

# Oracle Enterprise Repository

## Role API

### Overview

The Role Subsystem provides a Web Services-based mechanism that can be used to create, read, update, query, and otherwise manipulate Oracle Enterprise Repository Roles.

#### *Related subsystems*

- [User Subsystem](#)

#### *Additional Import(s) Required*

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

### Use Cases

#### Use Case: Manipulate Roles

##### *Description*

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

## Sample Code:

```
package com.flashline.sample.roleapi;

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

import javax.xml.rpc.ServiceException;

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

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

            //////////////////////////////////////
            // Connect to Oracle Enterprise Repository
            //////////////////////////////////////
            URL 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 a new role
            String IName = "new_role_type_name";
            String IDesc = "new_role_type_desc";
            boolean IDefaultForNewUser = true;
            Role IRole = null;
            IRole = repository.roleCreate(authToken, IName, IDesc,
                IDefaultForNewUser);

            // -----
            // Read a Role
            String IRoleName = IName;
            Role IRole2 = null;
```

```

IRole2 = repository.roleRead(authToken, IRoleName);

// -----
// Update a Role
String IRoleName3 = IName;
Role IRole3 = null;
IRole3 = repository.roleRead(authToken, IRoleName3);
IRole3.setName("user_modified");
IRole3.setStatus(20);
IRole3 = repository.roleUpdate(authToken, IRole);

// -----
// Delete a Role
String IRoleName4 = "user_modified"; // role name must exist in OER
Role IRole4 = repository.roleRead(authToken, IRoleName4);
if (IRole4==null) {
    IRole4 = repository.roleRead(authToken, IName);
}
if (IRole4!=null) {
    try {
        repository.roleDelete(authToken, IRole4.getName());
    } catch (OpenAPIException e) {
        e.printStackTrace();
    }
}

// -----
// This method is used to query for roles.
Role[] IRoles = null;
RoleCriteria IRoleCriteria = null;
IRoleCriteria = new RoleCriteria();
IRoleCriteria.setNameCriteria("user");
IRoles = repository.roleQuery(authToken, IRoleCriteria);

} 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();
}
}
}

```