter the required information into the **Name**, **Type**, and **Base URI** fields.
4. Enter the necessary information into the **Resources** panel.
5. From the Operation Bindings panel, click the **Add** icon and select **Add operations based on WADL Service**. The WADL Chooser dialog is displayed.
6. Select the **Oracle Enterprise Repository** tab.
7. Use the navigation tree to find the service that you want to invoke/consume.

Part III

Developing Custom Integrations Using the REX API

This part describes developing custom integrations for Oracle Enterprise Repository using the REX API Framework.

This part contains the following chapter:

4

Using the Repository Extensibility Framework

This chapter describes the Repository Extensibility Framework (REX) architecture and discusses how to enable the OpenAPI and consume the WSDL.

This chapter includes the following sections:

4.1 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 enables 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 enables programmatic interaction with a .NET client.

> **Note:** Instances of "flashline" and "registry" appear in this documentation, particularly in the java package structure and in the REX class names.

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/

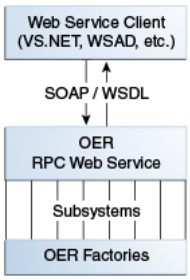
> **Note:** The examples provided in the documentation are for illustrative purposes and will not necessarily compile due to package structure differences between versions of the Oracle Enterprise Repository WSDL. You must change the package structure to appropriately target your version of Oracle Enterprise Repository.

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

4.2 REX Architecture

Figure 4–1 describes the REX architecture.

Figure 4–1 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 4.2.4, "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.

4.2.1 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 4.2.2, "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

4.2.2 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 4–1 provides a detailed list of the detailed operations that the subsystem can have apart from the CRUD-Q operations.

Table 4–1 Subsystems 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	

Table 4–1 (Cont.) Subsystems and CRUD-Q Convention Relationship

	Create	Read	Update	Delete	Query	Other Features
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	

4.2.2.1 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.

4.2.2.2 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.

4.2.3 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 can 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 its IDEs or web services toolkits at the following URL:

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

If you are creating custom integrations with OER from Oracle BPM 11g, use the following URL instead:

http://appserver/oer/services/RexAPI?wsdl

Java stubs for the Oracle Enterprise Repository REX WSDL can 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
```

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.

4.2.4 Versioning Considerations for the Oracle Enterprise Repository REX

The evolution of the Oracle Enterprise Repository REX parallels the evolution of Oracle Enterprise Repository. As a result of this process, incompatibilities may emerge between older and newer versions of REX. While full version compatibility is our goal, backward compatibility is subject to unpredictable and therefore potentially unavoidable limitations. Oracle Enterprise Repository REX includes the following backward 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 backward 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.

4.3 Basic Concepts

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

4.3.1 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.

4.3.2 Consuming 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.

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.

Example 4–1 How to Retrieve an AuthToken and use in 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 deserialized 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();
}
```

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.

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:

Example 4–2 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:

Example 4–3 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);
```

4.4 REX API Descriptions and Use Cases

This section describes and provides use cases for the REX API.

- Section 4.4.1, "ArtifactStore API"
- Section 4.4.2, "AcceptableValueLists API"
- Section 4.4.3, "Asset API"
- Section 4.4.4, "AssetType API"
- Section 4.4.5, "Categorization Types and Categorizations API"
- Section 4.4.6, "CMF Entry Type API"
- Section 4.4.7, "Custom Access Settings API"

4.4.1 ArtifactStore API

This section provides a use case for the ArtifactStore API that describes how to create a missing ArtifactStore in Oracle Enterprise Repository.

4.4.1.1 Overview

The ArtifactStore subsystem provides a web services-based mechanism that is used to query and create Oracle Enterprise Repository ArtifactStores.

4.4.1.2 Use Case: Create Missing ArtifactStore

Description

This use case describes how to create a missing artifactstore.

Sample Code

Example 4–4 Use Case: Create Missing ArtifactStore

```
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());
    }
  }
}
```

4.4.2 AcceptableValueLists API

This section provides use cases for the AcceptableValueLists API that describe how to create a new acceptable value list and enter it into Oracle Enterprise Repository and populate an asset's single or multiple selection lists with acceptable values.

4.4.2.1 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.

4.4.2.2 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

Example 4–5 Use Case: Create and Edit an Acceptable Value List

```
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();
    }
  }
}
```

4.4.2.3 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

Example 4–6 Use Case: Populate Lists with Acceptable Values

```
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();
    }
  }
}
```

4.4.3 Asset API

This section describes the Asset API and provides use cases for the Asset API that describe how to create or modify an asset, build an asset search, and work with asset status and asset tabs.

4.4.3.1 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.

  After 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 (for example, 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 from the asset's permissions. If specific File type permissions are not applied to a file, these permissions are inherited from the asset's permissions to which the files belong.

4.4.3.1.1 Definitions

This section provides 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 administrator 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.

4.4.3.1.2 Sample Code *Example 4–7 Sample Code for Custom Data Methods*

```
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();
    }
  }
}
```

4.4.3.1.3 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.

4.4.3.2 Use Cases

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

4.4.3.2.1 Use Case: Creating a New Asset Description

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

Sample Code

Example 4–8 Use Case: Creating a New Asset

```
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();
    }
  }
}
```

4.4.3.2.2 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 Example 4–9.

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

Example 4–9 Use Case: Creating a New Asset from XML

```
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();
      }
    }
}
```

4.4.3.2.3 Use Case: Modifying an Asset Description

Modify the metadata for an existing asset.

Sample Code

Example 4–10 Use Case: Modifying an Asset

```
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();
    }
  }
}
```

4.4.3.2.4 Use Case: Assign Users to an Asset Description

Multiple users can be assigned to an asset.

Sample Code

Example 4–11 Use Case: Assigning Users to an Asset

```
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;
  }
}
```

4.4.3.2.5 Use Case: Building an Asset Search Description

Finding all assets that meet certain criteria.

Sample Code

Example 4–12 Use Case: Building an Asset Search

```
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
     //accepteddate
     //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();
    }
  }
}
```

4.4.3.2.6 Use Case: Upgrading Asset Status Description

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

Sample Code

Example 4–13 Use Case: Upgrading Asset Status

```
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) {
    }
  }
}
```

4.4.3.2.7 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

Example 4–14 Use Case: Downgrading Asset Status

```
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) {
    }
  }
}
```

4.4.3.2.8 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

Example 4–15 Use Case: Applying and Removing Compliance Templates from a Project

```
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();
    }
  }
}
```

4.4.3.2.9 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

Example 4–16 Use Case: Creating a New Version of an Asset

```
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;
  }
}
```

4.4.3.2.10 Use Case: Deleting Groups of Assets Description

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

Sample Code

Example 4–17 Use Case: Deleting Unneeded Assets from the Repository

```
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

4.4.3.2.11 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

Example 4–18 Use Case: Finding Assets and Updating Common-Data

```
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();
    }
  }
}
```

4.4.3.2.12 Use Case: Reading an Asset's Tabs Description

Read the tabs of an asset.

Sample Code

Example 4–19 Use Case: Reading an Asset's Tabs

```
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();
    }
  }
}
```

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

Get a specific asset tab by tabtype.

Sample Code

Example 4–20 Use Case: Retrieving an Asset's Tab Based on Tab Type

```
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();
    }
  }
}
```

4.4.3.2.14 Use Case: Approving and Unapproving a Tab Description

Approve or unapprove an asset's tab.

Sample Code

Example 4–21 Use Case: Approving and Unapproving a Tab

```
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();
    }
  }
}
```

4.4.3.2.15 Use Case: Reading an Asset's Metadata for a Given Tab Description

Read the metadata of an asset based on the given tab.

Sample Code

Example 4–22 Use Case: Reading an Asset's Metadata for a Given Tab

```
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.AssetMetadataElement;
import com.flashline.registry.openapi.entity.AssetMetadataTableElement;
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 AssetGetMetadataByTab {
  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;

      ///////////////////////////////////
      // Login to OER
      ///////////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],pArgs[2]);

      ///////////////////////////////////
      // read an asset
      ///////////////////////////////////
      lAsset = repository.assetRead(authToken, 589);

      ///////////////////////////////////
      // Get the metadata elements based
      // on the asset ID and the tab name
      ///////////////////////////////////
      AssetMetadataElement[] lElements =
repository.assetReadTabMetadata(authToken, lAsset.getID(), "Overview");

      //An AssetMetadataElement represents a custom or mandatory data element of
an asset
      for (AssetMetadataElement lElement : lElements) {
        //This represents a TABLE element type
        if (lElement.getValue() instanceof AssetMetadataTableElement) {
          AssetMetadataTableElement lTable =
(AssetMetadataTableElement)lElement.getValue();
          System.out.println(lElement.getDisplayName());
          //A TABLE can have multiple elements
          for (AssetMetadataElement lTableElement : lTable.getElements()) {
            //An element of a TABLE can be another TABLE
            if (lTableElement.getValue() instanceofAssetMetadataTableElement) {

System.out.println(((AssetMetadataTableElement)lTableElement.getValue()).getDispla
yName());
              for (AssetMetadataElement lChildElement :
((AssetMetadataTableElement)lTableElement.getValue()).getElements()) {
                System.out.println(lChildElement.getDisplayName() + " : " +
```

```
lChildElement.getValue());
              }
            } else {
              //Or an element of a TABLE can be a regular value
              System.out.println(lTableElement.getDisplayName() + " : " +
lTableElement.getValue());
            }
          }
          //This represents a MULTIPLE ITEM LIST
        }  else if (lElement.getValue() instanceof String[]) {
          System.out.println(lElement.getDisplayName());
          for (String lString : (String[])lElement.getValue()) {
            System.out.println(lString);
          }
          //This represents a SINGLE ITEM
        } else {
          System.out.println(lElement.getDisplayName() + " : " +
lElement.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();
    }
  }
}
```

4.4.4 AssetType API

This section provides use cases for the AssetType API that describe how to add a new type to the repository, create a new compliance template type, locate a type, retrieve the list of tabs available for an asset type, and retrieve all asset type tabs within the Oracle Enterprise Repository.

4.4.4.1 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, 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 Administrator's 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 entityExpansionLimit. For example, passing the following parameter to the JVM increases the entity expansion limit to 5 MB:

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

4.4.4.2 Use Cases

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

4.4.4.2.1 Use Case: Create and Edit a New Type Description

Adding a new Type to the repository.

Sample Code

Example 4–23 Use Case: Create and Edit a New Type

```
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();
    }
  }
}
```

4.4.4.2.2 Use Case: Create a Compliance Template Type Description

Adding a new Compliance Template Type to the repository.

Sample Code

Example 4–24 Use Case: Create a Compliance Template Type

```
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();
    }
  }
}
```

4.4.4.2.3 Use Case: Find Types Description

Locating a Type in the repository.

Sample Code

Example 4–25 Use Case: Locate a Type

```
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

4.4.4.2.4 Use Case: Retrieve Tabs for Asset Type Description

Retrieve the list of tabs available for an asset type.

Sample Code

Example 4–26 Use Case: Retrieve a List of Tabs for an Asset Type

```
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();
```

```
    }
  }
}
```

4.4.4.2.5 Use Case: Retrieve all Asset Type Tabs Description

Retrieves all asset type tabs within the Oracle Enterprise Repository.

Sample Code

Example 4–27 Use Case: Retrieve All Asset Type Tabs

```
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();
}
```

4.4.5 Categorization Types and Categorizations API

This section provides use cases for the Categorization Types and Categorizations API that describe how to create a categorization type, manipulate categorizations, and manipulate categorization types.

4.4.5.1 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 enables 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 enables 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.

4.4.5.2 Use Cases

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

- Section 4.4.5.2.1, "Use Case: Create a Categorization Type"
- Section 4.4.5.2.2, "Use Case: Manipulate Categorization Types"
- Section 4.4.5.2.3, "Use Case: Manipulate Categorizations"

4.4.5.2.1 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

Example 4–28 Use Case: Create a Categorization Type

```
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();
    }
  }
}
```

4.4.5.2.2 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

- Exercise 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

Example 4–29 Use Case: Manipulate Categorization Types

```
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

4.4.5.2.3 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

Example 4–30 Use Case: Manipulate Categorizations

```
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`

4.4.6 CMF Entry Type API

This section provides a use case for the CMF Entry Type API and describes how to add a new CMF entry type to Oracle Enterprise Repository and assign an existing CMF entry type to an asset.

4.4.6.1 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;
```

4.4.6.2 Use Case: Manipulating CMF Entry Types

This section describes the use case using the CMF Entry Type API.

Description

- Adding a new CMF Entry Type to Oracle Enterprise Repository.
- Assigning an existing CMF Entry Type to an asset.

Sample Code

Example 4–31 Use Case: Manipulate CMF Entry Types

```
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();
    }
  }
}
```

4.4.7 Custom Access Settings API

This section provides use cases for the Custom Access Settings API and describes how to retrieve the list of custom access settings types and retrieve the list of default custom access settings of a particular type in the Oracle Enterprise Repository.

4.4.7.1 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 4–32 describes a sample Custom Access Settings code.

Example 4–32 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();
    }
  }
}
```

4.4.7.2 Use Case: Retrieve a List of Custom Access Settings Types

Description

This method is used to retrieve the list of Custom Access Settings Types as available in Oracle Enterprise Repository.

Sample Code

```
1. String[] lCustomAccessSettingNames = null;
2. lCustomAccessSettingNames = mFlashlineRegistry.customAccessSettingNamesGet
    (mAuthToken, "asset");
```

Annotations

Line 2 - Retrieve an array of Custom Access Setting Names of type "asset".

4.4.7.3 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

```
1. String[] lCustomAccessSettingNames = null;
2. lCustomAccessSettingNames = mFlashlineRegistry.customAccessSettingDefault
     NamesGet(mAuthToken, "asset");
```

Annotations

Line 2 - Retrieve an array of default Custom Access Setting Names of type "asset".

4.4.8 Department API

This section provides a use case for the Department API that describes how to create, update, query, and delete departments in Oracle Enterprise Repository.

4.4.8.1 Overview

Departments can be created, read, queried for, and modified. These operations are described below. Bear in mind that after 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;
```

4.4.8.2 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

Example 4–33 Use Case: Manipulate Departments

```
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();
      }
    }
}
```

4.4.9 Extraction API

This section provides use cases for the Extraction API that describe how to extract an asset, read an extraction, and update an extraction in Oracle Enterprise Repository.

4.4.9.1 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. 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 Subsystems

- AssetSubsystem
- ProjectSubsystem
- CategorizationTypeSubsystem
- SurveySubsystem

4.4.9.2 Use Cases

This section describes the use cases using the Extraction API. It includes the following topics:

- Section 4.4.9.2.1, "Use Case: Extract an Asset"
- Section 4.4.9.2.2, "Use Case: Read an Extraction"
- Section 4.4.9.2.3, "Use Case: Update an Extraction"

4.4.9.2.1 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

Example 4–34 Use Case: Extract an Asset

```
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, for example, fileInfo.getDownloadURI().
2. Using the OpenAPI method repositoryFileTranslator passing the FileInfo object, for example, flashlineRegistry.repositoryFileTranslator(authToken, fileInfo).

> **Note:** DO NOT use the URI property on the FileInfo object which represents an Oracle Enterprise Repository specific path.

4.4.9.2.2 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

Example 4–35 Use Case: Read an Extraction

```
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();
    }
  }
}
```

4.4.9.2.3 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

Example 4–36 Use Case: Update an Extraction

```
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.

4.4.10 Localization of REX Clients

This section provides use cases for the localization of REX clients that describe how to create localized messages from REX Exceptions and REX Audit Messages in Oracle Enterprise Repository.

4.4.10.1 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].
```

4.4.10.2 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

Sample Code

Example 4–37 Use Case: Creating Localized Messages from REX Exceptions

```
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);
    }
}
```

4.4.10.3 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 messages appropriate for the client locale
- Get the string associated with the error code and replace the token with the message arguments

Sample Code

Example 4–38 Use Case: Creating Localized Messages from REX Audit Messages

```
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();
  }
}
```

4.4.11 Notification API

This section provides a use case for the Notification API that describes how to create a new read notification substitution list and how to create an Oracle Enterprise Repository notification.

4.4.11.1 Overview

The Notification subsystem provides a web services-based mechanism that is used to create Oracle Enterprise Repository notifications.

4.4.11.2 Use Case: Create a Read Notification Substitution List and Create a Notification

Description

To create a read notification substitution list and create an Oracle Enterprise Repository notification.

Sample Code

Example 4–39 Use Case: Create a Read Notification Substitution List and Create a Notification

```
    String[] lSubstitutions = null;
    String[] lRecipients = null;
    String lTemplateType = "asset_registered";
    NameValue[] lNameValues = null;
```

```
    lRecipients = new String[] {"recipient@xyz.com"};

    try {
      //////////////////////////
      // read the existing substitutions based on the given template
      lSubstitutions =
mFlashlineRegistry.notificationSubstitutionsRead(mAuthToken, lTemplateType);

      //////////////////////////
      // create an array of namevalue pairs; a namevalue pair for each required
substitution
      lNameValues = new NameValue[lSubstitutions.length];

      // populate the namevalues
      for(int i=0; i<lSubstitutions.length; i++) {
        lNameValues[i] = new NameValue();
        lNameValues[i].setName(lSubstitutions[i]);
        lNameValues[i].setValue("valueof-"+lSubstitutions[i]);
      }

      ////////////////////////////////////////
      // create the notification
      mFlashlineRegistry.notificationCreate(mAuthToken, lTemplateType,
lRecipients, lNameValues);

    } catch(Exception e) {
      fail(e.getMessage());
    }
```

4.4.12 Policy API

This section provides use cases for the Policy API that describe how to create a new policy, get all policies, get or set policy assertions, obtain policies applied to an asset, and determine an asset's compliance against all applied policies or specific policies.

4.4.12.1 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()`

4.4.12.2 Use Cases

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

4.4.12.2.1 Use Case: Create a Policy Description

To create a new policy, create a new asset based on the Policy Type (102).

Sample Code

Example 4–40 Use Case: Create a Policy

```
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);
  }
}
```

4.4.12.2.2 Use Case: Get All Policies Description

To get all policies, find all assets whose asset type's archetype is a policy archetype (102).

Sample Code

Example 4–41 Use Case: Get All Policies

```
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();
    }
  }
}
```

4.4.12.2.3 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

Example 4–42 Use Case: Get/Set Policy Assertions

```
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;
  }
}
```

4.4.12.2.4 Use Case: Get Policies That Have Been Applied To An Asset Description

Call `assetReadAppliedPolicies` to obtain policies applied to an asset.

Sample Code

Example 4–43 Use Case: Get Policies That Have Been Applied to an Asset

```
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();
    }
  }
}
```

4.4.12.2.5 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

Example 4–44 Use Case: Update Policies Applied to an Asset

```
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();
    }
  }
}
```

4.4.12.2.6 Use Case: Evaluate Asset Compliance Description

Use `assetEvaluateAgainstPolicy` to determine an asset's compliance with a specified policy. Use `assetEvaluateAgainstAllPolicies` to determine an asset's compliance against all applied policies.

Sample Code

Example 4–45 Use Case: Evaluate Asset Compliance

```
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();
    }
  }
}
```

4.4.13 Projects API

This section provides use cases for the Projects API that describe how to create, read, update, query, and validate projects, work with project assets, and make changes to project users in oracle Enterprise Repository.

4.4.13.1 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;
```

4.4.13.2 Use Cases

This section describes the use cases using the Projects API. It includes the following topics:

- Section 4.4.13.2.1, "Use Case: Create a New Project"
- Section 4.4.13.2.2, "Use Case: Read a Project"

4.4.13.2.1 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

Example 4–46 Use Case: Create a New Project

```
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());
    }
  }
}
```

4.4.13.2.2 Use Case: Read a Project Description

Searches for a project and reads its extractions, produced assets, users, and related projects.

Sample Code

Example 4–47 Use Case: Read a Project

```
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();
    }
  }
}
```

4.4.13.2.3 Use Case: Validate a Project Description

Validating a project enables the user to catch any validation errors before a project save is attempted.

Sample Code

Example 4–48 Use Case: Validate a Project

```
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();
    }
  }
}
```

4.4.13.2.4 Use Case: Update a Project Description

Update the information and data associated with a specific project.

Sample Code

Example 4–49 Use Case: Update a Project

```
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;
  }
}
```

4.4.13.2.5 Use Case: Update a Project's Produced Assets Description

Enables the user to perform a single database transaction to set the produced assets of a project.

Sample Code

Example 4–50 Use Case: Update a Project's Produced Assets

```
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;
  }
}
```

4.4.13.2.6 Use Case: Remove Produced Assets from a Project Description

Remove produced assets from a project.

Sample Code

Example 4–51 Use Case: Remove Produced Assets from a Project

```
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();
    }
  }
}
```

4.4.13.2.7 Use Case: Update a Project's Asset Usage Description

Enables the user to reject extractions that are associated with a project.

Sample Code

Example 4–52 Use Case: Update a Project's Asset Usage

```
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();
    }
  }
}
```

4.4.13.2.8 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. After 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

Example 4–53 Use Case: Closing a Project with Hidden Assets

```
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();
    }
  }
}
```

4.4.13.2.9 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

Example 4–54 Use Case: Add Users and Related Projects to a Project

```
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

Example 4–55 Use Case: Add and Remove Users for a Project

```
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;
  }
}
```

4.4.13.2.10 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

Example 4–56 Use Case: Remove Related Projects and Users from a Project

```
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

Example 4–57 Use Case: Keep Specific Users and Projects in a Project

```
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();
    }
  }
}
```

4.4.13.2.11 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

Example 4–58 Use Case: Update a Project's Extractions

```
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);
  }
}
```

4.4.13.2.12 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

Example 4–59 Use Case: Reassign Project User and User's Extraction to Another Project

```
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);
  }
}
```

4.4.13.2.13 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

Example 4–60 Use Case: Reassign User Only to Another Project

```
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);
  }
}
```

4.4.13.2.14 Use Case: Read the Value-Provided for a Project and Asset Description

Reads the value-provided detail for a project and asset.

Sample Code

Example 4–61 Use Case: Read the Value-Provided for a Project and Asset

```
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;
  }
}
```

4.4.13.2.15 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

Example 4–62 Use Case: Update the Value Provided for a Project and Asset - Use Predicted Value

```
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);
  }
}
```

4.4.13.2.16 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

Example 4–63 Use Case: Update the Value Provided for a Project and Asset - Use Consumer Value

```
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);
  }
}
```

4.4.13.2.17 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

Example 4–64 Use Case: Update the Value Provided for a Project and Asset - Use Project Lead Value

```
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);
  }
}
```

4.4.14 Relationship Types API

This section provides use cases for the Relationship Types API that describe how to create a new relationship type and modify or query related assets in Oracle Enterprise Repository.

4.4.14.1 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.

4.4.14.2 Use Cases

This section describes the use cases using the Relationship Types API. It includes the following topics:

- Section 4.4.14.2.1, "Use Case: Create a New Relationship Type"
- Section 4.4.14.2.2, "Use Case: Modify Related Assets"
- Section 4.4.14.2.3, "Use Case: Query Related Assets"

4.4.14.2.1 Use Case: Create a New Relationship Type Description

Creating a new type of relationship to be used between assets.

Sample Code

Example 4–65 Use Case: Create a New Relationship Type

```
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();
                  }
                }
}
```

4.4.14.2.2 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

Example 4–66 Use Case: Modify Related Assets

```
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()

4.4.14.2.3 Use Case: Query Related Assets Description

Querying for related asset types.

Sample Code

Example 4–67 Use Case: Query Related Assets

```
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();
}
```

4.4.15 Role API

This section provides a use case for the Role API that describes how to create, read, update, delete, or query a role in Oracle Enterprise Repository.

4.4.15.1 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 Section 4.4.18, "User API".

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.Role;
```

```
import com.flashline.registry.openapi.query.RoleCriteria;
```

4.4.15.2 Use Case: Manipulate Roles

Description

- Create a new role
- Read a role
- Update a role
- Delete a role
- Query for roles

Sample Code

Example 4–68 Use Case: Manipulate Roles

```
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();
    }
  }
}
```

4.4.16 Subscriptions API

This section provides use cases for the Subscriptions API that describe how to create, read, or delete subscriptions to assets and how to read users subscribed to an asset in Oracle Enterprise Repository.

4.4.16.1 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. After 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 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.

4.4.16.1.1 Use Cases

This section describes the use cases using the Subscriptions API. It includes the following topics:

- Section 4.4.16.1.2, "Use Case: Create Subscription to Assets"
- Section 4.4.16.1.3, "Use Case: Delete Subscription to Assets"
- Section 4.4.16.1.4, "Use Case: Read Subscriptions for Assets"
- Section 4.4.16.1.5, "Use Case: Read Users Subscribed to an Asset"

4.4.16.1.2 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

Example 4–69 Use Case: Create a Subscription to Assets

```
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();
    }
  }
}
```

4.4.16.1.3 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

Example 4–70 Use Case: Delete a Subscription to Assets

```
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();
    }
  }
}
```

4.4.16.1.4 Use Case: Read Subscriptions for Assets Description

- Authenticate with REX.
- Read the list of subscribed assets for the authenticated user.

Sample Code

Example 4–71 Use Case: Read Subscriptions for Assets

```
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();
    }
  }
}
```

4.4.16.1.5 Use Case: Read Users Subscribed to an Asset Description

- Authenticate with REX.
- Read the list of users subscribed to a particular asset.

Sample Code

Example 4–72 Use Case: Read Users Subscribed to an Asset

```
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();
    }
  }
}
```

4.4.17 System Settings API

This section provides a use case for the System Settings API that describes how to query for system settings in Oracle Enterprise Repository.

4.4.17.1 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.

4.4.17.2 Use Case: Query for System Settings

Description

Query for system settings in REX.

Sample Code

Example 4–73 Use Case: Query for System Settings

```
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();
    }
  }
}
```

4.4.18 User API

This section provides a use case for the User API that describes how to create, retrieve, update, and deactivate users or query for users.

4.4.18.1 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 Section 4.4.15, "Role API".

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.RegistryUser;
import com.flashline.registry.openapi.query.UserCriteria;
```

4.4.18.2 Use Case: Manipulating Users

Description

- Create a new user.
- Retrieve an existing user.
- Update a user.
- Deactivate a user.
- Query for users.

Sample Code

Example 4–74 Use Case: Manipulate User

```
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();
    }
  }
}
```

4.4.19 Vendor API

This section provides a use case for the Vendor API that describes how to add or assign vendors in Oracle Enterprise Repository.

4.4.19.1 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 (for example, 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;
```

4.4.19.2 Use Case: Manipulating Vendors

Description

- Adding a new Vendor to Oracle Enterprise Repository.
- Assigning an existing Vendor to an asset.

Sample Code

Example 4–75 Use Case: Add or Assign Vendors

```
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();
        }
      }
    }
```