

# Relationship Types API

## Overview

### *Description*

The Relationship Type defines the structure of a relationship that can be used to associate two assets.

### *Asset subsystems*

When creating or editing assets, Relationship Types are used to define or modify the relationships that exist between assets.

## Use Cases

### Use Case: Create a new relationship type

#### *Description*

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

#### Sample Code:

```
package com.flashline.sample.relationshiptypeapi;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;

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

public class CreateNewRelationshipType {
    public static void main(String pArgs[]) throws java.rmi.RemoteException, OpenAPIException {
```

```

try{

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

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

    // -----
    // create the new relationship type
    String newRelationshipTypeName = "My-NewRelationshipTypeName"; //Relationship Type name must contain only alpha characters
or hyphens
    RelationshipType newRelationshipType = repository.relationshipTypeCreate(authToken, newRelationshipTypeName);

    System.out.println("The new relationshipType id = "+newRelationshipType.getID()+" ");

// -----
// set the direction definition and the display text describing the relationship type
//// Two-way = "BIDIRECTIONAL"
//// Two-way, order matters = "ORDERED-BIDIRECTIONAL"
//// One-way = "UNIDIRECTIONAL"
newRelationshipType.setDirection("ORDERED-BIDIRECTIONAL");
newRelationshipType.setDisplayPrimary("Contained In"); // Source Asset - 'Contained In' - Target Asset
newRelationshipType.setDisplaySecondary("Contains"); // Target Asset - 'Contains' - Source Asset
newRelationshipType = repository.relationshipTypeUpdate(authToken, newRelationshipType);

// -----
// delete the new relationship type
repository.relationshipTypeDelete(authToken, newRelationshipType.getID());

} catch (OpenAPIException 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: Modify related assets

### Description

A target asset is related to other assets using *My RelationshipType*. Using this same relationship type, establish a relationship to an additional asset.

## Sample Code:

```
package com.flashline.sample.relationshipypeapi;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;

import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RelationshipType;
import com.flashline.registry.openapi.query.RelationshipTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

public class FindRelationshipTypeAndUseInAsset {

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

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

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

            Asset myAsset = repository.assetRead(authToken, 563);
            //MY_OTHER_ASSET_ID should be an integer and should be the id of an asset in the repository

            RelationshipType[] allRelationshipTypes = getAllRelationshipTypes(repository, authToken);
            for (int i = 0; i < allRelationshipTypes.length; i++) {
                if (allRelationshipTypes[i].getName().equals("MyRelationshipType2")) {
                    //This is the relationship type, modify the assets that are related
                    // using it
                    RelationshipType myRelationshipType = allRelationshipTypes[i];
                    Asset otherAsset = repository.assetRead(authToken, 569); //569= MY_OTHER_ASSET_ID
                    //MY_OTHER_ASSET_ID should be an integer and should be the id of an asset in the repository
                    //add this asset to the list of related assets
                    long[] oldSecondaryIDs = myRelationshipType.getSecondaryIDs();
                    long[] newSecondaryIDs = new long[oldSecondaryIDs.length + 1];
                    for (int j = 0; j < oldSecondaryIDs.length; j++) {
                        newSecondaryIDs[j] = oldSecondaryIDs[j];
                    }
                    newSecondaryIDs[newSecondaryIDs.length - 1] = otherAsset.getID();
                    myRelationshipType.setSecondaryIDs(newSecondaryIDs);
                }
            }
            myAsset.setRelationshipTypes(allRelationshipTypes);
            repository.assetUpdate(authToken, myAsset);

        }catch(OpenAPIException IEx) {
            System.out.println("ServerCode = "+ IEx.getServerErrorCode());
            System.out.println("Message = "+ IEx.getMessage());
        }
    }
}
```

```

        System.out.println("StackTrace:");
        lEx.printStackTrace();
    } catch (RemoteException lEx) {
        lEx.printStackTrace();
    } catch (ServiceException lEx) {
        lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
        lEx.printStackTrace();
    }
}

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

```

### *Pitfalls*

Methods to avoid:

- `SetPromptNotifySecondary()`

## **Use Case: Query related assets**

### *Description*

Querying for related asset types.

Sample Code:

```

package com.flashline.sample.relationshiptypeapi;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;

import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.Asset;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.RelationshipType;
import com.flashline.registry.openapi.query.RelationshipTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

```

```

public class FindRelationshipTypeAndUseInAsset {

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

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

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

            Asset myAsset = repository.assetRead(authToken, 563);
            //MY_OTHER_ASSET_ID should be an integer and should be the id of an asset in the repository

            RelationshipType[] allRelationshipTypes = getAllRelationshipTypes(repository, authToken);
            for (int i = 0; i < allRelationshipTypes.length; i++) {
                if (allRelationshipTypes[i].getName().equals("MyRelationshipType2")) {
                    //This is the relationship type, modify the assets that are related
                    // using it
                    RelationshipType myRelationshipType = allRelationshipTypes[i];
                    Asset otherAsset = repository.assetRead(authToken, 569); //569= MY_OTHER_ASSET_ID
                    //MY_OTHER_ASSET_ID should be an integer and should be the id of an asset in the repository
                    //add this asset to the list of related assets
                    long[] oldSecondaryIDs = myRelationshipType.getSecondaryIDs();
                    long[] newSecondaryIDs = new long[oldSecondaryIDs.length + 1];
                    for (int j = 0; j < oldSecondaryIDs.length; j++) {
                        newSecondaryIDs[j] = oldSecondaryIDs[j];
                    }
                    newSecondaryIDs[newSecondaryIDs.length - 1] = otherAsset.getID();
                    myRelationshipType.setSecondaryIDs(newSecondaryIDs);
                }
            }
            myAsset.setRelationshipTypes(allRelationshipTypes);
            repository.assetUpdate(authToken, myAsset);

        }catch(OpenAPIException 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();
        }
    }

}

/**
 * This method returns every relationship type in the repository
 * @param repository
 * @param authToken
 * @return
 * @throws RemoteException
 */
public static RelationshipType[] getAllRelationshipTypes(FlashlineRegistry repository, AuthToken authToken) throws RemoteException {

```

```

//Create an empty relationship type criteria object
RelationshipTypeCriteria criteria = new RelationshipTypeCriteria();
criteria.setNameCriteria("");
RelationshipType[] allRelationshipTypes = repository.relationshipTypeQuery(authToken, criteria);
return allRelationshipTypes;
}
}

```

### *Example of the RelationshipTypeQuery*

```

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

```