

Oracle Enterprise Repository

Projects API

Overview

Description

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;
```

Use Cases

- [Create a New Project](#)
- [Read a Project](#)
- [Validate a Project](#)
- [Update a Project](#)
- [Update a Project's Produced Assets](#)
- [Remove Produced Assets From a Project](#)
- [Update a Project's Asset Usage](#)
- [Close a Project With Hidden Assets](#)

- **Add Users and Related Projects to a Project**
- **Remove Related Projects and Users from a Project**
- **Update a Project's Extractions - Reassign Extractions to a Different User**
- **Update a Project's User - Reassign User and Extractions**
- **Update a Project's User - Reassign User Only**
- **Read the Value-Provided for a Project and Asset**
- **Update the Value-Provided for a Project and Asset - Predicted Value**
- **Update the Value-Provided for a Project and Asset - Consumer Value**
- **Update the Value-Provided for a Project and Asset - Project Lead Value**

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:

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

```

        .getFlashlineRegistry(IURL);

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

Project IProject = null;
ProjectEntities IProjectEntities = null;
String[] ILeaderIds = null;

// -----

IProject = new Project();
IProjectEntities = new ProjectEntities();

// -----
// set the name of project
IProject.setName("NEW_PROJECT_NAME");

// -----
// set the name of the project's department
IProject.setDepartmentID(50000); // a department with id 50000 must
                                // already exist

// -----
// set the userids of the project leaders
ILeaderIds = new String[] { "99" };
IProjectEntities.setLeaderIDs(ILeaderIds);

// -----

repository.projectCreate(authToken, IProject, IProjectEntities);

} 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());
}

}

}

```

Use Case: Read a project

Description

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

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

            ////////////////////////////////////
        }
    }
}
```

```

// 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[] IExtractions = repository.projectReadExtractions(
            authToken, projectRead);
        Asset[] IAssets = 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 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: Validate a project

Description

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

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

            ////////////////////////////////////
        }
    }
}
```

```

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

Project IProject = new Project();
Results IResults = new Results();

String[] ILeaders = { "100" };
Department department = repository.departmentRead(authToken, "Department Name");

ProjectEntities IProjectEntities = new ProjectEntities();

// -----
// set the project data
IProjectEntities.setLeaderIDs(ILeaders);
IProject.setName("Project Name");
IProject.setDepartmentName("DEPARTMENT_NAME");

// -----
// Validate a project
IResults = repository.projectValidate(authToken, IProject, IProjectEntities);

KeyValuePair[] IPairs = IResults.getErrors();

for (int i = 0; i < IPairs.length; i++) {
    KeyValuePair IPair = IPairs[i];
    System.out.println(IPair.getValue());
}
} 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: Update a project

Description

Update the information and data associated with a specific project.

Sample Code:

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

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

            Project IProject = new Project();
            Department department = new Department();
            ProjectEntities IProjectEntities = 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;
}
}
```

Use Case: Update a project's produced assets

Description

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

Sample Code:

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

```

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

ProjectEntities IProjectEntities = new ProjectEntities();

Asset ISampleAsset1 = createAsset(repository, authToken);
Asset ISampleAsset2 = createAsset(repository, authToken);

String[] assetIds = { ""+ISampleAsset1.getID(), ""+ISampleAsset2.getID() };
try {

    // -----
    // read an existing project
    Project projectRead = repository.projectRead(authToken, 50000);

    // -----
    // set the produced asset ids
    IProjectEntities.setAssetIDs(assetIds);

    // -----
    // update the project
    repository.projectUpdate(authToken, projectRead,
        IProjectEntities);
} catch (APIValidationException ex) {
    ex.printStackTrace();
}
} 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 Asset createAsset(FlashlineRegistry repository, AuthToken authToken)
    throws OpenAPIException, RemoteException {
    Asset myAsset = repository.assetCreate(authToken,
        "My Produced Asset", ""+Calendar.getInstance().getTimeInMillis(), 144);
    return myAsset;
}
}

```

Use Case: Remove produced assets from a project

Description

Remove produced assets from a 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.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 IURL = null;
            IURL = new URL(pArgs[0]);
            FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
                .getFlashlineRegistry(IURL);

            // //////////////////////////////////////
            // Authenticate with OER
            // //////////////////////////////////////
```

```

AuthToken authToken = repository.authTokenCreate(pArgs[1],
    pArgs[2]);

ProjectEntities IProjectEntities = new ProjectEntities();

String[] assetIds = { "569", "589" };
try {

    // -----
    // read an existing project
    Project projectRead = repository.projectRead(authToken, 50000);

    // -----
    // set the remove assets ids
    IProjectEntities.setRemovedAssetIDs(assetIds);

    // -----
    // update the project
    repository.projectUpdate(authToken, projectRead,
        IProjectEntities);
} catch (APIValidationException ex) {
    ex.printStackTrace();
}

// -----
// revert extractions
repository.extractionResetDatabase();
} 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();
}
}
}

```

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

            // //////////////////////////////////////
            // 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 IEntities = new ProjectEntities();
            Project projectRead = new Project();

            try {

                // -----
                // read an existing project
                projectRead = repository.projectRead(authToken, 50000);
            }
        }
    }
}

```



```

// -----
// set the entities of the produced assets
IEntities.setAssetIDs(assetIDs);

// -----
// update the project
repository.projectUpdate(authToken, projectRead, IEntities);
} catch (OpenAPIException ex) {
    ex.printStackTrace();
}

// -----
// revert extractions
repository.extractionResetDatabase();
} 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: Update a Project's Asset Usage

Description

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

Sample Code:

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

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

```

Use Case: Closing a Project with Hidden Assets

Overview

When closing a project, the project lead is required to update the usage status of assets consumed in the project that have not already been designated as **DEPLOYED** or **REJECTED**.

However, certain **Advanced Role Based Access Control (RBAC)** settings in AquaLogic Enterprise Repository may prevent the project lead from seeing *all* assets consumed by the project.

If the project is closed, any hidden assets not already rejected are automatically designated as **DEPLOYED**.

When using AquaLogic Enterprise Repository, the project lead in this situation is notified that the project contains hidden assets, and is provided with the opportunity to contact users who have the necessary access to update the usage status of the hidden assets and to complete an asset value survey. Once the project lead is confident that the appropriate users have taken the necessary action, he/she can close the project.

The following example demonstrates a programmatic FLEX mechanism for handling the status update of assets that are hidden from the project lead at project closure.

Sample Code:

```
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.decision.ExtractionDeploymentDecision;
```

```

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.RegistryUser;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

```

```

public class CloseProjectWithHiddenAssets {
    public static void main(String[] pArgs) {
        Project mProject = new Project();
        FlashlineRegistry repository = null;
        AuthToken authToken = null;
        try {

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

            // //////////////////////////////////////
            // Authenticate with OER
            // //////////////////////////////////////
            authToken = repository.authTokenCreate(pArgs[1], pArgs[2]);

        } catch (RemoteException IEx) {
            IEx.printStackTrace();
        } catch (ServiceException IEx) {
            IEx.printStackTrace();
        } catch (MalformedURLException IEx) {
            IEx.printStackTrace();
        }
        }

        try {
            mProject = repository.projectRead(authToken, 50000);
            RegistryUser[] IUsers = repository.extractionGetHiddenAssetUsers(
                authToken, mProject.getID());
            System.out.println("Are there any hidden assets?" + IUsers.length);
        } catch (Exception e) {
            e.printStackTrace();
        }

        mProject.setSettleHiddenAssets(false);
        mProject.setClosed(true);
        try {
            repository.projectUpdate(authToken, mProject,

```

```

    new ProjectEntities());
} catch (APIValidationException e) {
    // Error will be caught
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

mProject.setSettleHiddenAssets(true);
mProject.setClosed(true);

try {
    Extraction[] lExtractions = repository.projectReadExtractions(
        authToken, mProject);
    ExtractionDeploymentDecision[] lDecisions = new ExtractionDeploymentDecision[lExtractions.length];

    // This will deploy all viewable consumed assets
    for (int i = 0; i < lExtractions.length; i++) {
        Extraction lEx = lExtractions[i];

        ExtractionDeploymentDecision lDecision = new ExtractionDeploymentDecision();
        lDecision.setExtractionID(lEx.getID());

        lDecision.setStatus("DEPLOYED");
        lDecisions[i] = lDecision;
    }

    ProjectEntities lEntities = new ProjectEntities();
    lEntities.setDeploymentIDs(lDecisions);
    Project lProject = repository.projectUpdate(authToken, mProject,
        lEntities);

    // All extractions (including hidden) should be deployed

    Extraction[] lExs = repository.extractionReadProjectExtractions(
        authToken, lProject.getID(), true);

    for (int i = 0; i < lExs.length; i++) {
        Extraction lEx = lExs[i];

        System.out.println("Extraction is deployed:" + lEx.getStatus());
    }
} catch (Exception e) {
    e.printStackTrace();
}

// revert project to open
mProject.setClosed(false);
try {
    repository.projectUpdate(authToken, mProject,

```

```
        new ProjectEntities());  
    } catch (APIValidationException e) {  
        // Error will be caught  
        e.printStackTrace();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}  
  
}
```


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:

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

            ////////////////////////////////////
```

```

// 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 will be the **ONLY** users/projects assigned to the project. Any users/projects not included in the String Array of IDs will be removed from the project. This option combines adding and removing users into one step.

Sample Code:

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

            // //////////////////////////////////////
            // 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 will be the ONLY
            // users/projects assigned to the project.
            // Any users/projects not included in the String Array of IDs will be
            // 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 IParentProject = createNewProject(repository, authToken, "My Parent Project");
Project IChildProject = createNewProject(repository, authToken, "My Child Project");
String[] newParentIDs = { ""+IParentProject.getID() };
String[] newChildIDs = { ""+IChildProject.getID() };

// -----
// create two new users
RegistryUser IUserOne = createNewUser(repository, authToken, "one");
RegistryUser IUserTwo = createNewUser(repository, authToken, "two");
String[] newMemberIDs = { ""+IUserOne.getID(), ""+IUserTwo.getID() };

// -----
// set the leader ids
IEntities.setLeaderIDs(newLeaderIDs);

// -----
// set the member ids
IEntities.setMemberIDs(newMemberIDs);

// -----
// set the children project ids
IEntities.setChildIDs(newChildIDs);

// -----
// set the parent project ids
IEntities.setParentIDs(newParentIDs);

// -----
// update the project
repository.projectUpdate(authToken, projectRead, IEntities);

} catch (OpenAPIException ex) {
    throw ex;
}
} 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 Project createNewProject(FlashlineRegistry repository, AuthToken authToken, String pName)  
    throws APIValidationException, RemoteException {  
    Project IProject = new Project();  
    ProjectEntities IProjectEntities = new ProjectEntities();  
  
    IProject.setName(pName+" "+Calendar.getInstance().getTimeInMillis()); // force uniqueness  
    IProject.setDepartmentID(50000); // a department with id 50000 must already exist  
    String[] ILeaderIds = new String[] { "99" };  
    IProjectEntities.setLeaderIDs(ILeaderIds);  
  
    IProject = repository.projectCreate(authToken, IProject, IProjectEntities);  
    return IProject;  
}
```

```
protected static RegistryUser createNewUser(FlashlineRegistry repository, AuthToken authToken, String pUserName)  
    throws APIValidationException, RemoteException {  
    String IUserName = pUserName + Calendar.getInstance().getTimeInMillis(); // force uniqueness  
    RegistryUser IRegistryUser = repository.createUser(authToken, IUserName, "First", pUserName,  
        pUserName+"@example.com", pUserName, false, false, false);  
    return IRegistryUser;  
}  
}
```

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:

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

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

            long IParentProjectID = createProject(repository, authToken).getID();
            long IChildProject1ID = createProject(repository, authToken).getID();
            long IChildProject2ID = createProject(repository, authToken).getID();
            long IUser1ID = createUser(repository, authToken).getID();
            long IUser2ID = createUser(repository, authToken).getID();
            long IUser3ID = createUser(repository, authToken).getID();
            long IUser4ID = createUser(repository, authToken).getID();
        }
    }
}
```

```

long IUser5ID = createUser(repository, authToken).getID();
long IUser6ID = createUser(repository, authToken).getID();

// -----
// Remove related projects and users from a project
Project projectRead = new Project();
String[] removedParentProjectIDs = { ""+IParentProjectID };
String[] removedChildProjectIDs = { ""+IChildProject1ID, ""+IChildProject2ID };
String[] removedLeaderIDs = { ""+IUser1ID };
String[] removedMemberIDs = { ""+IUser2ID };
ProjectEntities IEntities = new ProjectEntities();
try {
    projectRead = repository.projectRead(authToken, 50000);
} catch (OpenAPIException ex) {
    throw ex;
}
try {
    // -----
    // set the removed parent project ids
    IEntities.setRemovedParentIDs(removedParentProjectIDs);

    // -----
    // set the removed children project ids
    IEntities.setRemovedChildIDs(removedChildProjectIDs);

    // -----
    // set the remove leader ids
    IEntities.setRemovedLeaderIDs(removedLeaderIDs);

    // -----
    // set the removed member ids
    IEntities.setRemovedMemberIDs(removedMemberIDs);

    // -----
    // set the extraction reassignment decisions
    ExtractionReassignmentDecision[] decisions = new ExtractionReassignmentDecision[2];
    ExtractionReassignmentDecision decision = new ExtractionReassignmentDecision();
    decision.setUserID(IUser3ID);
    decision.setReassignUserID(IUser4ID);
    decisions[0] = decision;
    decision = new ExtractionReassignmentDecision();
    decision.setUserID(IUser5ID);
    decision.setReassignUserID(IUser6ID);
    decisions[1] = decision;

    // -----
    // set the userid for the reassigned extractions
    IEntities.setReassignIDs(decisions);

    // -----
    // update the project
    repository.projectUpdate(authToken, projectRead, IEntities);
} 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:

```

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

// //////////
// 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[] IParentProjectIDs = { "50003" };
String[] IChildProjectIDs = { "50002", "50001" };
String[] ILeaderIDs = { "50001" };
String[] IMemberIDs = { "50005" };
ProjectEntities IEntities = new ProjectEntities();
try {
    projectRead = repository.projectRead(authToken, 50000);
} catch (OpenAPIException ex) {
    throw ex;
}
try {
    // -----
    // set the parent project ids
    IEntities.setParentIDs(IParentProjectIDs);

    // -----
    // set the children project ids
    IEntities.setChildIDs(IChildProjectIDs);

    // -----
    // set the leader ids
    IEntities.setLeaderIDs(ILeaderIDs);

    // -----
    // set the member ids
    IEntities.setMemberIDs(IMemberIDs);

    // -----
    // 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 extractions
    IEntities.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();
}
}
}

```

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:

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

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

            Project IProject = 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
IProject = repository.projectRead(authToken, currentProjectID);
someProjectID = createProject(repository, authToken).getID();

// -----
// get a member of the project to reassign
Project IReassignProject = repository.projectRead(authToken, someProjectID);
String[] memberIDs = repository.projectReadMemberIDs(authToken, IReassignProject);
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 IProjectEntities = new ProjectEntities();
    reassignUserID = createUser(repository, authToken).getID();
    String[] newMemberIDs = { ""+reassignUserID };
    IProjectEntities.setAddedMemberIDs(newMemberIDs);
    repository.projectUpdate(authToken, IReassignProject, IProjectEntities);
}

// -----
// set the extraction reassignment decision
ExtractionReassignmentDecision[] IDecisions = new ExtractionReassignmentDecision[1];
ExtractionReassignmentDecision IDecision = new ExtractionReassignmentDecision();

// -----
// set the reassigned project id
IDecision.setProjectID(someProjectID);

// -----
// specify which extraction (by id)
IDecision.setExtractionID(extractionID);

// -----
// set the reassigned user id
IDecision.setReassignUserID(reassignUserID);
IDecisions[0] = IDecision;

// -----
// reassign project extractions
repository.projectReassignExtractions(authToken, IProject,
    IDecisions);
} 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) {

```

```

        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);
}
}

```

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:

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

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

            Project IProject = 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
IProject = repository.projectRead(authToken, currentProjectID);
someProjectID = createProject(repository, authToken).getID();

// -----
// get a member of the project to reassign
Project IReassignProject = repository.projectRead(authToken, someProjectID);
String[] memberIDs = repository.projectReadMemberIDs(authToken, IReassignProject);
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 IProjectEntities = new ProjectEntities();
    reassignUserID = createUser(repository, authToken).getID();
    String[] newMemberIDs = { ""+reassignUserID };
    IProjectEntities.setAddedMemberIDs(newMemberIDs);
    repository.projectUpdate(authToken, IReassignProject, IProjectEntities);
}

// -----
// set the extraction reassignment decision
ExtractionReassignmentDecision[] IDecisions = new ExtractionReassignmentDecision[1];
ExtractionReassignmentDecision IDecision = new ExtractionReassignmentDecision();

// -----
// set the reassigned project id
IDecision.setProjectID(someProjectID);

// -----
// specify which extraction (by id)
IDecision.setExtractionID(extractionID);

// -----
// set the reassigned user id
IDecision.setReassignUserID(reassignUserID);
IDecisions[0] = IDecision;

// -----
// reassign project extractions
repository.projectReassignExtractions(authToken, IProject,
    IDecisions);
} 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 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);
}
}

```

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:

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

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

            Project IProject = 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
IProject = 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, IProject);
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 IProjectEntities = new ProjectEntities();
    extractionReassignUserID = createUser(repository, authToken).getID();
    IProjectEntities.setAddedMemberIDs(new String[]{ ""+extractionReassignUserID });
    repository.projectUpdate(authToken, IProject, IProjectEntities);
}
if (projectReassignUserID==0) {
    ProjectEntities IProjectEntities = new ProjectEntities();
    projectReassignUserID = createUser(repository, authToken).getID();
    IProjectEntities.setAddedMemberIDs(new String[]{ ""+projectReassignUserID });
    repository.projectUpdate(authToken, IProject, IProjectEntities);
}

// get extraction for user or create one
Extraction[] IAssetExtractions = repository.projectReadExtractions(authToken, IProject);
if (IAssetExtractions!=null && IAssetExtractions.length>0) {
    extractionID = IAssetExtractions[0].getID();
}
if (extractionID==0) {
    // create new extraction
    ProjectEntities IProjectEntities = new ProjectEntities();
    Asset IAsset = repository.assetRead(authToken, 569);
    IProjectEntities.setAssetIDs(new String[]{ ""+IAsset.getID() });
    repository.projectUpdate(authToken, IProject, IProjectEntities);
}

// -----
// 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);
}
}

```

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

Description

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

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

```

// 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;
}
}
```

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:

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

            ////////////////////////////////////////////////////
            // 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 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 createUser(FlashlineRegistry repository, AuthToken authToken) throws OpenAPIException, RemoteException {
    String IUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, IUserName, "First", "User", "user@example.com", "user", false, false, false);
}
}

```

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:

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

            ////////////////////////////////////////////////////
            // 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 IUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.userCreate(authToken, IUserName, "First", "User", "user@example.com", "user", false, false, false);
}
}

```

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:

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

            //////////////////////////////////////////////////
            // 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(new Value()); // Set to a new value.
    ProjectAssetValue projectAssetValueSelection = new ProjectAssetValue();
    projectAssetValue = repository.projectAssetValueUpdate(
        authToken, projectAssetValue, "predicted_selected");
}

} 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) {

```

```
    lEx.printStackTrace();
}
}

protected static RegistryUser createUser(FlashlineRegistry repository, AuthToken authToken) throws OpenAPIException, RemoteException {
    String lUserName = "user"+Calendar.getInstance().getTimeInMillis();
    return repository.createUser(authToken, lUserName, "First", "User", "user@example.com", "user", false, false, false);
}
}
```