

Oracle Enterprise Repository

Asset API

Overview

Description

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 will be generated by REX.

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

Once a summary query has been performed, the full asset objects can be retrieved for assets of interest using

either `assetRead`, or `assetReadArrayFromSummary`. If multiple assets are desired, use of the `assetReadArrayFromSummary` method is recommended. See the API documentation for details on using this method.

• Performance

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

▪ Iterative Reads

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

▪ Listing Operations

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

• Access Control

- Which assets a user of REX can see, and to some extent the information in those assets, is controlled by access settings. The same access restrictions that exist for a user accessing the system via the web gui also apply to the REX asset subsystem.
- Query Restrictions - users can only retrieve assets in a call to `assetQuery` or `assetRead` for which they have **view** permission.
- Update Restrictions - users can only update assets for which they have **edit** permission.
- File restrictions - users can only view the files for which they have download permissions as set in the File type Custom Access Settings applied to each individual file. This means that a user might be able to view an asset, but might not be able to view any of the asset's files. Each file can have it's own permissions, different than the asset's permissions. If specific File type permissions are not applied to a file, these permissions will be inherited from the asset's permissions to which the files belong.

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 hexadecimal 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 will fail.

• Name and Version

- String fields that combine to uniquely identify an asset.

• CustomData

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

Sample Code:

```
package com.flashline.sample.assetapi;

import java.io.StringReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.text.Format;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import javax.xml.rpc.ServiceException;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;
import org.jdom.xpath.XPath;

import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

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

        try {

            ////////////////////////////////////////////////////
            // Connect to Oracle Enterprise Repository
            ////////////////////////////////////////////////////
            URL IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ////////////////////////////////////////////////////
            // 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-yyyyymmdd-",
"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"
XPath = null;
Elms = null;
//First read the Node "documentation" of the specific asset
IXMLDocumentation = repository.assetReadCustomDataNode(authToken, 589, "documentation");

//Using DOM, convert the XML to a Document
IDoc = null;
IBuilder = new SAXBuilder();
IReader = null;
IReader = new StringReader(IXMLDocumentation);
IBuilder.setValidation(false);
IBuilder.setIgnoringElementContentWhitespace(true);

IDoc = IBuilder.build(IReader);

XPath = XPath.newInstance("documentation/document");
Elms = XPath.selectNodes(IDoc);

//Cycle through the "document" elements until we find the one we want. Then update it.
for (int i=0;i<Elms.size();i++) {
    Element lElm = (Element)Elms.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(IDoc);

//Get the next element
IXMLDocumentation = repository.assetReadCustomDataNode(authToken, 589, "also-known-as");
IDoc = null;
IBuilder = new SAXBuilder();
IReader = null;
IReader = new StringReader(IXMLDocumentation);
IBuilder.setValidation(false);
IBuilder.setIgnoringElementContentWhitespace(true);

IDoc = IBuilder.build(IReader);

XPath = XPath.newInstance("also-known-as");
Elms = XPath.selectNodes(IDoc);

//Get the also-known-as element
for (int i=0;i<Elms.size();i++) {
    Element lElm = (Element)Elms.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();
}
}
}

```

Related subsystems

- **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.
- **CategorizationTypes**
 - 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

Use Cases

Use Case: Creating a new asset

Sample Code:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            // 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 be 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();
}
}
}
}

```

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 can be used to validate the asset XML before creation. The schema for an asset type is available through the Open API as can be seen in the example below.

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

Sample Code:

```

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 ISaxParserFactory = null;
SAXParser ISaxParser = null;
```

```
try {

    //////////////////////////////////////
    // Connect to Oracle Enterprise Repository
    //////////////////////////////////////
    URL IURL = null;
    IURL = new URL(pArgs[0]);
    FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
        .getFlashlineRegistry(IURL);

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

    // Anonymous class to handle validation errors
    DefaultHandler IDefaultHandler = 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
////////////////////////////////////
ISaxParserFactory = SAXParserFactory.newInstance();
ISaxParserFactory.setNamespaceAware(true);
ISaxParserFactory.setValidating(true);
ISaxParser = ISaxParserFactory.newSAXParser();
ISaxParser.setProperty(SCHEMA_LANGUAGE, XML_SCHEMA);
ISaxParser.setProperty(SCHEMA_SOURCE, new InputSource(new StringReader(
    schema)));
ISaxParser.parse(new InputSource(new StringReader(assetXML)),
    IDefaultHandler);

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

```

Use Case: Modifying an asset

Description

Modify the metadata for an existing asset.

Sample Code:

```

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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ////////////////////////////////////////////////////
            // 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();
}
}
}

```

Use Case: Assign users to an asset

Description

Multiple users can be assigned to an asset.

Sample Code:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            //////////////////////////////////////////////////
            // 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[] IUsers = 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
// will be used.)
////////////////////////////////////

////////////////////////////////////
// Add AssignedUser objects to the array
////////////////////////////////////
AssignedUser IUser = new AssignedUser();
IUser.setUserID(99); // 99 is the admin user id
IUsers[0] = IUser;

IUser = new AssignedUser();
RegistryUser IRegistryUser1 = createRegistryUser(repository, authToken);
IUser.setUserID(IRegistryUser1.getID());
IUsers[1] = IUser;

IUser = new AssignedUser();
RegistryUser IRegistryUser2 = createRegistryUser(repository, authToken);
IUser.setUserID(IRegistryUser2.getID());
IUsers[2] = IUser;

////////////////////////////////////
// Add array to the asset that is being updated
////////////////////////////////////
myAsset.setAssignedUsers(IUsers);

////////////////////////////////////
// save the modifications
////////////////////////////////////
repository.assetUpdate(authToken, myAsset);

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

protected static RegistryUser createRegistryUser(FlashlineRegistry repository, AuthToken authToken) throws OpenAPIException,
RemoteException {
    RegistryUser IRet = null;
    String IUserName = "user"+Calendar.getInstance().getTimeInMillis();
    IRet = repository.createUser(authToken, IUserName, "", IUserName, IUserName+"@example.com", IUserName, false, false, false);
    return IRet;
}
}

```

Use Case: Building an asset search

Description

Finding all assets that meet certain criteria.

Sample Code:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            //////////////////////////////////////
            // 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 via the AssetCriteria Object
// -----
// If no operator is specified when implementing a search
// using the setSearchTerms method in the AssetCriteria
// object, the system will default 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
ITerm.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.
ITerm.setDateField("registereddate");
ITerm.setDateFormat("yyyy-MM-yy");
ITerm.setBeginDate("2007-01-01");
ITerm.setBeginOperator("gte");

ITerm.setEndDate("2007-01-10");
ITerm.setEndOperator("lte");

//The following SearchTerm translates to "Assets where the registereddate is less than or equal to Jan. 10, 2007

criteria.setSearchTerms(new SearchTerm[] {ITerm});

//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 ITabTerm = new TabStatusSearchTerm();

//tabstatus is the type of search we want to do. overview is the name of the tab we want to search on
ITabTerm.setKey("tabstatus");
ITabTerm.setTabNames(new String[] { "overview" });
ITabTerm.setApproved(true);

criteria.setSearchTerms(new SearchTerm[] {ITabTerm});

//This query returns all assets with the Overview tab being approved
assets = repository.assetQuerySummary(authToken, criteria);

//You may also search by a date range.
ITabTerm.setKey("tabstatus");
ITabTerm.setTabNames(new String[] { "overview" });
ITabTerm.setApproved(false);
ITabTerm.setBeginDate("2007-1-01");
ITabTerm.setBeginOperator("lte");

criteria.setSearchTerms(new SearchTerm[] {ITabTerm});

//The following will return 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 will return 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();
}
}
}
}

```

Use Case: Upgrading asset status

Description

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

Sample Code:

```

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) {
}
}
}
}

```

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:

```

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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ////////////////////////////////////////////////////
            // 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 (!KeyValuePair.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) {
    }
}
}

```

Use Case: Apply and remove Compliance Templates from a project

Description

Compliance Templates can be added and removed from multiple projects.

NOTE: An OpenAPIException will occur if an asset is applied to a project and that asset is NOT a Compliance Template.

Sample Code:

```

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 IURL = null;
            IURL = new URL(pArgs[0]);

            ////////////////////////////////////////////////////
            // Connect to Oracle Enterprise Repository
            ////////////////////////////////////////////////////
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ////////////////////////////////////////////////////
            // 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 {

```

```

    IComplianceTemplateAsset = repository.assetCreate(authToken, "My Compliance Template",
        ""+Calendar.getInstance().getTimeInMillis(), ctType.getID());
}

////////////////////////////////////
// Create a String array of Project IDs that the Compliance
// Template will be applied to.
////////////////////////////////////
String[] IProjectIDs = { "50000" };

// //////////////////////////////////////
// Apply template to the projects.
// //////////////////////////////////////
repository.assetApplyToProjects(authToken, IProjectIDs,
    IComplianceTemplateAsset);

////////////////////////////////////
// Retrieve an array of Projects that this template is
// applied to.
////////////////////////////////////
Project[] IProjects = repository.assetReadAppliedToProjects(
    authToken, IComplianceTemplateAsset);
String lMsg = "Compliance Template " + IComplianceTemplateAsset.getName();
lMsg += " applied to Project(s): ";
for (int i=0; IProjects!=null && i<IProjects.length; i++) {
    lMsg += ""+IProjects[i].getName()+(i+1==IProjects.length ? "." : ", ");
}
System.out.println(lMsg);

////////////////////////////////////
// Create a String array of Project IDs that the Compliance
// Template will be removed from.
////////////////////////////////////
String[] IRemoveProjectIDs = { "50000" };

// //////////////////////////////////////
// Remove template from the projects.
// //////////////////////////////////////
repository.assetRemoveAppliedToProjects(authToken,
    IRemoveProjectIDs, IComplianceTemplateAsset);

} 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();
}
}
}

```

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:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ////////////////////////////////////////////////////
            // 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;
}
}

```

Use Case: "Housekeeping"

Description

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

Sample Code:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ////////////////////////////////////////////////////
            // 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
- etAcceptedByName
- setAcceptedByDate
- setActiveStatus
- setAssigned
- setAssignedToID
- setAssignedToName
- 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
- setTypeNames
- 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**

- The customizable metadata for an asset is contained in XML stored in the customData String on the asset.

- **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 will validate that:
 - CustomData is well formed XML
 - CustomData contains XML that is valid based on the asset type

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:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            ///////////////////////////////////////////////////
            // 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 IEx) {
            System.out.println("ServerCode = " + IEx.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();
    }
}
}

```

Use Case: Reading an Asset's Tabs

Description

Read the tabs of an asset.

Sample Code:

```

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
//////////
ITabBeans = 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();
}
}
}
}

```

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

Description

Get a specific asset tab by tabtype.

Sample Code:

```

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();  
}  
}  
}
```

Use Case: Approving and Unapproving a tab

Description

Approve or unapprove an asset's tab.

Sample Code:

```
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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator().getFlashlineRegistry(IURL);

            //////////////////////////////////////
            // 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();
        }
    }
}

```