Designing with Sun JCA Adapters

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–4719–10
April 2008

090911@22749

Contents

Designing with Sun JCA Adapters

The following sections provide installation and deployment instructions for Sun JCA Adapters. Some of the information pertains to deploying a specific Oracle project. The following topics are covered:

Technical Overview for Sun JCA Adapters

The components provided for developing Sun JCA Adapters include the runtime JCA resource adapter component as well as the design time tools to facilitate developing applications that utilize the JCA resource adapter. The design time tools included are:

- Code wizards that generate template JCA resource adapter client code.
- Wizards that generate code based on higher level object models pertaining to the external applications. The model used is referred to as Object Type Definitions (OTD) and the application-specific wizards are referred to as OTD Wizards.

Inbound JCA Resource Adapter Client Code

JCA Resource Adapters that support inbound communication depend on deployment of a Message-Driven Bean (MDB). To develop an application utilizing a JCA Adapter for inbound communication, the JCA Message-Drive Bean wizard must be used. When the wizard is finished, the message driven bean code skeleton is generated. For example, for the File JCA Adapter, the following MDB code skeleton is created:

```
/*

* To change this template, choose Tools | Templates

* and open the template in the editor.

*/

package test;

import javax.ejb.MessageDriven;

import com.stc.connector.fileadapter.eway.FileListener;

import com.stc.connector.framework.eway.FaultException;

import com.stc.connector.framework.eway.InboundException;

/**

*

* @author SUN

*/

@MessageDriven(name="test.FileJCAMessageBean")

public class FileJCAMessageBean implements FileListener {

public FileJCAMessageBean() {

}

public void onContents(byte[] data, String encoding) throws FaultException,

InboundException {

// implement listener interface here

}

}
```

Note – The MDB generated implements the associated Listener interface for the JCA Adapter specified in the JCA Message-Driven Bean wizard. In the case of File JCA Adapter, the FileListener interface is implemented.

Outbound JCA Resource Adapter Client Code

JCA Adapters that support outbound connection to an external application are utilized via the client interface supported by the adapter. The client interface supported by Sun JCA Adapters are based on a simple Application Client Interface. The client interface involves establishment of an application connection via JNDI. The application connection is a container managed connection obtained from a connection pool. The application connection is used along with the OTD objects provided with the JCA adapter to perform operations on the external application.

The tooling for creating code, based on the outbound JCA's client interface is provided via the adapter in the form of specific objects from the Palette draggable onto the code editor. The Palette is made visible by selecting Palette from the Window top-level drop-down menu.

Upon dragging an item from the Palette to the code editor, the JCA wizard is started. In the JCA wizard, a method name must be specified. The method will be added to the code being edited. Depending on the type of external dragged from the Palette, the choice of OTD to be used may need to be specified. The JCA client interface code, required to obtain a JCA connector resource from JNDI, establishes then closes the JCA connection. It then instantiates the associated OTD object, which is also added. The method specified in the JCA wizard is created to simplify development of application logic as the method implementation can simply use OTD objects passed in.

For example, dragging the File object from the Palette to the onContents method of the File MDB and specifying a method name "send" in the JCA wizard adds the following code in the onContents method implementation:

```
public void onContents(byte[] data, String encoding) throws FaultException,

InboundException {

try {

_invoke_send(data, encoding);

} catch (java.lang.Throwable t) {

ectx.setRollbackOnly();

java.util.logging.Logger.getLogger(this.getClass().getName(

)).log(java.util.logging.Level.WARNING, "Failed to invoke _invoke_send: " + t,
t);
}

}
```

The user application logic can be implemented in the send method which works directly with the OTD objects passed in.

```
private void send(byte[] data, String encoding,
com.stc.connector.fileadapter.appconn.FileClientApplication fileOTD) throws
java.lang.Exception {

}
```

The additional code shown below are also added providing the connection establishment and OTD instantiation.

```
// <editor - fold defaultstate= collapsed desc="Connection setup and takedown.

Click on the + sign on the left to edit the code.">

private void _invoke_send(byte[] data, String encoding) throws
java.lang.Exception {

com.stc.connector.appconn.common.ApplicationConnection fileConnection = null;

try {

if (fileConnection != null) {

fileConnection.close();

}

} catch (Exception e) {

}

}

} // </editor-fold>

// <editor-fold defaultstate="collapsed" desc="file resource declaration. Click
on the + sign on the left to edit the code.">

// comments for inserted variable

@javax.annotation.Resource(name = "jca/file", description = "", shareable =
false)

private com.stc.connector.appconn.common.ApplicationConnectionFactory file; //
</editor-fold>
```

```
// <editor-fold defaultstate="collapsed" desc="EJBContext declaration. Click on
the + sign on the left to edit the code.">

@javax.annotation.Resource

private javax.ejb.EJBContext ectx; // <editor-fold>
```

Object Type Definition Wizards

The client interface to certain external applications may involve dynamically generating Object Type Definition code based on the external application metadata. This step is accomplished by executing the OTD wizards. The jar containing the generated OTD code is added to the NetBeans project where it can be used as a library.

The classes generated by the OTD wizard varies for each external system or application. The database OTD wizards, for instance, will generate code that will facilitate querying or updating databases while the SAP OTD wizards will provide objects representing SAP BAPIs or IDocs to facilitate operations on those entities. For more detailed information on the specific OTDs or models, refer to the corresponding sections on Designing with the respective JCA Adapters.

Installing the Sun JCA Adapters

There are three components to install for the Sun JCA Adapters: design–time components in NetBeans, runtime components in GlassFish, and third-party JAR files.

Perform the following steps to install the Adapters:

- "Installing the NetBeans Modules" on page 9
- "Installing the Runtime Components for Sun JCA Adapters" on page 12
- "Installing Third-Party JAR Files" on page 13

Installing the NetBeans Modules

This topic describes installation instructions for the NetBeans modules. A common library package (contained in the commonLib directory) is a prerequisite before installing any individual components.

▼ Installing the Modules Pack

1. **Extract the contents of the `AdapterPack.zip` file to a local directory.**

2. **To install the common library modules (NBMs) launch NetBeans.**

3. **Click the Tools menu and then select Plugins.**

4 **Click on the Downloaded tab, and then click Add Plugins.**

5 **Navigate to the** *ZipFile*`\AdapterPack\NetBeansModules\CommonLib` **directory, where** *ZipFile* **is the directory where the** `AdapterPack.zip` **file was extracted, and select all the available NBM files.**

Note – If you are installing the NBMs into a Java CAPS environment, you should only need to install `com-sun-soabi-common-wizard-library.nbm`. The remaining libraries should be pre-installed.

6 **Right-click in the Plugins list and click Select All.**

7 **Click Install at the bottom of the window.**

8 **Follow the steps on the NetBeans IDE Installer. Ignore any warnings that may appear.**

9 **Repeat the above steps, selecting all the NBM files in** *ZipFile*`\AdapterPack\NetBeansModules` **directory, where** *ZipFile* **is the directory where the** `AdapterPack.zip` **file was extracted.**

This installs the necessary Wizards and tools. The image below shows the Oracle OTD Wizard, for example.

Note – If you are installing the JCA Adapters into a Java CAPS environment, many of the NBMs in this directory are pre-installed; however, you do need to install the wizards and certain support modules. If you select all NBMs to add, the installer will only install the ones that are not already installed.

Installing the Runtime Components for Sun JCA Adapters

Follow the instructions below to install the required base components for Sun JCA Adapters.

▼ To Install Runtime Components

Before You Begin Make sure you have GlassFish Server installed. You can access GlassFish downloads from `https://glassfish.dev.java.net/public/downloadsindex.html#top`.

1. **Stop all running GlassFish domains.**

2. **Extract the contents of the** `AdapterPack.zip` **file to a local directory.**

3. **Download the JBI core installer from** `http://download.java.net/jbi/binaries/open-esb/main/nightly/latest/CORE/jbi-core-installer.jar` **or use the** `jbi-core-installer.jar` **file available in your GlassFish installation.**

4. **Run the JBI Core Installer by executing the following command:**

   ```
   java -jar jbi-core-installer.jar GlassfishHome install
   ```

   where *GlassFishHome* is the full path to the GlassFish directory.

5. **Start the GlassFish Server.**

6. **In the directory where you extracted the Adapter Pack, navigate to** `Runtime/install` **and open** `install.properties` **in a text editor.**

7. **Modify the properties for your GlassFish installation, and then save and close the file. Be sure to uncomment the line that specifies your operating system.**

8. **Run the following command from** *GlassFishHome*`/bin`**:**

   ```
   asant -f AdapterPack/Runtime/install.xml
   ```

   where *AdapterPack* is the directory where you extracted the Adapter Pack.

Installing Third-Party JAR Files

JCA Adapters for certain systems, such as SAP and Siebel, require that third-party JAR files be installed. The Adapter Pack provides a JAR file installer that you can access by extracting `com-sun-adapters-thirdpartylib-installer.zip`. The extracted files include a readme file to help you through the installation You can also find additional information and instructions at *Installing Third-Party JAR Files*.

Configuring Runtime Components in an EJB/JCA Application

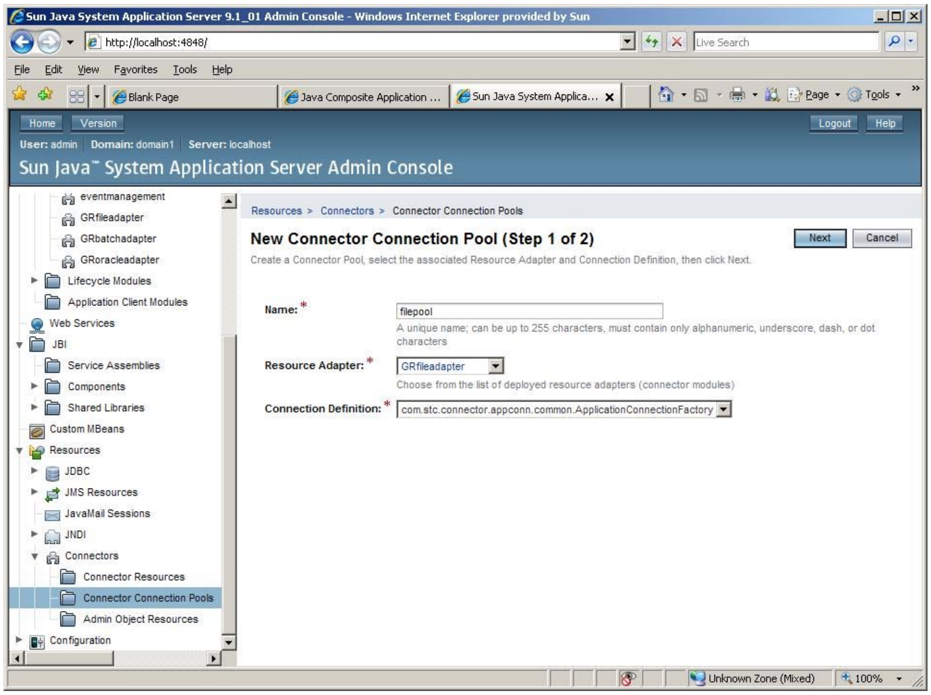
The following task outlines the steps need to configure runtime components in an EJB/JCA application Specific examples are provided for File and Oracle JCA adapters.

▼ Configuring Connector Pools for File Adapter

1. **To create an Outbound Connector Connection Pool for the File Adapter, navigate to the Application Server Admin Console at** `htt://localhost:4848`**.**

2. **Select the Resources > Connectors > Connector Connection Pools tree node, and click New.**

3. **Enter a name for the pool name, and select GRfileadapter as the Resource Adapter type.**



New Connector Connection Pool (Step 1 of 2)

Create a Connector Pool, select the associated Resource Adapter and Connection Definition, then click Next.

Name: * filepool
A unique name; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

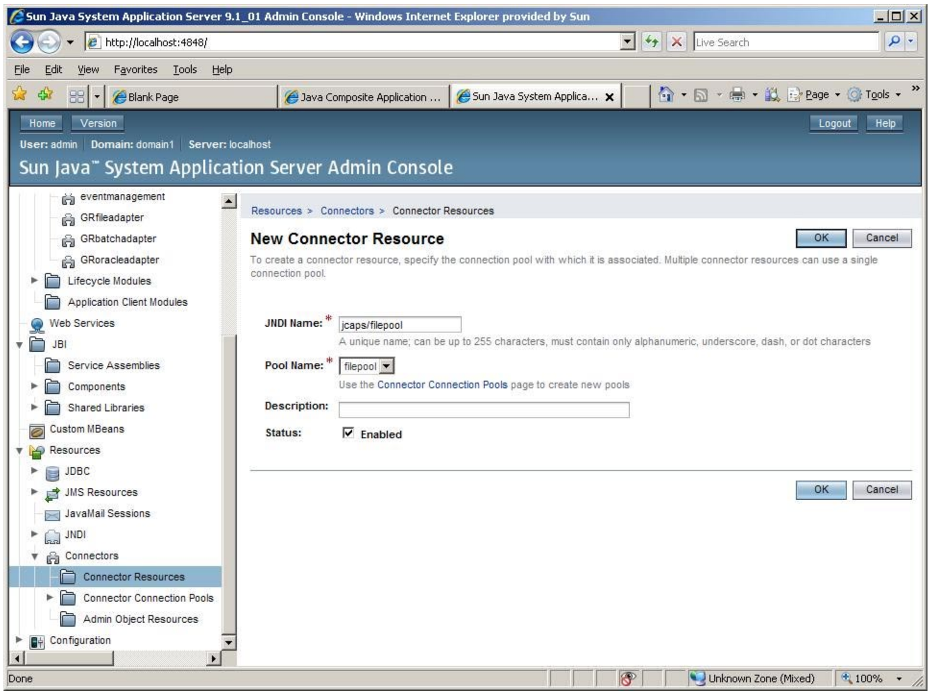
Resource Adapter: * GRfileadapter
Choose from the list of deployed resource adapters (connector modules)

Connection Definition: * com.stc.connector.appconn.common.ApplicationConnectionFactory

4 **Click Next, and then Finish.**

5 **Create an Outbound Connector Resource from the Resource Pool above. In the Admin Console, select Resources > Connectors > Connector Resources, and click New.**

6 **Fill in the JNDI Name for the Resource, and select the name of the Resource Pool created previously, and click OK.**



New Connector Resource

To create a connector resource, specify the connection pool with which it is associated. Multiple connector resources can use a single connection pool.

JNDI Name: * jcaps/filepool
A unique name; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

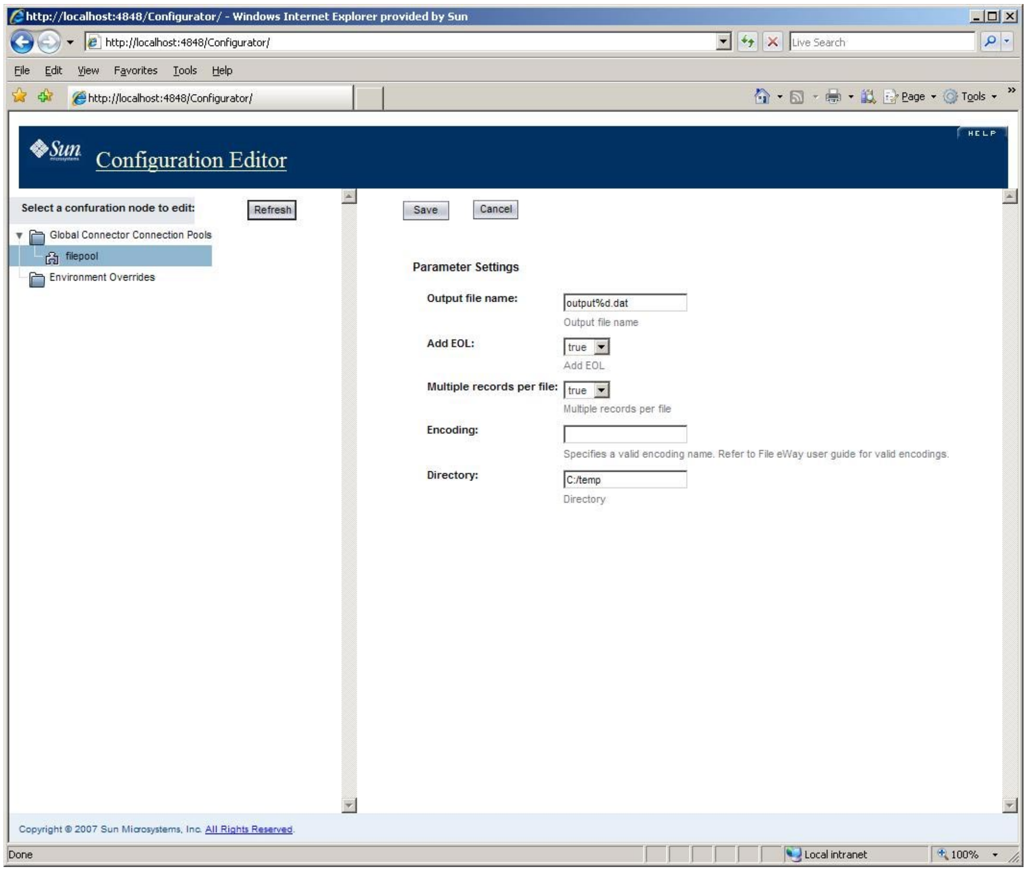
Pool Name: * filepool
Use the Connector Connection Pools page to create new pools

Description:

Status: ☑ Enabled

7 **To configure the File outbound pool, navigate to the web-based configuration editor at** `htt://localhost:4848/Configurator`**.**

8 **Select filepool from tree.**


http://localhost:4848/Configurator/ - Windows Internet Explorer provided by Sun
File Edit View Favorites Tools Help
Sun Configuration Editor
HELP
Select a confuration node to edit:
Refresh
Global Connector Connection Pools
filepool
Environment Overrides
Save
Cancel
Copyright © 2007 Sun Microsystems, Inc. All Rights Reserved.
Done
Local intranet
100%

Parameter Settings

Output file name: output%d.dat
Output file name

Add EOL: true
Add EOL

Multiple records per file: true
Multiple records per file

Encoding:
Specifies a valid encoding name. Refer to File eWay user guide for valid encodings.

Directory: C:/temp
Directory

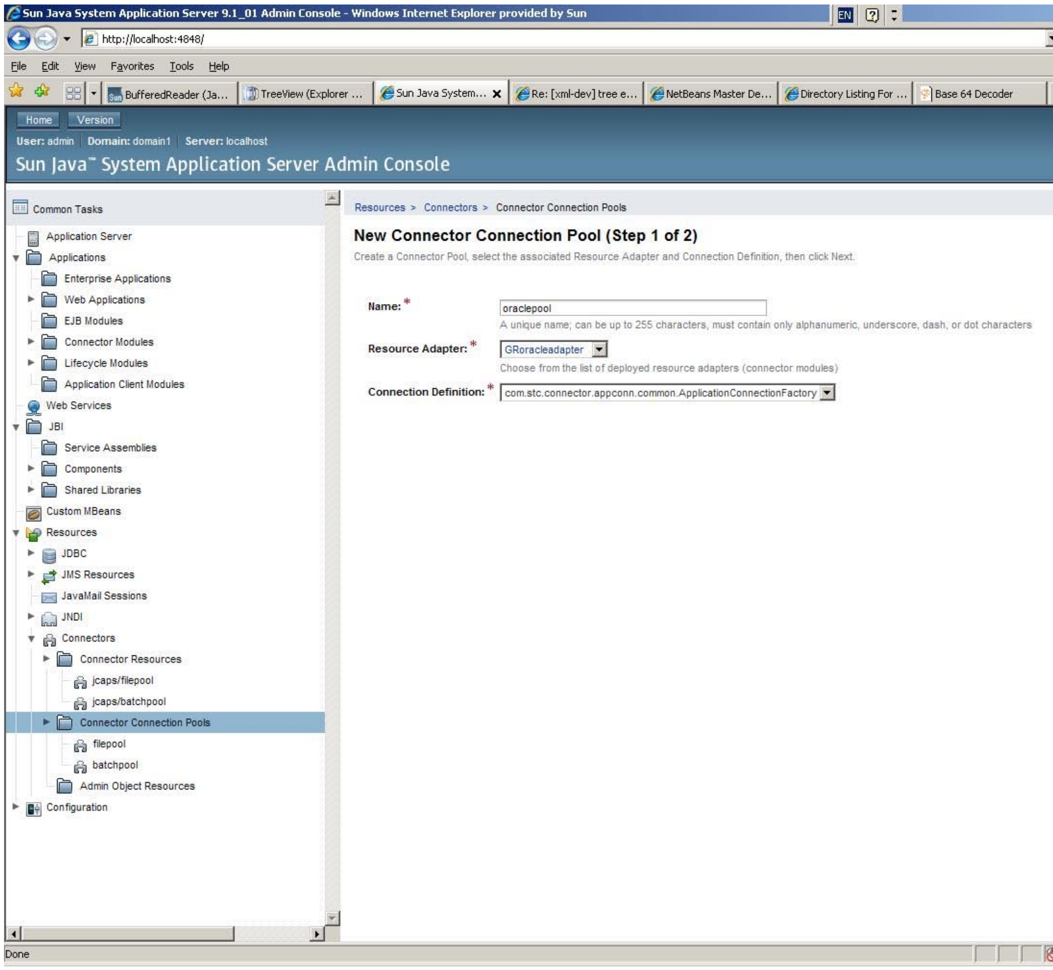
9 **Modify the Output file name, Add EOL, Multiple records per file, Encoding, and Directory as required for your project.**

▼ Configuring Connector Pools for Oracle Adapter

1 **To create an Outbound Connector Connection Pool for the File Adapter, navigate to the Application Server Admin Console at** `htt://localhost:4848`**.**

2 **Select the Resources > Connectors > Connector Connection Pools tree node, and click New.**

3 **Enter a name for the pool name, and select GRoracleadapter as the Resource Adapter type.**



Resources > Connectors > Connector Connection Pools

New Connector Connection Pool (Step 1 of 2)

Create a Connector Pool, select the associated Resource Adapter and Connection Definition, then click Next.

Name: * oraclepool
A unique name; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

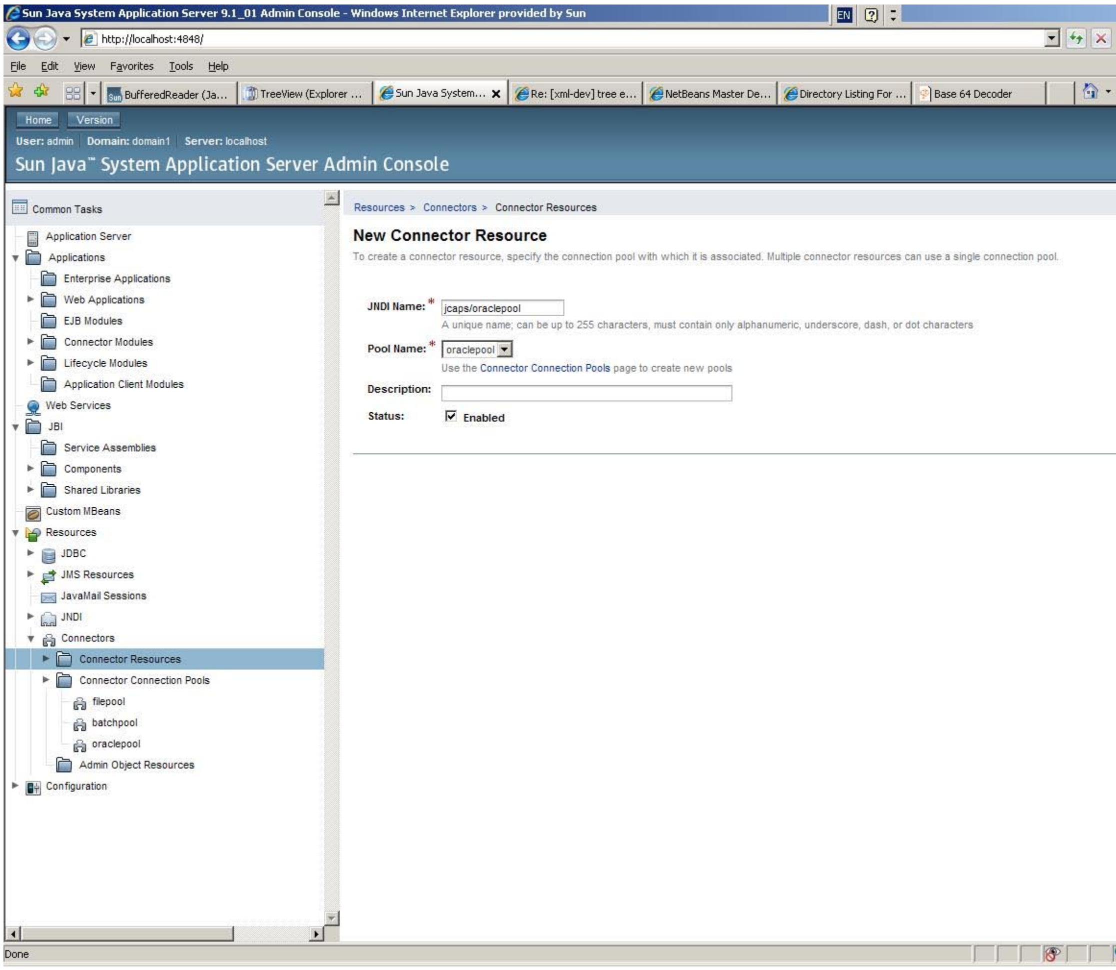
Resource Adapter: * GRoracleadapter
Choose from the list of deployed resource adapters (connector modules)

Connection Definition: * com.stc.connector.appconn.common.ApplicationConnectionFactory

4 **Click Next, and then Finish.**

5 **Create an Outbound Connector Resource from the Resource Pool above. In the Admin Console, select Resources > Connectors > Connector Resources, and click New.**

6 **Fill in the JNDI Name for the Resource, and select the name of the Resource Pool created previously, and click OK.**



New Connector Resource

To create a connector resource, specify the connection pool with which it is associated. Multiple connector resources can use a single connection pool.

JNDI Name: * jcaps/oraclepool
A unique name; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

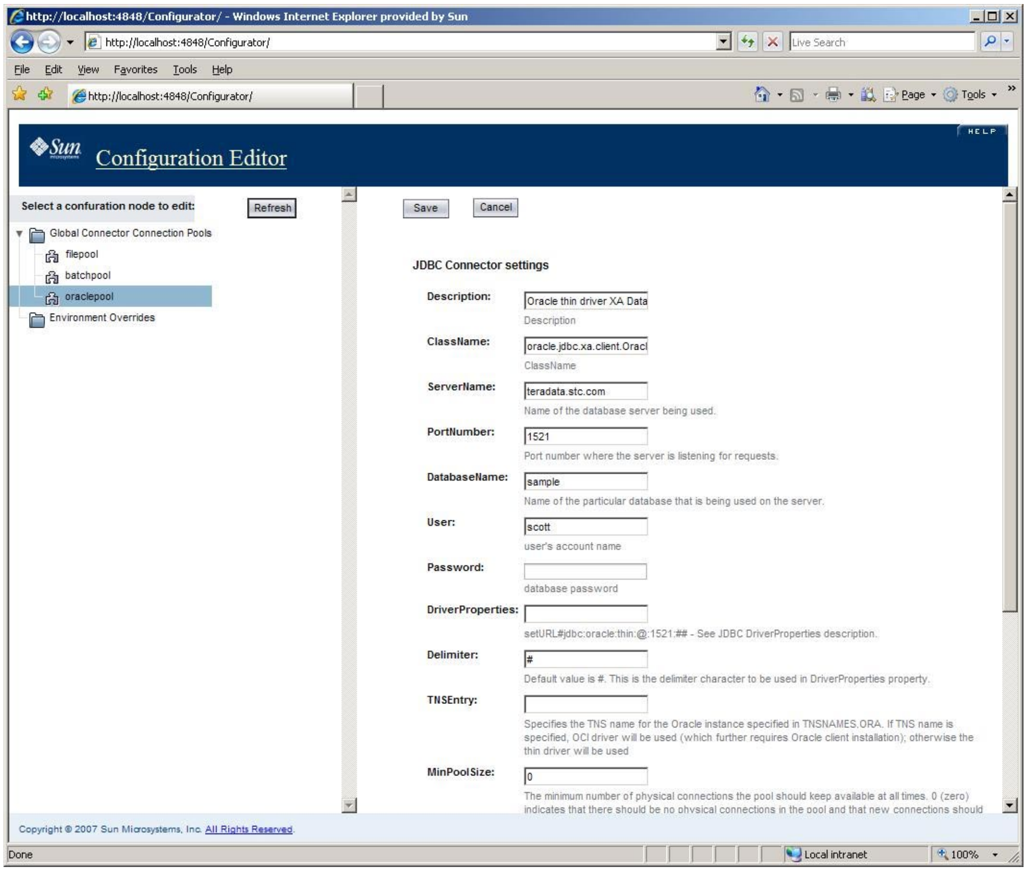
Pool Name: * oraclepool
Use the Connector Connection Pools page to create new pools

Description:

Status: ☑ Enabled

7 To configure the Oracle outbound pool, navigate to the web-based configuration editor at `htt://localhost:4848/Configurator`.

8 Select oraclepool from tree.


http://localhost:4848/Configurator/ - Windows Internet Explorer provided by Sun
http://localhost:4848/Configurator/
Live Search
File Edit View Favorites Tools Help
Page Tools
HELP
Sun
Configuration Editor
Select a confuration node to edit:
Refresh
Global Connector Connection Pools
filepool
batchpool
oraclepool
Environment Overrides
Save
Cancel
Copyright © 2007 Sun Microsystems, Inc. All Rights Reserved.
Done
Local intranet
100%

JDBC Connector settings

Description: Oracle thin driver XA Data
Description

ClassName: oracle.jdbc.xa.client.Oracl
ClassName

ServerName: teradata.stc.com
Name of the database server being used.

PortNumber: 1521
Port number where the server is listening for requests.

DatabaseName: sample
Name of the particular database that is being used on the server.

User: scott
user's account name

Password:
database password

DriverProperties:
setURL#jdbc:oracle:thin:@:1521:## - See JDBC DriverProperties description.

Delimiter: #
Default value is #. This is the delimiter character to be used in DriverProperties property.

TNSEntry:
Specifies the TNS name for the Oracle instance specified in TNSNAMES.ORA. If TNS name is specified, OCI driver will be used (which further requires Oracle client installation); otherwise the thin driver will be used

MinPoolSize: 0
The minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and that new connections should

9 Modify the following fields as required for your project:

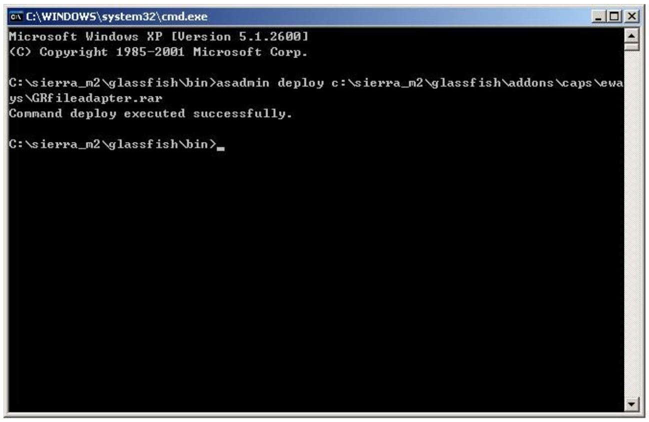
- Description
- ClassName
- ServerName
- PortNumber
- User
- Password
- DriverProperties
- Delimiter
- TNSEntry
- MinPoolSize

Deployment of Sun JCA Adapters

The Sun JCA Adapter runtime components (RAR files) are contained in the Adapter Pack under the `Runtime/adapters` directory.

To Deploy JCA Adapters via Command Line

Use the `asadmin` command to deploy a RAR file. For example, to deploy the File JCA RAR file`sun-file-adapter.rar`, navigate to the `c:\glassfish\bin`directory (where `c:\glassfish` is the full path of the GlassFish installation) run the following command: `asadmin deploy c:\AdapterPack\Runtime\eways\GRfileadapter.rar`.

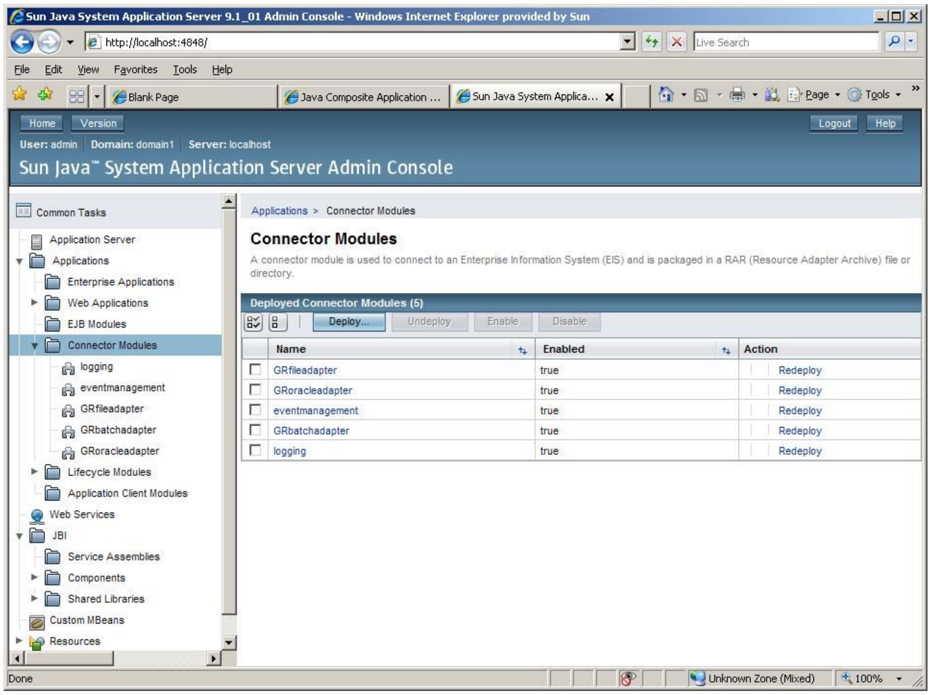


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\sierra_m2\glassfish\bin>asadmin deploy c:\sierra_m2\glassfish\addons\caps\ewa
ys\GRfileadapter.rar
Command deploy executed successfully.

C:\sierra_m2\glassfish\bin>_
```

To Deploy JCA Adapters via the Admin Console

1. Start the application server.
2. Log in to the Admin Console from a browser. If the application server is running locally, browse to `http://localhost:4848`.
3. Locate the **Connector Modules** subnode under the **Applications** node.
4. Click **Deploy**.

5. Specify the RAR file you want to deploy by clicking on the **Browse** button under the selected check box **Packaged file to be uploaded to the server**. The RAR files are available in the `c:\AdapterPack\Runtime` directory where `c:\AdapterPack` is the location of the extracted contents of the `AdapterPack.zip` file.
6. Click **OK** and then **Finish**.
7. Verify successful deployment of the JCA Adapters in the Admin Console.



Applications > Connector Modules

Connector Modules

A connector module is used to connect to an Enterprise Information System (EIS) and is packaged in a RAR (Resource Adapter Archive) file or directory.

Deployed Connector Modules (5)

Deploy... Undeploy Enable Disable

	Name	Enabled	Action
☐	GRfileadapter	true	Redeploy
☐	GRoracleadapter	true	Redeploy
☐	eventmanagement	true	Redeploy
☐	GRbatchadapter	true	Redeploy
☐	logging	true	Redeploy

Using the Oracle Wizard and JCA Adapter Tooling with an EJB Project

The following task outlines the steps need to implement a specific EJB project where a directory is polled for input data using a query statement (the statement is executed against an Oracle database), and the results are written to an output file.

To implement the Oracle JCA Adapter with an EJB Project

1. **Create the required connector or connectors in the Application Server. In this particular example, create a connector for the File eWay and another for the Oracle eWay.**
2. **Navigate to the Application Server Admin Console at** `htt://localhost:4848`**.**
3. **Configure the parameters for your connectors under Resources > Connector > Connector Connection Pool and Resources > Connector > Connector Resources.**
4. **Launch NetBeans.**
5. **From the File menu, select Enterprise > EJB Module.**
6. **Create a new EJB project OracleInvokeEJB**
7. **Launch the Oracle wizard by right—clicking on the OracleInvokeEJB project and selecting New.**

8 **Select Other > SOA > Oracle OTD Wizard.**

9 **Click Next.**

10 **Enter the Oracle database connection information, and click Next.**

11 **Add the database objects (tables, stored procedures, and prepared statements) required for the project, and click OK.**

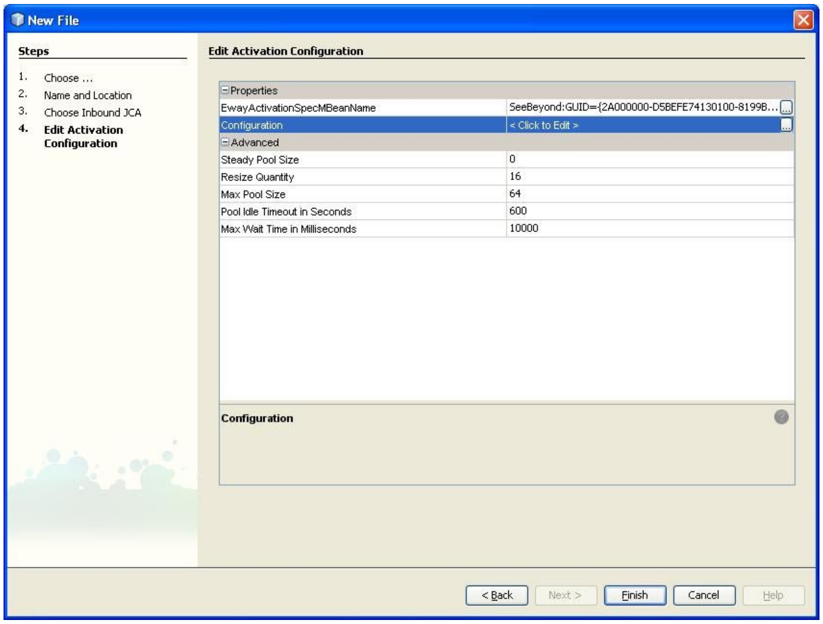
12 Enter a name for the OTD.

An OTD and a JAR file containing the code corresponding to the above selected database objects are generated. The OTD is generated in the project at the location `src/java/otds`. The JAR file is added to the project libraries.

13 **Create a JCA-driven MDB by navigating to the File menu, and selecting New Project > Enterprise.**

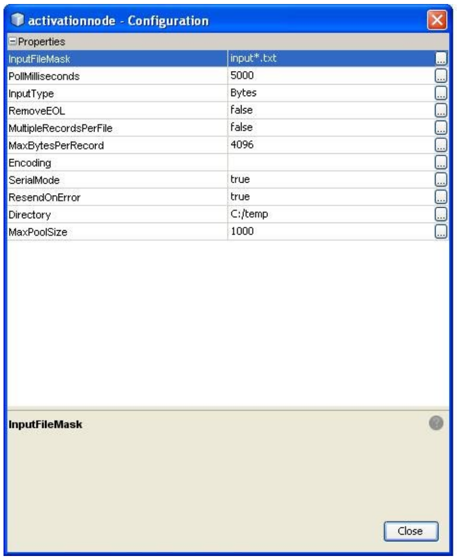
14 **Select File Inbound RAR as the Inbound JCA.**

15 **Edit the Inbound File JCA configuration parameters in the following two dialogue windows.**



⊟ Properties	
EwayActivationSpecMBeanName	SeeBeyond:GUID={2A000000-D5BEFE74130100-8199B...
Configuration	< Click to Edit >
⊟ Advanced	
Steady Pool Size	0
Resize Quantity	16
Max Pool Size	64
Pool Idle Timeout in Seconds	600
Max Wait Time in Milliseconds	10000

Configuration



InputFileMask	input*.txt
PollMilliseconds	5000
InputType	Bytes
RemoveEOL	false
MultipleRecordsPerFile	false
MaxBytesPerRecord	4096
Encoding	
SerialMode	true
ResendOnError	true
Directory	C:/temp
MaxPoolSize	1000

InputFileMask

An MDB is created.

16 **In the generated MDB, drag Oracle from the palette to invoke Oracle.**

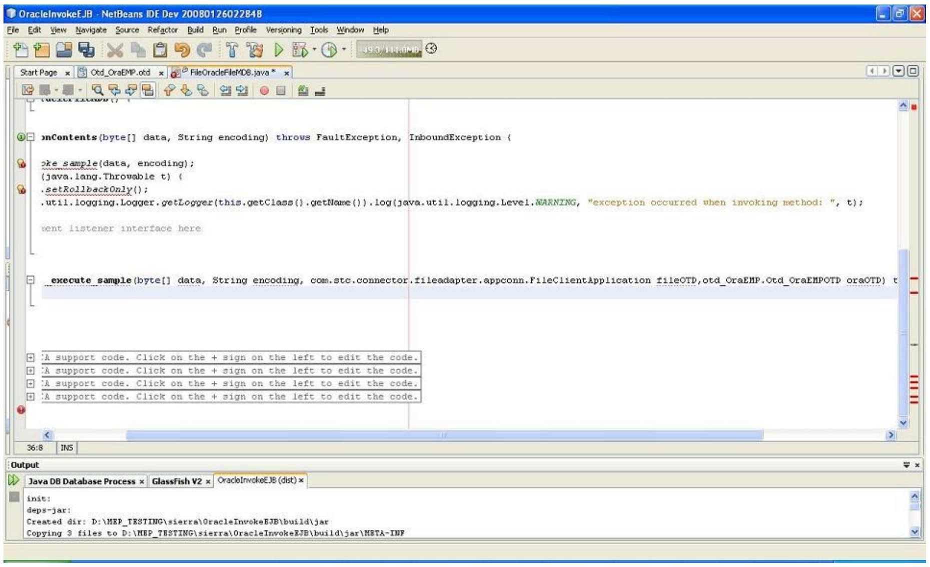
17 **Drag and drop the File JCA onto the onContents() of the MDB, and enter the parameters for the File eWay JCA Adapter declarations.**

The Resource JNDI Name should be the same as that declared in the GlassFish Application Server (for example, Resources > Connector > Connector Resources)

18 Select the Oracle OTD that is generated and click Finish.

19 Issue a select statement and iterate through the resultset, and write to File.

20 The resulting Java template is created.


OracleInvokeEJB - NetBeans IDE Dev 20080126022848
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Start Page
Otd_OraEMP.otd
FileOracleFileMDB.java *
36:8
INS
Output
Java DB Database Process
GlassFish V2
OracleInvokeEJB (dist)

```
onContents(byte[] data, String encoding) throws FaultException, InboundException {

oke sample(data, encoding);
(java.lang.Throwable t) {
.setRollbackOnly();
.util.logging.Logger.getLogger(this.getClass().getName()).log(java.util.logging.Level.WARNING, "exception occurred when invoking method: ", t);

ment listener interface here


 _execute_sample(byte[] data, String encoding, com.stc.connector.fileadapter.appconn.FileClientApplication fileOTD,otd_OraEMP.Otd_OraEMPOTD oraOTD) t



:A support code. Click on the + sign on the left to edit the code.
:A support code. Click on the + sign on the left to edit the code.
:A support code. Click on the + sign on the left to edit the code.
:A support code. Click on the + sign on the left to edit the code.
```

```
init:
deps-jar:
Created dir: D:\MEP_TESTING\sierra\OracleInvokeEJB\build\jar
Copying 3 files to D:\MEP_TESTING\sierra\OracleInvokeEJB\build\jar\META-INF
```

21 Implement the following Java code:

The following code is the output:

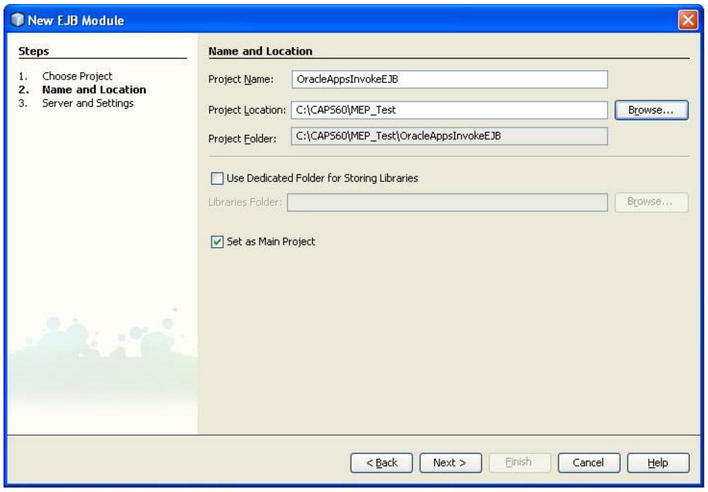
```
public void onContents(byte[] data,
                String encoding) throws FaultException, InboundException {

    try {
        _invoke_getEmpData(data, encoding);
    } catch (java.lang.Throwable t) {
    ectx.setRollbackOnly();
    java.util.logging.Logger.getLogger(this.getClass().getName()).log(java.util.
        logging.Level.WARNING,
    "exception occurred when invoking method: ", t);
    }

    try {
        _invoke_writeEmpData(data, encoding);
    } catch (java.lang.Throwable t) {
        ectx.setRollbackOnly();
        java.util.logging.Logger.getLogger(this.getClass().getName()).log(java.util.
            logging.Level.WARNING,
        "exception occurred when invoking method: ", t);
        }
   }
    private String _execute_getEmpData(byte[] data, String encoding,
            otd_emp1.Otd_emp1OTD oraOTD) throws java.lang.Exception {
    //select all records from EMP table using the method select available
    //on the OTD
    oraOTD.getEMP().select("");
    //Iterate thru the resultset and write to File.
    while(oraOTD.getEMP().next()) {
          enameValues = oraOTD.getEMP().getENAME() + ", " + enameValues;
    }

}

    private void _execute_writeEmpData(byte[] data, String encoding,
                    com.stc.connector.fileadapter.appconn.FileClientApplication fileOTD)
                        throws java.lang.Exception {

       fileOTD.setText(new String(data));
    fileOTD.write();
     }
```

Using the Oracle Applications Wizard and JCA Adapter Tooling with an EJB Project

The following task outlines the steps need to implement a specific EJB project using the Oracle Applications Adapter Wizard. The results are written to an output file.

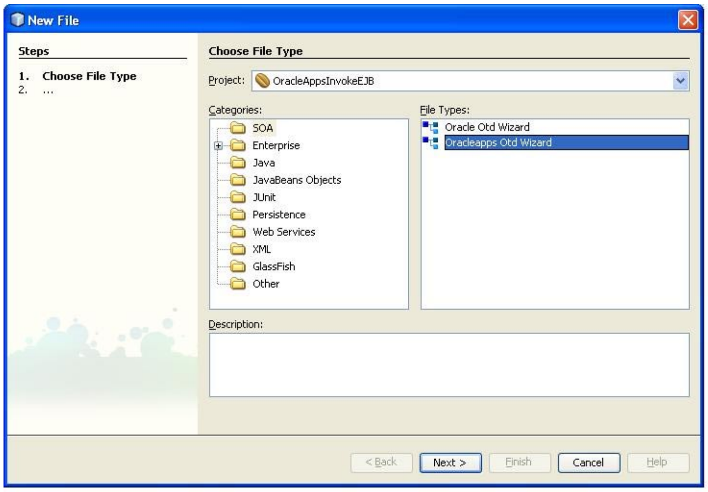
▼ To implement the Oracle Applications JCA Adapter with an EJB Project

1. **Create the required connector or connectors in the Application Server. In this particular example, create a connector for the File eWay and another for the Oracle Applications eWay.**
2. **Navigate to the Application Server Admin Console at** `htt://localhost:4848`**.**
3. **Configure the parameters for your connectors under Resources > Connector > Connector Connection Pool and Resources > Connector > Connector Resources.**
4. **Launch NetBeans.**
5. **From the File menu, select Enterprise > EJB Module.**
6. **Create a new EJB project OracleAppsInvokeEJB**



Project Name: OracleAppsInvokeEJB
Project Location: C:\CAPS60\MEP_Test
Project Folder: C:\CAPS60\MEP_Test\OracleAppsInvokeEJB
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Set as Main Project

7 **Launch the Oracle wizard by right—clicking on the OracleAppsInvokeEJB project and selecting New.**

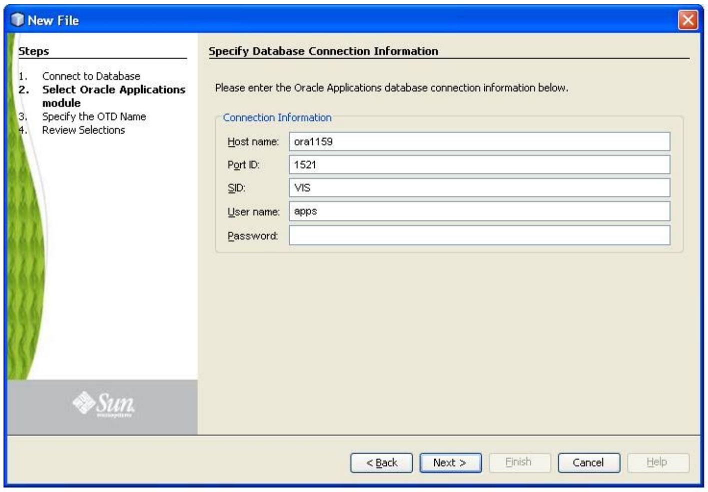


Project: OracleAppsInvokeEJB
Categories: SOA, Enterprise, Java, JavaBeans Objects, JUnit, Persistence, Web Services, XML, GlassFish, Other
File Types: Oracle Otd Wizard, Oracleapps Otd Wizard
Description:

8 **Select Other > SOA > Oracleapps OTD Wizard.**

9 **Click Next.**

10 Enter the Oracle Applications database connection information, and click Next.



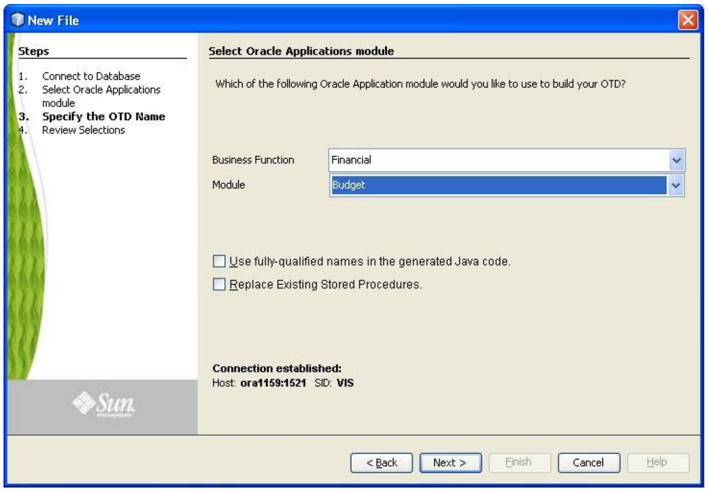
Specify Database Connection Information

Please enter the Oracle Applications database connection information below.

Connection Information

Host name: ora1159
Port ID: 1521
SID: VIS
User name: apps
Password:

11 Select the Oracle Applications Business Function and Module for the project, and click Next.



Select Oracle Applications module

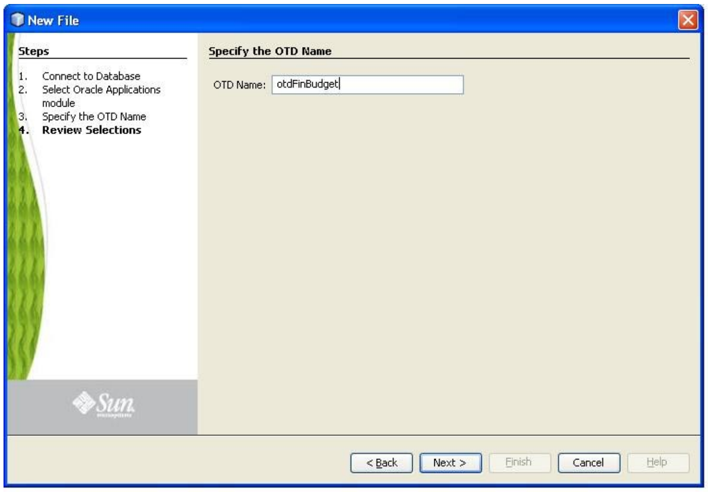
Which of the following Oracle Application module would you like to use to build your OTD?

Business Function: Financial
Module: Budget

☐ Use fully-qualified names in the generated Java code.
☐ Replace Existing Stored Procedures.

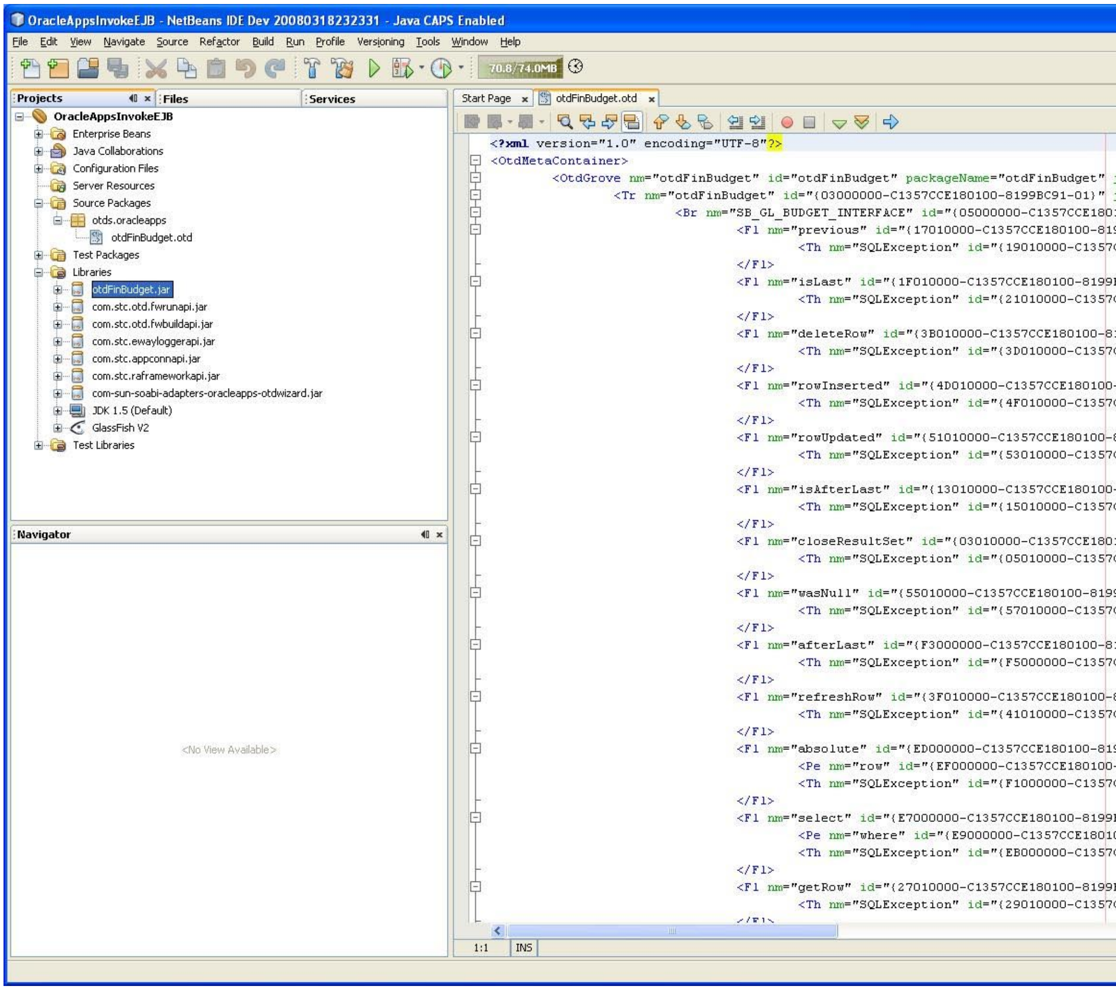
Connection established:
Host: **ora1159:1521** SID: **VIS**

12 Enter a name for the OTD.



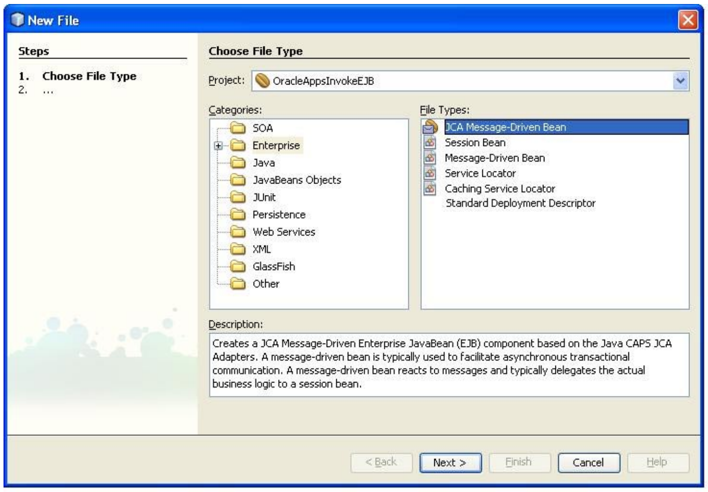
OTD Name: otdFinBudget

An OTD and a JAR file containing the code corresponding to the above selected database objects are generated. The OTD is generated in the project at the location `src/java/otds`. The JAR file is added to the project libraries.



```
<?xml version="1.0" encoding="UTF-8"?>
<OtdMetaContainer>
        <OtdGrove nm="otdFinBudget" id="otdFinBudget" packageName="otdFinBudget"
                <Tr nm="otdFinBudget" id="{03000000-C1357CCE180100-8199BC91-01}"
                        <Br nm="SB_GL_BUDGET_INTERFACE" id="{05000000-C1357CCE180
                                <Fl nm="previous" id="{17010000-C1357CCE180100-81
                                        <Th nm="SQLException" id="{19010000-C1357
                                </Fl>
                                <Fl nm="isLast" id="{1F010000-C1357CCE180100-8199
                                        <Th nm="SQLException" id="{21010000-C1357
                                </Fl>
                                <Fl nm="deleteRow" id="{3B010000-C1357CCE180100-8
                                        <Th nm="SQLException" id="{3D010000-C1357
                                </Fl>
                                <Fl nm="rowInserted" id="{4D010000-C1357CCE180100-
                                        <Th nm="SQLException" id="{4F010000-C1357
                                </Fl>
                                <Fl nm="rowUpdated" id="{51010000-C1357CCE180100-
                                        <Th nm="SQLException" id="{53010000-C1357
                                </Fl>
                                <Fl nm="isAfterLast" id="{13010000-C1357CCE180100-
                                        <Th nm="SQLException" id="{15010000-C1357
                                </Fl>
                                <Fl nm="closeResultSet" id="{03010000-C1357CCE180
                                        <Th nm="SQLException" id="{05010000-C1357
                                </Fl>
                                <Fl nm="wasNull" id="{55010000-C1357CCE180100-819
                                        <Th nm="SQLException" id="{57010000-C1357
                                </Fl>
                                <Fl nm="afterLast" id="{F3000000-C1357CCE180100-8
                                        <Th nm="SQLException" id="{F5000000-C1357
                                </Fl>
                                <Fl nm="refreshRow" id="{3F010000-C1357CCE180100-
                                        <Th nm="SQLException" id="{41010000-C1357
                                </Fl>
                                <Fl nm="absolute" id="{ED000000-C1357CCE180100-81
                                        <Pe nm="row" id="{EF000000-C1357CCE180100-
                                        <Th nm="SQLException" id="{F1000000-C1357
                                </Fl>
                                <Fl nm="select" id="{E7000000-C1357CCE180100-8199
                                        <Pe nm="where" id="{E9000000-C1357CCE1801
                                        <Th nm="SQLException" id="{EB000000-C1357
                                </Fl>
                                <Fl nm="getRow" id="{27010000-C1357CCE180100-8199
                                        <Th nm="SQLException" id="{29010000-C1357
                                </Fl>
```

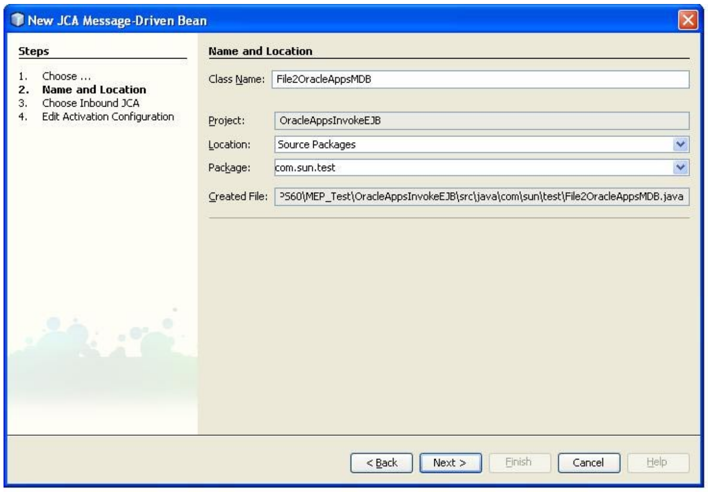
13 Create a JCA-driven MDB by navigating to the File menu, and selecting New Project > Enterprise.



Project: OracleAppsInvokeEJB

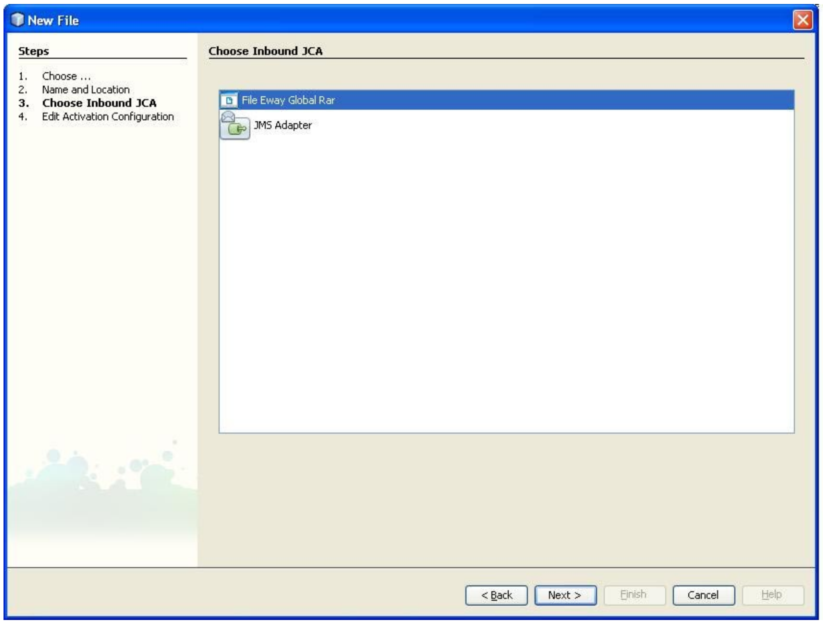
Description:
Creates a JCA Message-Driven Enterprise JavaBean (EJB) component based on the Java CAPS JCA Adapters. A message-driven bean is typically used to facilitate asynchronous transactional communication. A message-driven bean reacts to messages and typically delegates the actual business logic to a session bean.

14 **Enter the required information for the JCA-driven MDB.**

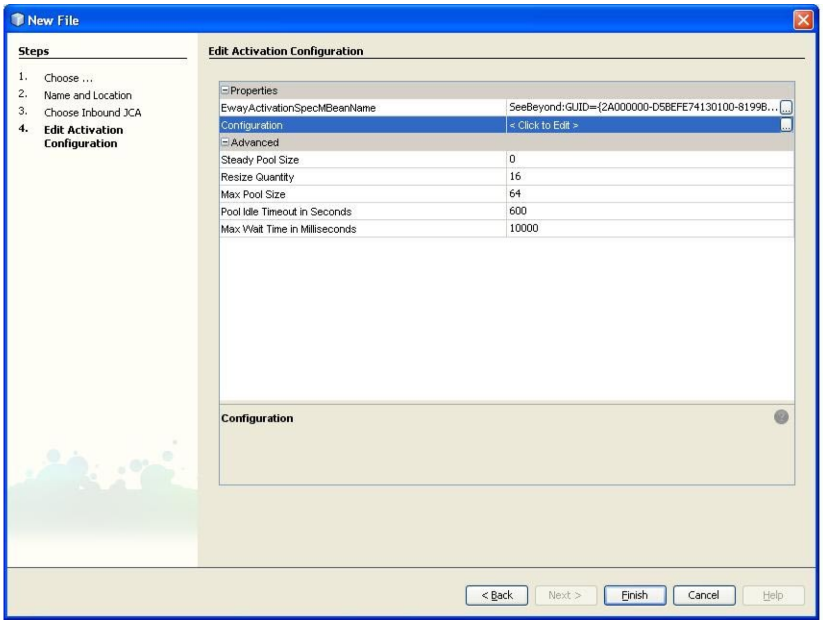


Class Name: File2OracleAppsMDB
Project: OracleAppsInvokeEJB
Location: Source Packages
Package: com.sun.test
Created File: ?S60\MEP_Test\OracleAppsInvokeEJB\src\java\com\sun\test\File2OracleAppsMDB.java

15 **Select File Inbound RAR as the Inbound JCA.**

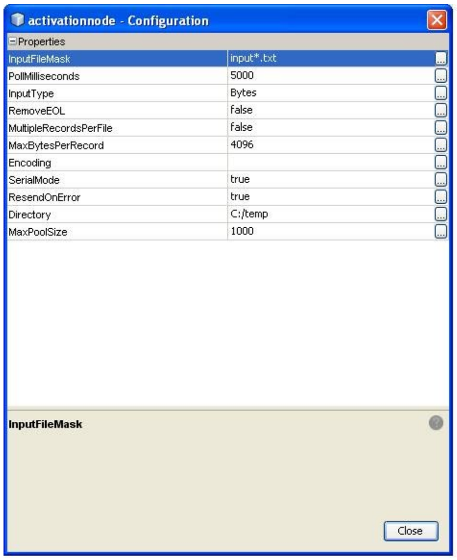


16 **Edit the Inbound File JCA configuration parameters in the following two dialogue windows.**



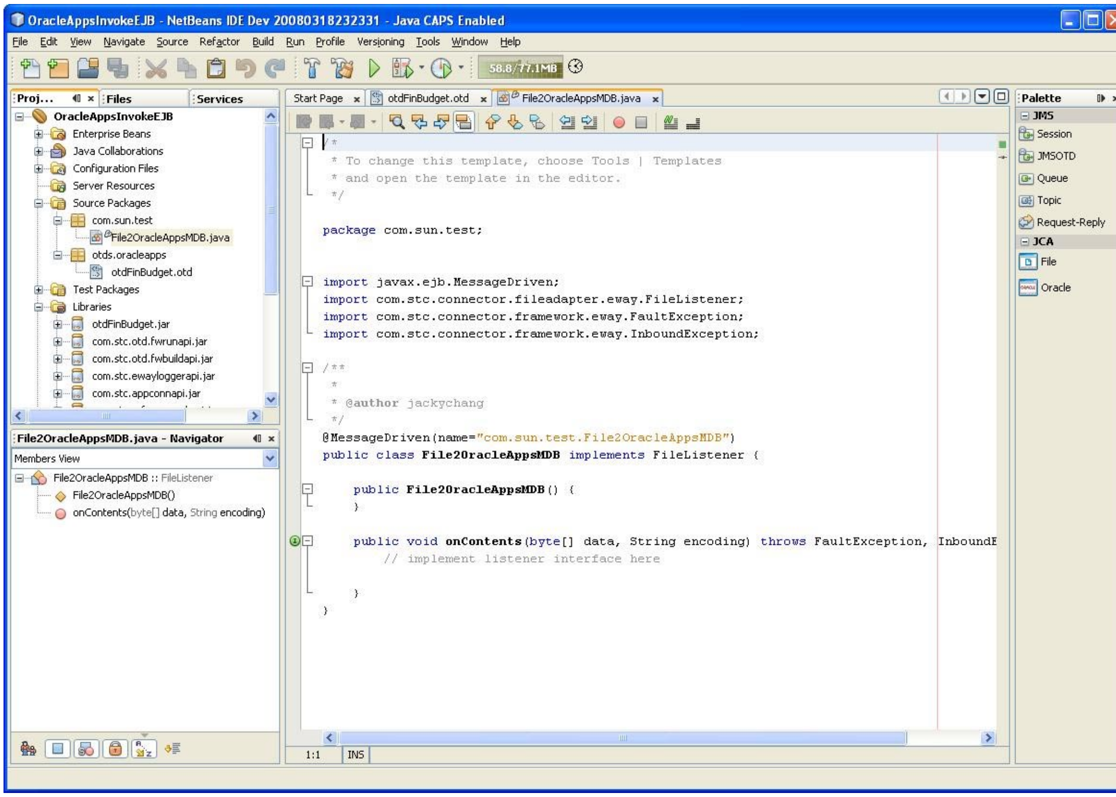
Properties	
EwayActivationSpecMBeanName	SeeBeyond:GUID={2A000000-D5BEFE74130100-8199B...
Configuration	< Click to Edit >
Advanced	
Steady Pool Size	0
Resize Quantity	16
Max Pool Size	64
Pool Idle Timeout in Seconds	600
Max Wait Time in Milliseconds	10000

Configuration



InputFileMask	input*.txt
PollMilliseconds	5000
InputType	Bytes
RemoveEOL	false
MultipleRecordsPerFile	false
MaxBytesPerRecord	4096
Encoding	
SerialMode	true
ResendOnError	true
Directory	C:/temp
MaxPoolSize	1000

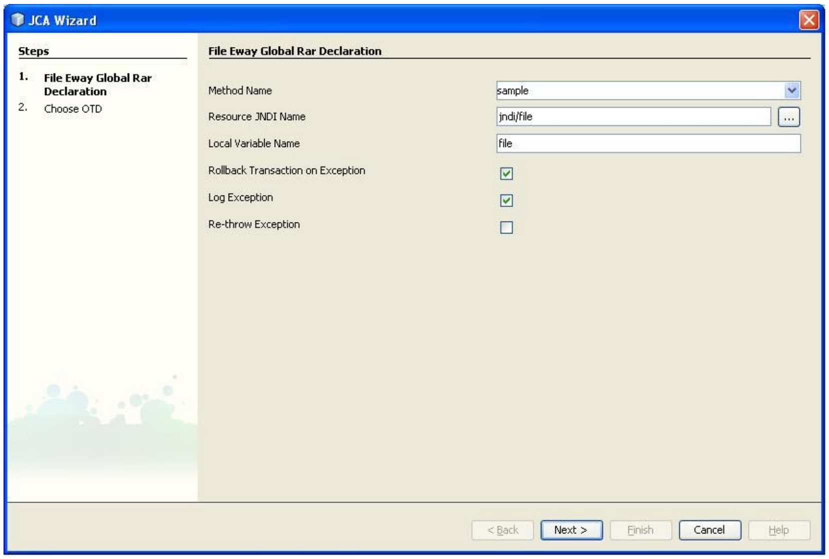
An MDB is created.



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.sun.test;


import javax.ejb.MessageDriven;
import com.stc.connector.fileadapter.eway.FileListener;
import com.stc.connector.framework.eway.FaultException;
import com.stc.connector.framework.eway.InboundException;

/**
 *
 * @author jackychang
 */
@MessageDriven(name="com.sun.test.File2OracleAppsMDB")
public class File2OracleAppsMDB implements FileListener {

    public File2OracleAppsMDB() {
    }

    public void onContents(byte[] data, String encoding) throws FaultException, InboundE
        // implement listener interface here

    }
}
```

17 **In the generated MDB, drag Oracle Applications from the palette to invoke Oracle Applications.**

18 **Drag and drop the File JCA onto the onContents() of the MDB, and enter the parameters for the File eWay JCA Adapter declarations.**

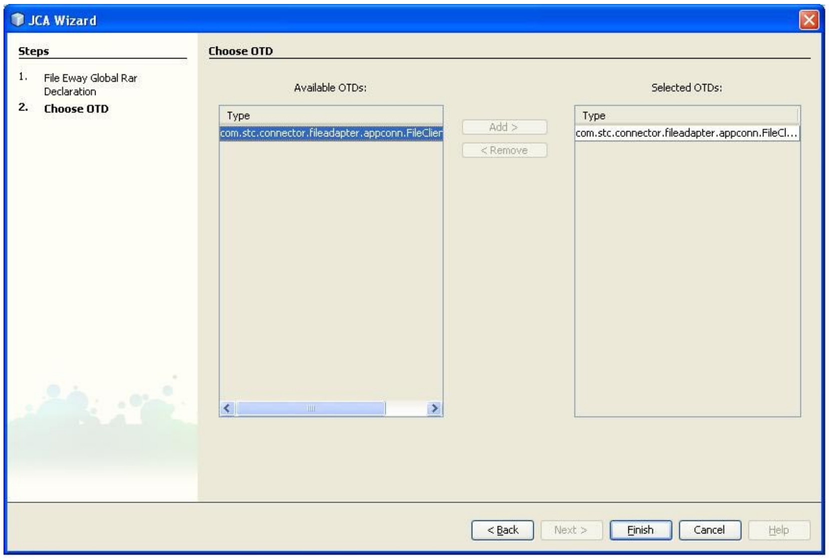


File Eway Global Rar Declaration

Method Name	sample
Resource JNDI Name	jndi/file
Local Variable Name	file
Rollback Transaction on Exception	✓
Log Exception	✓
Re-throw Exception	

The Resource JNDI Name should be the same as that declared in the GlassFish Application Server (for example, Resources > Connector > Connector Resources)

19 Select the File OTD that is generated and click Finish.



Choose OTD

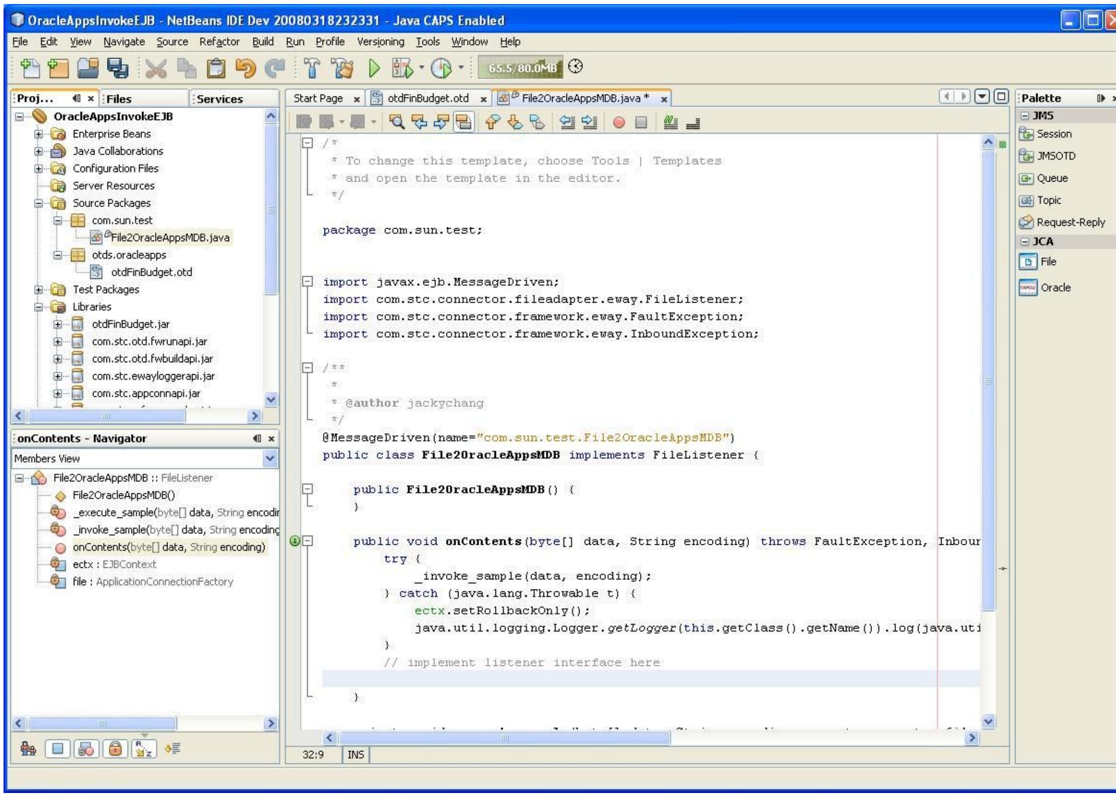
Available OTDs:

Type
com.stc.connector.fileadapter.appconn.FileClier

Selected OTDs:

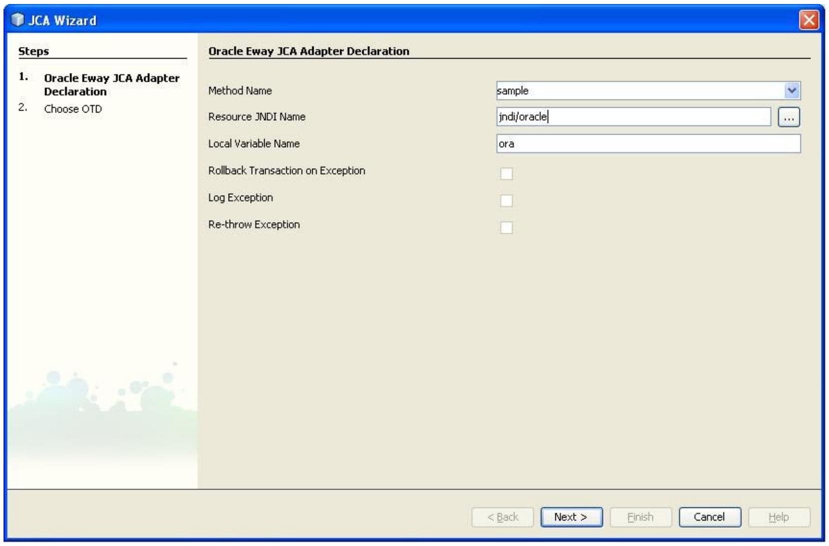
Type
com.stc.connector.fileadapter.appconn.FileCl...

20 The resulting Java template is created.



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.sun.test;


import javax.ejb.MessageDriven;
import com.stc.connector.fileadapter.eway.FileListener;
import com.stc.connector.framework.eway.FaultException;
import com.stc.connector.framework.eway.InboundException;

/**
 *
 * @author jackychang
 */
@MessageDriven(name="com.sun.test.File2OracleAppsMDB")
public class File2OracleAppsMDB implements FileListener {

    public File2OracleAppsMDB() {
    }

    public void onContents(byte[] data, String encoding) throws FaultException, Inbour
        try {
            _invoke_sample(data, encoding);
        } catch (java.lang.Throwable t) {
            ectx.setRollbackOnly();
            java.util.logging.Logger.getLogger(this.getClass().getName()).log(java.uti
        }
        // implement listener interface here

    }
```

21 **Drag and drop the Oracle Applications JCA onto the onContents() of the MDB, and enter the parameters for the Oracle Applications JCA Adapter declarations.**

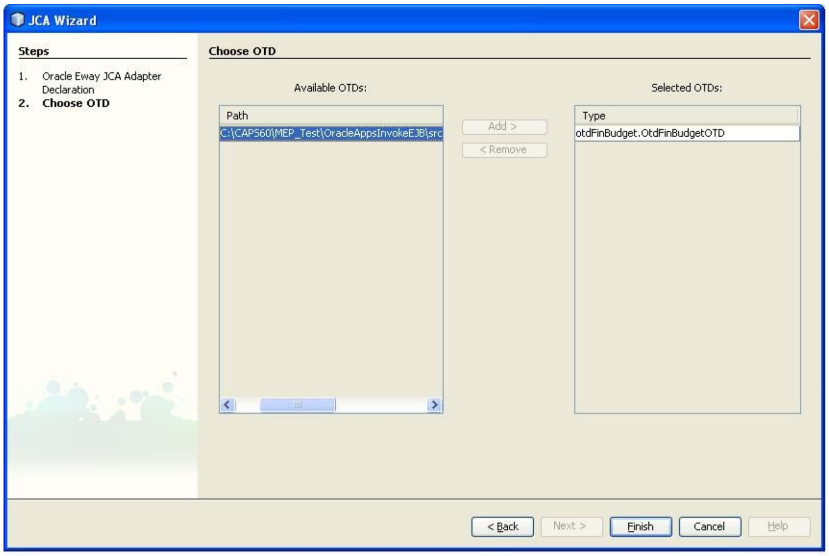

JCA Wizard
Steps
1. Oracle Eway JCA Adapter Declaration
2. Choose OTD
Oracle Eway JCA Adapter Declaration

Method Name	sample
Resource JNDI Name	jndi/oracle
Local Variable Name	ora
Rollback Transaction on Exception	☐
Log Exception	☐
Re-throw Exception	☐

< Back
Next >
Finish
Cancel
Help

The Resource JNDI Name should be the same as that declared in the GlassFish Application Server (for example, Resources > Connector > Connector Resources)

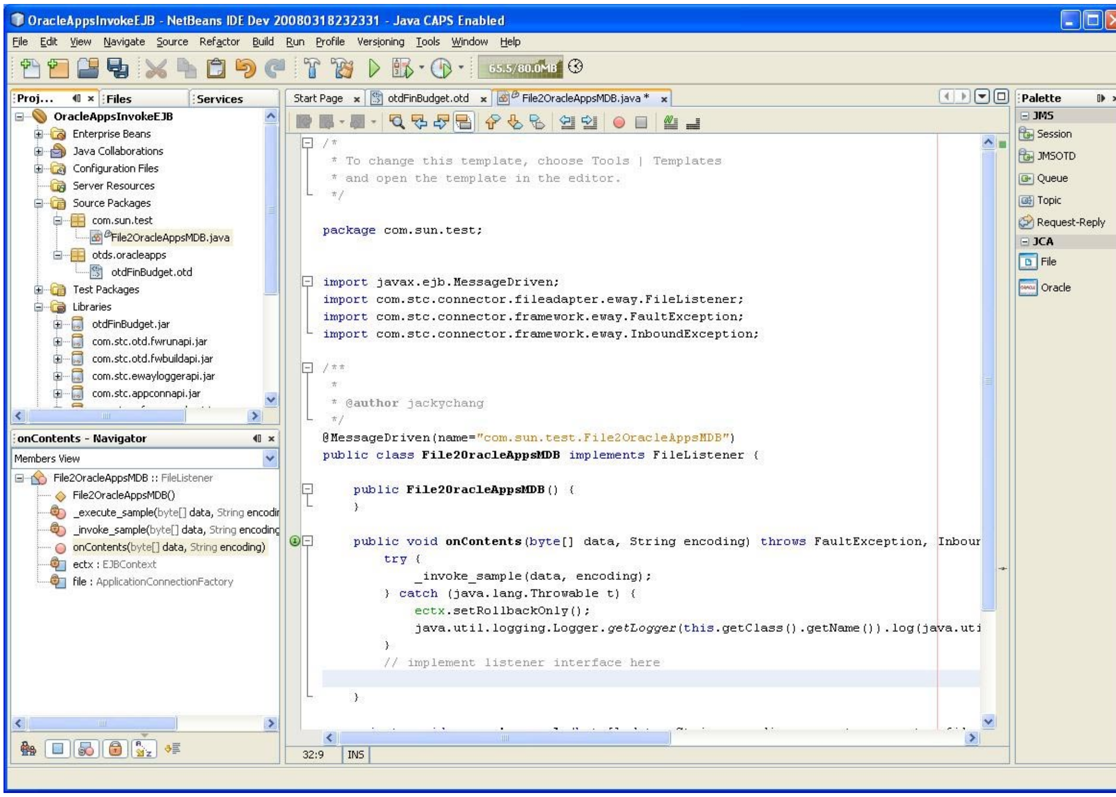
22 Select the Oracle Applications OTD that is generated and click Finish.



Available OTDs: C:\CAPS60\MEP_Test\OracleAppsInvokeEJB\src

Selected OTDs: otdFinBudget.OtdFinBudgetOTD

23 The resulting Java template is created.



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package com.sun.test;


import javax.ejb.MessageDriven;
import com.stc.connector.fileadapter.eway.FileListener;
import com.stc.connector.framework.eway.FaultException;
import com.stc.connector.framework.eway.InboundException;

/**
 *
 * @author jackychang
 */
@MessageDriven(name="com.sun.test.File2OracleAppsMDB")
public class File2OracleAppsMDB implements FileListener {

    public File2OracleAppsMDB() {
    }

    public void onContents(byte[] data, String encoding) throws FaultException, Inbour
        try {
            _invoke_sample(data, encoding);
        } catch (java.lang.Throwable t) {
            ectx.setRollbackOnly();
            java.util.logging.Logger.getLogger(this.getClass().getName()).log(java.uti
        }
        // implement listener interface here

    }
```

24 Implement the following Java code to the _execute_sample method:

```
private void _execute_sample(byte[] data, String encoding,
com.stc.connector.fileadapter.appconn.FileClientApplication fileOTD,otdFinBudget.
OtdFinBudgetOTD oraOTD) throws java.lang.Exception {

        int counter = 0;

        int budget_count = 0;

        int budget_error_count = 0;

        int request_ID = -1;

        String org_ID = "204";

        // Set eWayID

        String eWayID = "B";
        // Set GroupID

        String GroupID = "B";

        // Set ObjectID

        String ObjectID = "B";

        fileOTD.setText("1. =====> Insert record into Budget SB Staging table ..");

        fileOTD.write();

        oraOTD.getSB_GL_BUDGET_INTERFACE().insert();

        oraOTD.getSB_GL_BUDGET_INTERFACE().setBUDGET_ENTITY_ID
(new java.math.BigDecimal("1000"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setBUDGET_ENTITY_NAME("RP_Opnames");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setBUDGET_VERSION_ID
(new java.math.BigDecimal("1002"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setBUDGET_NAME("CORPORATE 1996");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setCURRENCY_CODE("USD");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setFISCAL_YEAR
(new java.math.BigDecimal("1996"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setUPDATE_LOGIC_TYPE("A");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSET_OF_BOOKS_ID
(new java.math.BigDecimal("1"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD_TYPE("Month");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setDR_FLAG("Y");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setACCOUNT_TYPE("A");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD1_AMOUNT
(new java.math.BigDecimal("100"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD2_AMOUNT
(new java.math.BigDecimal("200"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD3_AMOUNT
```

```
(new java.math.BigDecimal("300"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD4_AMOUNT
(new java.math.BigDecimal("400"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD5_AMOUNT
(new java.math.BigDecimal("500"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD6_AMOUNT
(new java.math.BigDecimal("600"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD7_AMOUNT
(new java.math.BigDecimal("700"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD8_AMOUNT
(new java.math.BigDecimal("800"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD9_AMOUNT
(new java.math.BigDecimal("900"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD10_AMOUNT
(new java.math.BigDecimal("1000"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD11_AMOUNT
(new java.math.BigDecimal("1100"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD12_AMOUNT
(new java.math.BigDecimal("1200"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setCODE_COMBINATION_ID
(new java.math.BigDecimal("17378"));

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT1("01");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT2("760");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT3("7420");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT4("0000");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT5("000");

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSB_EWAY_ID(eWayID);

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSB_GROUP_ID(GroupID);

        oraOTD.getSB_GL_BUDGET_INTERFACE().setSB_OBJECT_ID(ObjectID);
        oraOTD.getSB_GL_BUDGET_INTERFACE().insertRow();

        fileOTD.setText("2. =====> Counting record in Budget staging table ...");

        fileOTD.write();

        oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_GROUP_ID(GroupID);

        oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_OBJECT_ID(ObjectID);

        oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_EWAY_ID(eWayID);

        oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_CODE("A");

        oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().execute();

        budget_count = oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().
getRETURN_VALUE1().intValue();

        fileOTD.setText("     returned Budget Count  =
".concat( Integer.toString( budget_count )));

        fileOTD.write();

        fileOTD.setText("\nDone with Step 2");

        fileOTD.write();

    }
```

Using the Oracle Applications Object Type Definition

Each of the Oracle Applications Object Type Definition modules is divided into submodules, representing a logical division of workflow. Each submodule uses at least one corresponding set of tables and scripts. For a high level hierarchy of the Manufacturing and Financial modules and further details on their submodules, see "Building an Oracle Applications Custom Pre-Validation Package" in *Designing with Application Adapters*

Once an Oracle Applications OTD is generated, all of the staging tables and stored procedures for the corresponding submodules are generated in the Oracle database. These staging tables and procedures can be invoked through the OTD.

In general, a user can take the following steps to invoke operations in a database:

1. Insert data into the staging table.
   - Get a count of the data in the staging table.

2. Run a pre-validation script to check data integrity.
   - Count invalid records, if any.
   - Find error codes, if any.
   - Count the valid records in the staging table.
3. Move the valid records to the Open Interface table.
4. Calling the initialize function.
5. Calling the concurrent manager.
6. Getting the Request status.
7. Cleaning the staging table.

For what follows we will use the Financial –> General Ledger —> Budget module as a sample. Assume the Oracle Applications OTD is called **oraOTD**.

1. Insert data into the staging table:

```
oraOTD.getSB_GL_BUDGET_INTERFACE().insert();

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setBUDGET_ENTITY_ID(new java.math.BigDecimal("1000"));

oraOTD.getSB_GL_BUDGET_INTERFACE().setBUDGET_ENTITY_NAME("RP_Opnames");

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setBUDGET_VERSION_ID(new java.math.BigDecimal("1002"));

oraOTD.getSB_GL_BUDGET_INTERFACE().setBUDGET_NAME("CORPORATE 1996");

oraOTD.getSB_GL_BUDGET_INTERFACE().setCURRENCY_CODE("USD");

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setFISCAL_YEAR(new java.math.BigDecimal("1996"));

oraOTD.getSB_GL_BUDGET_INTERFACE().setUPDATE_LOGIC_TYPE("A");

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setSET_OF_BOOKS_ID(new java.math.BigDecimal("1"));

oraOTD.getSB_GL_BUDGET_INTERFACE().setPERIOD_TYPE("Month");

oraOTD.getSB_GL_BUDGET_INTERFACE().setDR_FLAG("Y");

oraOTD.getSB_GL_BUDGET_INTERFACE().setACCOUNT_TYPE("A");

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD1_AMOUNT(new java.math.BigDecimal("100"));
```

```
oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD2_AMOUNT(new java.math.BigDecimal("200"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD3_AMOUNT(new java.math.BigDecimal("300"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD4_AMOUNT(new java.math.BigDecimal("400"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD5_AMOUNT(new java.math.BigDecimal("500"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD6_AMOUNT(new java.math.BigDecimal("600"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD7_AMOUNT(new java.math.BigDecimal("700"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD8_AMOUNT(new java.math.BigDecimal("800"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD9_AMOUNT(new java.math.BigDecimal("900"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD10_AMOUNT(new java.math.BigDecimal("1000"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD11_AMOUNT(new java.math.BigDecimal("1100"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setPERIOD12_AMOUNT(new java.math.BigDecimal("1200"));

oraOTD.getSB_GL_BUDGET_INTERFACE().
        setCODE_COMBINATION_ID(new java.math.BigDecimal("17378"));

oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT1("01");

oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT2("760");

oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT3("7420");

oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT4("0000");

oraOTD.getSB_GL_BUDGET_INTERFACE().setSEGMENT5("000");

oraOTD.getSB_GL_BUDGET_INTERFACE().setSB_EWAY_ID(eWayID);
oraOTD.getSB_GL_BUDGET_INTERFACE().setSB_GROUP_ID(GroupID);

oraOTD.getSB_GL_BUDGET_INTERFACE().setSB_OBJECT_ID(ObjectID);

oraOTD.getSB_GL_BUDGET_INTERFACE().insertRow();
```

Get a count of the data in the staging table.

```
oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_GROUP_ID(GroupID);

oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_OBJECT_ID(ObjectID);

oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_EWAY_ID(eWayID);

oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_CODE("A");

oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().execute();

budget_count = oraOTD.getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().
        getRETURN_VALUE1().intValue();
```

2. Run a pre-validation script to check data integrity.

```
oraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_VALIDATE_GL_BUDGET_PKG_VALIDATE().setP_GROUP_ID( GroupID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_VALIDATE_GL_BUDGET_PKG_VALIDATE().setP_OBJECT_ID( ObjectID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_VALIDATE_GL_BUDGET_PKG_VALIDATE().setP_EWAY_ID( eWayID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_VALIDATE_GL_BUDGET_PKG_VALIDATE().execute();
```

- Count invalid records, if any.

```
OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_GROUP_ID( GroupID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_OBJECT_ID( ObjectID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_EWAY_ID( eWayID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_CODE( "F" );
```

```
OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().execute();

                    budget_error_count = OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().getRETURN_VALUE1().intValue();
```

- Find error codes, if any.

```
if (budget_error_count > 0) {

OraOTD.getSB_GL_BUDGET_INTERFACE().
qqgetGET_INVALID_ROWS_GL_BUDGET_INT().setEWAY_ID( eWayID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
qqgetGET_INVALID_ROWS_GL_BUDGET_INT().setGROUP_ID( GroupID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
qqgetGET_INVALID_ROWS_GL_BUDGET_INT().setOBJECT_ID( ObjectID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
qqgetGET_INVALID_ROWS_GL_BUDGET_INT().executeQuery();

                        if (OraOTD.getSB_GL_BUDGET_INTERFACE().
        getGET_INVALID_ROWS_GL_BUDGET_INT().resultsAvailable()) {

                            while (OraOTD.getSB_GL_BUDGET_INTERFACE().
        getGET_INVALID_ROWS_GL_BUDGET_INT().
        get$GET_INVALID_ROWS_GL_BUDGET_INTResults().next()) {

                                JMS_1.sendText( "     SB_ERROR_CODE: ".
        concat( OraOTD.getSB_GL_BUDGET_INTERFACE().
        getGET_INVALID_ROWS_GL_BUDGET_INT().
        get$GET_INVALID_ROWS_GL_BUDGET_INTResults().
        getSB_ERROR_CODE() ).concat( "     SB_ERROR_MESSAGE: ".
        concat( OraOTD.getSB_GL_BUDGET_INTERFACE().
        getGET_INVALID_ROWS_GL_BUDGET_INT().
        get$GET_INVALID_ROWS_GL_BUDGET_INTResults().getSB_ERROR_MESSAGE() ) ) );

                            }

                        }

                    }
```

- Count the valid records in the staging table.

```
OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_GROUP_ID( GroupID );
```

```
OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_OBJECT_ID( ObjectID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_EWAY_ID( eWayID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().setP_CODE( "P" );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().execute();

                    budget_count = OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_FN_CNT_GL_BUDGET_INT().getRETURN_VALUE1().intValue();
```

3. Move the valid records to the Open Interface table.

```
OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_MOV_GL_BUDGET_INT().setP_GROUP_ID( GroupID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_MOV_GL_BUDGET_INT().setP_OBJECT_ID( ObjectID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_MOV_GL_BUDGET_INT().setP_EWAY_ID( eWayID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_MOV_GL_BUDGET_INT().setP_CODE( "P" );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_MOV_GL_BUDGET_INT().execute();
```

4. Calling the initialize function.

```
OraOTD.getSB_INITIALIZE_INITIALIZE_PROFILE().setP_ORGID( org_ID );

OraOTD.getSB_INITIALIZE_INITIALIZE_PROFILE().execute();
```

5. Calling the concurrent manager.

```
OraOTD.getFN_REQUEST_BUDGET().setP_USER_NAME( "OPERATIONS" );

OraOTD.getFN_REQUEST_BUDGET().setP_RESPONSIBILITY( "GENERAL_LEDGER_SUPER_USER" );

OraOTD.getFN_REQUEST_BUDGET().setP_ORGID( org_ID );

OraOTD.getFN_REQUEST_BUDGET().setAPPLICATION( "SQLGL" );

OraOTD.getFN_REQUEST_BUDGET().setPROGRAM( "GLBBSU" );
OraOTD.getFN_REQUEST_BUDGET().setDESCRIPTION( "Budget Spreadsheet Upload" );

OraOTD.getFN_REQUEST_BUDGET().execute();

request_ID = OraOTD.getFN_REQUEST_BUDGET().getRETURN_VALUE1().intValue();

OraOTD.commit();
```

6. Getting the Request status.

```
if (request_ID > 0) {

    OraOTD.getFN_REQUEST_STATUS().
        setINP_REQUEST_ID( new java.math.BigDecimal( Integer.toString( request_ID ) ) );

    OraOTD.getFN_REQUEST_STATUS().
        setINP_INTERVAL_SEC( new java.math.BigDecimal( "15" ) );

    OraOTD.getFN_REQUEST_STATUS().
        setINP_MAXIMUM_SEC( new java.math.BigDecimal( "30" ) );

OraOTD.getFN_REQUEST_STATUS().execute();

Result_status = OraOTD.getFN_REQUEST_STATUS().getRETURN_VALUE1() );
```

7. Cleaning the staging table.

```
OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_DEL_GL_BUDGET_INT().setP_GROUP_ID( GroupID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_DEL_GL_BUDGET_INT().setP_OBJECT_ID( ObjectID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_DEL_GL_BUDGET_INT().setP_EWAY_ID( eWayID );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_DEL_GL_BUDGET_INT().setP_CODE( "A" );

OraOTD.getSB_GL_BUDGET_INTERFACE().
        getSB_BUDGET_UTILS_PKG_SP_DEL_GL_BUDGET_INT().execute();
```