

## Oracle Enterprise Repository

# Localization of REX Clients

## Overview

Statuses are passed back to a REX client as either an `OpenAPIException` or `AuditMsg` object. `OpenAPIException` objects are used for exceptions, whereas, `AuditMsg` objects are used for processes that run asynchronously. Both of these objects return a text error message to the REX client.

The interface of both objects has been expanded to include an error code and a list of message arguments so that REX clients can display error or status messages in another language. Clients can continue to use the standard error messages or they can ignore the message and use the error code and the message arguments to construct their own error message.

For example, if you want to localize an application that uses REX, you would first get the properties file listing all the possible error messages. The messages look something like this:

```
ERR_9008 = Error updating project with ID = [{0}].
```

Then you must translate all the messages as necessary:

```
ERR_9008 = Errorway updatingay ojectpray ithway IDway = [{0}].
```

If the client code tries to modify a project with ID=123, and that modification fails, then your end-users will get an exception with this error message:

```
Error updating project with ID = [123].
```

If you want to display that error in a local language (such as, Pig Latin), you would take the error code, 9008, and look it up in your translated file to get the string "Errorway updatingay ojectpray ithway IDway = [{0}]." Then you would use the message arguments to replace the tokens. In this case, there is only one string, "123", so you should be able to find one message argument.

You can then construct a custom error message for your end-users:

Errorway updatingay ojectpray ithway IDway = [123].

## Use Cases

### Use Case: Creating localized messages from REX Exceptions

#### *Description*

- From the OpenAPIException get the server error code and the message arguments
- Get the resource bundle for the OpenAPIExceptions appropriate for the client locale
- Get the string associated with the error code and replace the token with the message arguments

OpenAPIException Sample Code:

```
package com.flashline.sample.localization;

import java.net.URL;
import java.text.MessageFormat;
import java.util.ResourceBundle;

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

public class SyncTest {
    private static final int INVALID_PROJECT_ID = 8672609;

    public static void main(String[] args) throws Exception {
        URL IURL = new URL("http://localhost:9080/registry/services/FlashlineRegistry");

        FlashlineRegistry reg = new FlashlineRegistryServiceLocator().getFlashlineRegistry(IURL);
        AuthToken token = reg.authTokenCreate("admin", "n0pa55w0rd");

        try {
            Project project = reg.projectRead(token,INVALID_PROJECT_ID);
        } catch (OpenAPIException ex) {
            String msg = createMessage(ex.getServerErrorCode(),ex.getMessageArguments());
            System.out.println(msg);
        }
    }

    private static String createMessage(int pServerErrorCode, Object[] pArgs) {
```

```

ResourceBundle mResourceBundle = ResourceBundle.getBundle("com.flashline.sample.localization.sync_error_messages");
return MessageFormat.format(mResourceBundle.getString("ERR_"+pServerErrorCode), pArgs);

}

}

```

## Use Case: Creating localized messages from REX Audit Messages

- From the AuditMsg and the ImpExpJob get the server error code and the message arguments from the AuditMsg
- Get the resource bundle for audit messages appropriate for the client locale
- Get the string associated with the error code and replace the token with the message arguments

### [?AuditMsg](#) Sample Code:

```

package com.flashline.sample.localization;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.text.MessageFormat;
import java.util.ResourceBundle;

import javax.activation.DataHandler;
import javax.xml.rpc.ServiceException;

import org.apache.axis.client.Stub;
import org.apache.soap.util.mime.ByteArrayDataSource;

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

public class AsyncTest {

    public void run() throws MalformedURLException, ServiceException, OpenAPIException, RemoteException{

        URL IURL = new URL("http://localhost:9080/registry/services/FlashlineRegistry");

        FlashlineRegistry reg = new FlashlineRegistryServiceLocator().getFlashlineRegistry(IURL);
        AuthToken token = reg.authTokenCreate("admin", "n0pa55w0rd");

        try {

```

```

File IFile = new File("samples/com/flashline/sample/localization/asynctest.zip");
//Import the file and save to db
InputStream IIS = new FileInputStream(IFile);

ByteArrayDataSource IDataSource = new ByteArrayDataSource(IIS, "application/x-zip-compressed");

DataHandler IDH = new DataHandler(IDataSource);
// add the attachment
((Stub)reg).addAttachment(IDH);

ImpExpJob IJob = reg.importExecute(token, "flashline", null, "Import Assets Test", null);

boolean IPassed = false;
for(int i=0; i<1000; i++){
    IJob = reg.importStatus(token, IJob);
    System.out.println("Import Job ["+IJob.getID()+"] - State: "+IJob.getState());

    String msg = createMessage(IJob.getAuditMsg().getSummaryID(),IJob.getAuditMsg().getSummaryArgs());
    System.out.println(msg);

    if( IJob.getState().equals("completed")){
        IPassed = true;
        break;
    }

    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

if (IPassed){
    System.out.println("Import Completed");
}

} catch (OpenAPIException ex) {
    String msg = createMessage(ex.getServerErrorCode(),ex.getMessageArguments());
    System.out.println(msg);
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

/**
 * @param args
 * @throws ServiceException
 * @throws RemoteException
 * @throws MalformedURLException
 * @throws OpenAPIException
 */

```

```

public static void main(String[] args) throws OpenAPIException, MalformedURLException, RemoteException, ServiceException {
    AsyncTest test = new AsyncTest();
    test.run();
}

private static String createMessage(int pServerErrorCode, Object[] pArgs) {
    ResourceBundle mResourceBundle = ResourceBundle.getBundle("com.flashline.sample.localization.async_error_messages");
    return MessageFormat.format(mResourceBundle.getString("ERR_" + pServerErrorCode), pArgs);
}

private String readZip(String pFileName) throws IOException {
    int    lNumRead = 0;
    char[] lBuf = new char[2048];
    StringBuffer lQuery = new StringBuffer();

    InputStreamReader lReader = new InputStreamReader(getClass().getResourceAsStream(pFileName));

    while( (lNumRead=lReader.read(lBuf)) != -1){
        lQuery.append(lBuf, 0, lNumRead);
    }

    return lQuery.toString();
}
}

```