SUN SEEBEYOND

eWAY™ ADAPTER FOR ORACLE USER'S GUIDE

Release 5.1.3

Part Number: 820-0994

Version 20070425114450

Contents

Chapter 1

Introducing the Oracle eWay

Welcome to the *Oracle eWay Intelligent Adapter User's Guide*. This document includes information about installing, configuring, and using the Oracle eWay Intelligent Adapter.

What's in This Chapter

1.1 About Oracle Databases

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

The database has logical structures and physical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

1.2 About the Oracle eWay

The Oracle eWay enables eGate Integrator Projects to exchange data with external Oracle databases. This user's guide describes how to install and configure the Oracle eWay.

1.3 What's New in This Release

This 5.1.3 version release provides general maintenance fixes for the Oracle eWay Intelligent Adapter.

What's New in Version 5.1.3

- **Added support:** Supports automatic deployment of EAR files to WebLogic Application Server (version 9.1).

What's New in Version 5.1.2

- This is a maintenance release. No new features.

What's New in Version 5.1.1

- This is a maintenance release. No new features.

New for Version 5.1.0

- **Version control:** An enhanced version control system allows you to effectively manage changes to the eWay components.
- **Multiple Drag-and-Drop Component Mapping from the Deployment Editor:** The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- **Support for Runtime LDAP Configuration:** Configuration properties now support LDAP key values.
- **Connection Retry Support:** Allows you to specify the number of attempts to reconnect, and the interval between retry attempts, in the event of a connection failure.
- **Connectivity Map Generator:** Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.

Many of these features are documented further in the *Sun SeeBeyond eGate Integrator User's Guide* or the *Sun SeeBeyond eGate Integrator System Administrator Guide*.

1.4 About This Document

This guide explains how to install, configure, and operate the Sun Java Composite Application Platform Suite™ Oracle eWay Adapter, referred to as the Oracle eWay throughout this guide.

1.4.1 What's in This Document

This document includes the following chapters:

- **Chapter 1 "Introducing the Oracle eWay"**: Provides an overview of the product as well as high-level information about this document.

- **Chapter 2 "Installing the Oracle eWay"**: Describes the system requirements and provides instructions for installing the Oracle eWay.
- **Chapter 3 "Configuring the Oracle eWay"**: Provides instructions for configuring the eWay to communicate with your legacy systems.
- **Chapter 4 "Using the Oracle eWay OTD Wizard"**: Provides information about .sag files and using the Oracle wizard.
- **Chapter 5 "Using Oracle Operations"**: Provides information about database operations used in the Oracle eWay to access the Oracle database.
- **Chapter 6 "Implementing the Oracle eWay Project"**: Provides instructions for installing and running the sample Projects.
- **Appendix A "Oracle eWay Data Types"**: Provides conversions between Oracle and OTD/Java datatypes.

1.4.2 Scope

This document describes the process of installing, configuring, and running the Oracle eWay.

This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the Oracle eWay Javadoc files from the Enterprise Manager.

1.4.3 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning Sun Java Composite Application Platform Suite. This person must also understand any operating systems on which the Sun Java Composite Application Platform Suite is to be installed (Windows or UNIX) and must be thoroughly familiar with Windows-style GUI operations.

1.4.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in **bold italic**	`java -jar` **filename**`.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 9

Table 1 Text Conventions (Continued)

Text Convention	Used For	Examples
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.4.5 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.5 Related Documents

The following Sun documents provide additional information about the Sun Java Composite Application Platform Suite:

- *eGate Integrator User's Guide*
- *Sun Java* Composite Application Platform Suite *Installation Guide*

1.6 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.sun.com

1.7 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

CAPS_docsfeedback@sun.com

Chapter 2

Installing the Oracle eWay

This chapter describes how to install the Oracle eWay.

What's in This Chapter

2.1 Before You Install

Open and review the **Readme.txt** file for the Oracle eWay for any additional information or requirements, prior to installation. The **Readme.txt file** is located on the installation disk.

2.2 Installing the Oracle eWay

The Enterprise Manager, a web-based application, is used to select and upload eWays and add-on files during the installation process. The following section describes how to install the components required for this eWay.

Refer to the readme for the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

Note: *When the Repository is running on a UNIX operating system, the eWays are loaded from the Enterprise Manager running on a Windows platform connected to the Repository server using Internet Explorer.*

2.2.1 Installing the Oracle eWay on an eGate Supported System

After you have installed Core Products, do the following:

1. From the Sun Java Composite Application Platform Suite Installer, click on the **Click to install additional products** link (on the Administration tab).
2. Expand the **eWay** option.
3. From **Select Sun Java Composite Application Platform Suite Products to Install**, select the products for your **Sun Java Composite Application Platform Suite** and include the following:
   - **FileeWay** (the File eWay is used by most sample Projects)
   - **OracleeWay**
   - eInsight (must be installed to use BPEL functionality)
4. Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.
5. From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Follow this procedure for each of your products. The Installing Files window appears after the last SAR file has been selected.
6. From the **Installing Files** window, review the product list. If it is correct, Click **Install Products**. The Enterprise Manager starts the installation.
7. When your product's installation is completed, click on the prompt, "**When installation completes, click here to continue**."

To upload the Sun SeeBeyond eWay™ Adapter for Oracle User's Guide, Help file, Javadoc, Readme, and sample Projects, do the following:

A. Expand the **Documentation** option.

B. Select **OracleeWayDocs.**

C. Click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.

Adding the eWay to an Existing Sun Java Composite Application Platform Suite Installation

It is possible to add the eWay to an existing Sun Java Composite Application Platform Suite installation.

Steps required to add an eWay to an Existing CAPS installation include:

1. Complete steps 1 through 6 on **Installing the Oracle eWay on an eGate Supported System** on page 12.
2. Open the Enterprise Designer and select **Update Center** from the Tools menu. The Update Center Wizard appears.
3. For Step 1 of the wizard, simply click **Next**.

4 For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.

5 For Step 3 of the wizard, wait for the modules to download, then click **Next**.

6 The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**

When prompted, restart the IDE (Integrated Development Environment) to complete the installation.

2.3 After You Install

To use the Thick driver (for Oracle 9iR2 version), you must copy the OCI driver to:

```
<JavaCAPS51_installation>\logicalhost\is\domains\domain1\lib
```

and then restart the logicalhost (domain).

2.4 Extracting the Sample Projects and Javadocs

The Oracle eWay includes sample Projects and Javadocs. The sample Projects are designed to provide you with a basic understanding of how certain database operations are performed using the eWay, while Javadocs provide a list of classes and methods exposed in the eWay.

Steps to extract the Javadoc include:

1 Click the Documentation tab of the Suite Installer, then click the Add-ons tab.

2 Click the Oracle eWay Intelligent Adapter link. Documentation for the Oracle eWay appears in the right pane.

3 Click the icon next to **Javadoc** and extract the ZIP file.

4 Open the index.html file to view the Javadoc.

Steps to extract the Sample Projects include:

1 Click the Documentation tab of the Suite Installer, then click the Add-ons tab.

2 Click the Oracle eWay Intelligent Adapter link. Documentation for the Oracle eWay appears in the right pane.

3 Click the icon next to **Sample Projects** and extract the ZIP file. Note that the **Oracle_eWay_Sample.zip** file contains two additional ZIP files for each sample Project.

Refer to **Importing a Sample Project** on page 69 for instructions on importing the sample Project into your repository via the Enterprise Designer.

2.5 Installing Enterprise Manager eWay Plug-Ins

The **Sun SeeBeyond Enterprise Manager** is a Web-based interface that allows you to monitor and manage your Composite Application Platform Suite applications. The Enterprise Manager requires an eWay specific "plug-in" for each different eWay you install. These plug-ins enable the Enterprise Manager to target specific alert codes for each eWay type, as well as to start and stop the inbound eWays.

The *Sun Java Composite Application Platform Suite Installation Guide* describes how to install Enterprise Manager. The *Sun SeeBeyond eGate Integrator System Administration Guide* describes how to monitor servers, Services, logs, and alerts using the Enterprise Manager and the command-line client.

The **eWay Enterprise Manager plug-ins** are available from the **List of Components to Download** under the Composite Application Platform Suite Installer's **DOWNLOADS** tab.

There are two ways to add the eWay Enterprise Manager plug-ins:

- From the **Sun SeeBeyond Enterprise Manager**
- From the **Sun Java Composite Application Platform Suite Installer**

To add plug-ins from the Enterprise Manager

1. From the **Enterprise Manager**'s Explorer toolbar, click **configuration**.
2. Click the **Web Applications Manager** tab, go to the **Auto-Install from Repository** tab, and connect to your Repository.
3. Select the application plug-ins you require, and click **Install**. The application plug-ins are installed and deployed.

To add plug-ins from the Sun Java Composite Application Platform Suite Installer

1. From the **Sun Java Composite Application Platform Suite Installer's Download tab**, select the Plug-Ins you require and save them to a temporary directory.
2. From the **Enterprise Manager**'s Explorer toolbar, click **configuration**.
3. Click the **Web Applications Manager** tab and go to the **Manage Applications** sub-tab.
4. Browse for and select the WAR file for the application plug-in that you downloaded, and click **Deploy**. The plug-in is installed and deployed.

Viewing Alert Codes

You can view and delete alerts using the Enterprise Manager. An alert is triggered when a specified condition occurs in a Project component. The purpose of the alert is to warn the administrator or user that a condition has occurred.

To View the eWay Alert Codes

1. Add the eWay Enterprise Manager plug-in for this eWay.
2. From the **Enterprise Manager**'s Explorer toolbar, click **configuration**.

3 Click the **Web Applications Manager** tab and go to the **Manage Alert Codes** tab.

4 Browse for and select the Alert Properties File for the application plug-in that you added. The Alert Properties Files are located in the **alertcodes** folder of your Sun Java Composite Application Platform Suite installation directory.

5 Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of available alert codes is displayed in Table 2.

Table 2 Alert Codes for the Oracle eWay

Alert Code\Description	Description Details	User Actions
DBCOMMON-CONNECT-FAILED000001=Failed to connect to database {0} on host {1}. Reason: The Pooled connection could not be allocated: [{2}]	Occurs during the initial database connection.	▪ Database is down; start your database. ▪ External configuration information is invalid. You may need to verify the following: ◆ Server name ◆ Database name ◆ User ◆ Password ◆ Port
DBCOMMON-CONNECT-FAILED000002=Operation failed because of a database connection error. Reason: [{0}]	Occurs while retrieving a connection from the database or connection pool.	▪ Verify that the database has not terminated with unexpected errors.
DBCOMMON-CONNECT-FAILED000005=Connection handle not usable. Reason:[{0}]	The connection in the pool is stale and unusable.	▪ Probably a database restart occurred causing the connection to be stale, retry the operation after the database is up.
DBCOMMON-XARESOURCE-FAILED000001=Unable to get XAResource for the database. Reason: [{0}]	Could not obtain XAResource for the connection.	▪ Check if the database supports XA and has been configured for Distributed Transactions.
DBCOMMON-XACONNECT-FAILED000001=Failed to connect to database {0} on host {1}. The XA connection could not be allocated: Reason [{2}]	Occurs during the initial database connection.	▪ Check if the database is configured for XA and if the database is running. ▪ External configuration information is invalid. You may need to verify the following: ◆ Server name ◆ Database name ◆ User ◆ Password ◆ Port
DBCOMMON-XASTART-FAILED000001=Unable to perform XAStart for the connection. Reason: [{0}]	A connection error has occurred which caused XASTART to fail.	▪ Check if the database is running, and there are no network issues.

Alert Code\Description	Description Details	User Actions
DBCOMMON-XAEND-FAILED000001=XAEnd failed. Reason: [{0}]	Error occurred during commit on XA connection.	▪ Look for the detailed error mentioned in the alert for the appropriate action.
DBCOMMON-CANNOT-GET-ISOLATION-LEVEL=Unable to get isolationLevel for the transaction. Reason: [{0}]	Could not read transaction isolation information of the connection.	▪ Transaction isolation is one of the following constants: ◆ Connection.TRANSACTION_READ_UNCOMMITTED ◆ Connection.TRANSACTION_READ_COMMITTED ◆ Connection.TRANSACTION_REPEATABLE_READ ◆ Connection.TRANSACTION_SERIALIZABLE ◆ Connection.TRANSACTION_NONE ***Note:*** Confirm with the vendor that the getIsolation() method of the connection is implemented correctly.

Note: *An alert code is a warning that an error has occurred. It is not a diagnostic. The user actions noted above are just some possible corrective measures you may take. Refer to the log files for more information. For information on Managing and Monitoring alert codes and logs, see the Sun SeeBeyond eGate Integrator System Administration Guide.*

2.6 Deploying an EAR File to Application Servers

The Sun Java CAPS Enterprise Designer can be configured to automatically deploy an EAR file to the Sun Java System Application Server. To configure the Enterprise Designer for deployment, follow the directions for deploying applications to the Sun Java System Application Server, provided in the *Sun SeeBeyond eGate Integrator System Administration Guide*. Because automatic deployment is not supported directly from Enterprise Designer for the Weblogic Application Server, additional instructions are provided below.

2.6.1 WebLogic Application Servers

1. Build the EAR file, which is generated in the Enterprise Designer.
2. Use your WebLogic Admin console to deploy the EAR file.

Refer to your application server's documentation for requirements regarding working directories.

Chapter 3

Configuring the Oracle eWay

This chapter describes how to set the properties of the Oracle eWay.

What's in This Chapter

3.1 Creating and Configuring the Oracle eWay

All eWays contain a set of parameters with properties that are unique to that eWay type. The Oracle eWay properties are modified from these locations:

- **Connectivity Map**: These parameters most commonly apply to a specific component eWay, and may vary from other eWays (of the same type) in the Project.
- **Environment Explorer**: These parameters are commonly global, applying to all eWays (of the same type) in the Project. The saved properties are shared by all eWays in the Oracle External System window.
- **Collaboration or Business Process**: Oracle eWay properties may also be set from your Collaboration or Business process, in which case the settings will override the corresponding properties in the eWay's configuration file. Any properties that are not overridden retain their configured default settings.

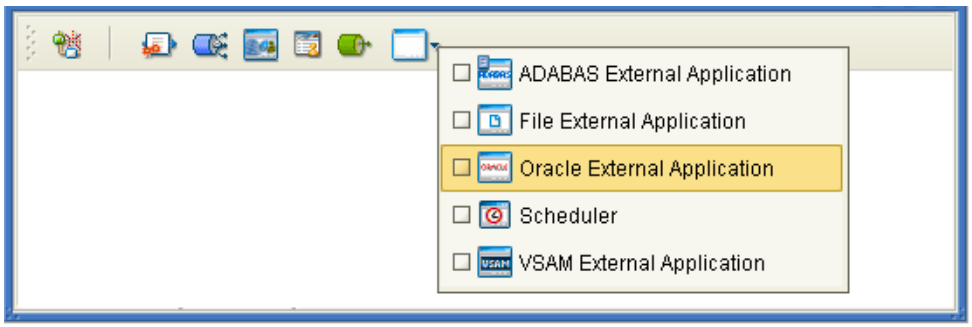
3.1.1 Selecting Oracle as the External Application

To create a Oracle eWay you must first create a Oracle External Application in your Connectivity Map. Oracle eWays are located between a Oracle External Application and a Service. Services are containers for Collaborations, Business Processes, eTL processes, and so forth.

To create the Oracle External Application

1. From the Connectivity Map toolbar, click the **External Applications** icon.
2. Select the **Oracle External Application** from the menu (see Figure 1). The selected Oracle External Application icon now appears on the Connectivity Map toolbar.

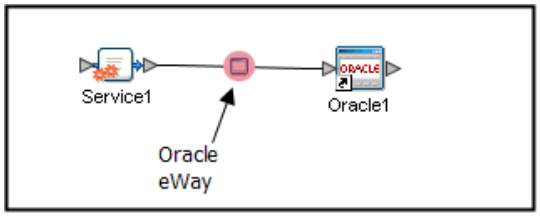
Figure 1 External Application menu



3 Drag the new **Oracle External Application** from the toolbar onto the Connectivity Map canvas. This icon now represents an external Oracle system.

From the Connectivity Map, you can associate (bind) the External Application to the Service to establish an eWay (see Figure 2).

Figure 2 eWay Location.



When Oracle is selected as the External Application, it automatically applies the default Oracle eWay properties, provided by the OTD, to the eWay that connects it with the Service. These properties can then be or modified for your specific system using the **Properties Editor**.

3.1.2 Configuring the Oracle eWay Properties

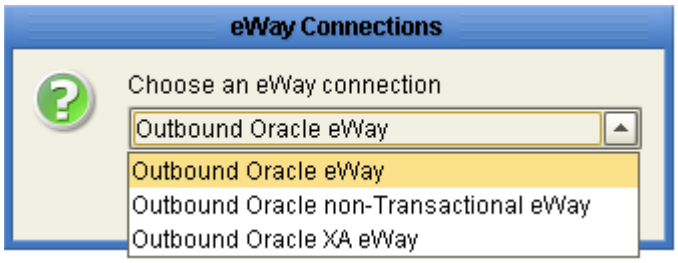
A Project's eWay properties can be modified after the eWay has been established in the Connectivity Map and the Environment has been created.

Configuring the Oracle eWay (Connectivity Map) Properties and setting the type of transaction support

1 On the Enterprise Designer's Connectivity Map, double-click the Oracle eWay icon. The eWay Connections window appears.

2 Select a transaction support level from the list and click **OK**.

Figure 3 Template window



The choices to make are as follows:

- **Outbound Oracle eWay:** Also referred to as LocalTransaction, this support level is opposite to NoTransaction, and this means that the transaction, when aborted, will roll back all changes made since the beginning of the transaction.
- **Outbound Oracle non-Transactional eWay:** Also referred to as NoTransaction, this support level indicates that the Collaboration does not support transactions. This means that when a transaction aborts, there is no ability to roll back any changes to the previous update.
- **Outbound Oracle XA-eWay:** Also referred to as XATransaction, this support level allows two-phase commit. This means that the transaction, when aborted, will roll back all changes when one of the updates fails. The update could occur in the database eWay or other eWays that support XA. Additionally, the Collaboration can contain only the database eWay, or a combination of database eWay and other eWays that support XA.

3 The Configuration properties window opens, displaying the default properties for the eWay

Configuring the Oracle eWay (Environment Explorer) Properties

1 From the **Environment Explorer** tree, right-click the Oracle External System. Select **Properties** from the shortcut menu. The **Properties Editor** opens with the Oracle eWay Environment properties.

2 Make any necessary changes to the Environment property values, and click **OK** to save the settings.

3.1.3 Transaction Support Levels Between Different Versions

The types of transaction support levels used in Java CAPS 5.1 may be different from the support levels used in Java CAPS 5.1.3. Projects that are imported from a Java CAPS 5.1.1 version can potentially display different results, depending on whether the 5.1.1 Java Collaboration Definition (JCD) included multiple (insert/update/delete) operations. This only affects non-XA transactions. If you are using an XA transaction, then you can skip this section.

Example:

In 5.1.0, five new records are to be inserted into a table. If the last record fails to insert (such as when a duplicate key exists), all previous records will have been inserted. This is the behavior of NoTransaction support.

In 5.1.3, five new records are to be inserted into a table. If one of the records fails to insert (such as when a duplicate key exists), the other four records will not be inserted. This is the behavior of the LocalTransaction.

In order to achieve the same result as in 5.1.1 versions, you can choose the method below:

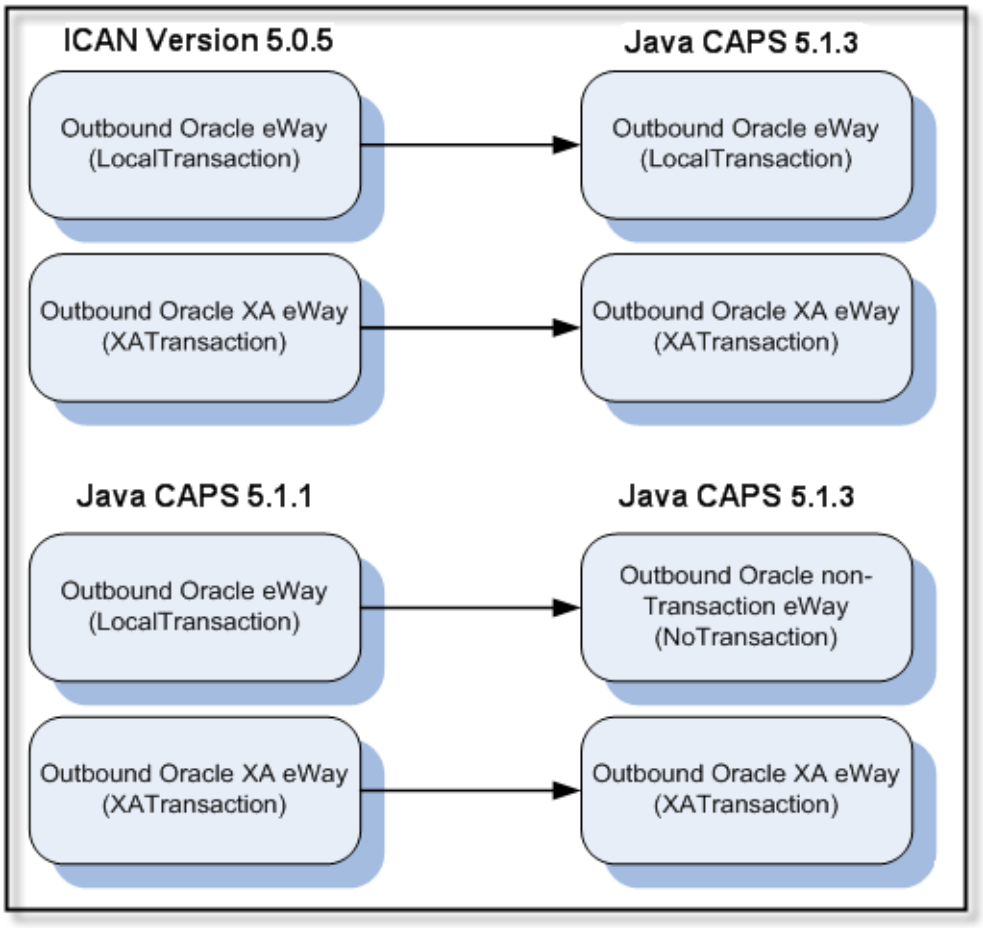
A In the Connectivity Map, delete the link to the database external application, then reconnect the link and select NoTransaction.

B Fill in the NoTransaction property for the database external system under the Environment.

C Rebuild the Project.

The following charts identifies what transaction support levels changed between 5.0.5 and 5.1.3, and 5.1.1 and 5.1.3, respectively. **Note that there are no changes when migrating from ICAN version 5.0.5 and Java CAPS 5.1.3.**

Figure 4 Transaction Support Levels



Under the scenario noted above, if you want 5.1.3 behavior for a LocalTransaction, then set your eWay connection to be Outbound Oracle non-Transactional eWay (NoTransaction).

3.1.4 Using the Properties Editor

Modifications to the eWay configuration properties are made from the Oracle eWay **Properties Editor**.

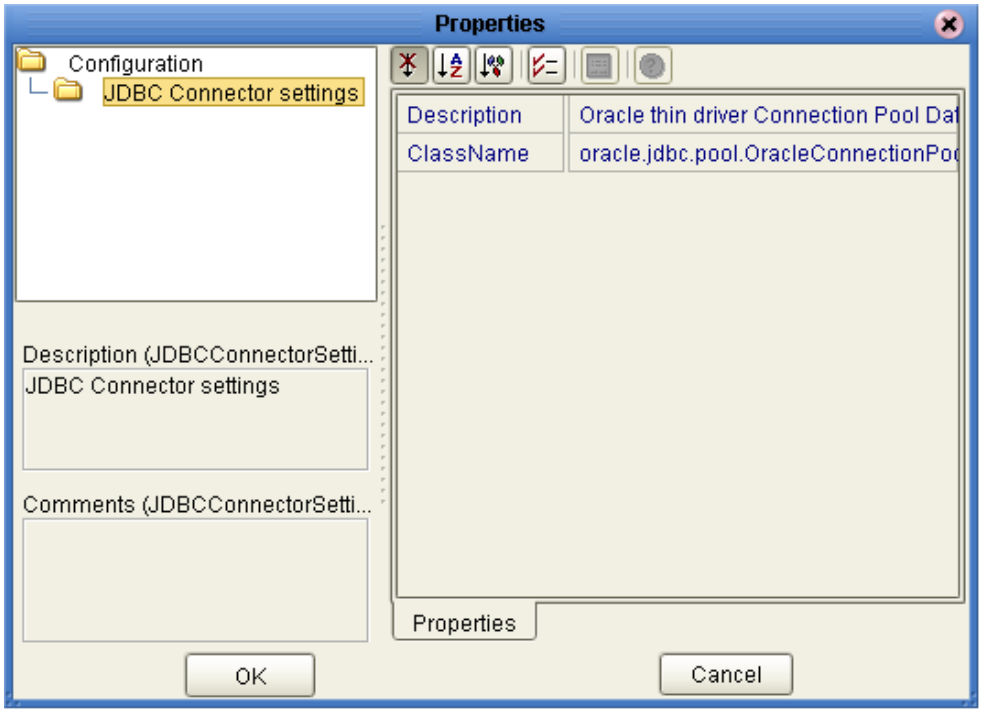
A description of each property is displayed in the **Description** pane when it is selected. This provides a brief explanation of the required settings or options.

The **Comments** pane provides an area to record notes and information regarding the currently selected property. These comments are saved when you close the editor.

Modifying the Default eWay Configuration Properties

1. From the Connectivity Map or the Environment Explorer, open the Properties Editor to the Oracle eWay default properties.
2. From the upper-right pane of the Properties Editor, select a subdirectory of the configuration directory. The parameters contained in that subdirectory are now displayed in the Properties pane of the Properties Editor. For example, if you click on the **JDBC Connector settings** subdirectory, the editable parameters are displayed in the right pane (see Figure 5).

Figure 5 Properties Editor -- Oracle Properties



3. Click on any property field to make it editable. For example, click on the **Description** property to edit the class value. If a property value is true/false or multiple choice, the field displays a submenu of property options.
4. Click on the ellipsis (**...**) in the properties field to open a separate configuration dialog box. This is helpful for large values that cannot be fully displayed in the parameter's property field. Enter the property value in the dialog box and click **OK**. The value is now displayed in the property field.

5 After modifying the configuration properties, click **OK** to close the Properties Editor and save your changes.

3.2 Oracle eWay Connectivity Map Properties

The Oracle eWay configuration parameters, accessed from the Connectivity Map, are organized into the following sections:

- **Outbound Connectivity Map Properties** on page 22
- **Inbound Connectivity Map Properties** on page 23

3.2.1 Outbound Connectivity Map Properties

The Outbound configuration parameters, accessed from the Connectivity Map, are organized into the following sections:

- **Properties in the Outbound eWay** on page 22
- **Properties in the Outbound non-Transactional eWay** on page 22
- **Properties in the Outbound eWay with XA Support** on page 23

Properties in the Outbound eWay

The **JDBC Connector Settings** section of the Oracle Connectivity Map properties contains the top-level parameters displayed in Table 3.

Table 3 Outbound Connectivity Map JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **Oracle thin driver Connection Pool Datasource**
ClassName	Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.	A valid class name. The default is **oracle.jdbc.pool.OracleConnectionPoolDataSource** ***Note:*** Do not change this value.

Properties in the Outbound non-Transactional eWay

The Outbound non-Transactional eWay Properties include outbound parameters used by the external database.

Table 4 Outbound non-Transactional eWay—JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **Oracle thin driver non-Transactional Connection Pool Datasource**.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.	A valid class name. The default is **oracle.jdbc.pool.OracleConnectionPoolDataSource**.

Properties in the Outbound eWay with XA Support

The **JDBC Connector Settings** section of the Oracle Connectivity Map properties contains the top-level parameters displayed in Table 5.

Table 5 Outbound with XA Support JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **Oracle thin driver XA Datasource**.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.	A valid class name. The default is **oracle.jdbc.xa.client.OracleXADataSource**.

3.2.2 Inbound Connectivity Map Properties

The **Parameter Settings** section of the Oracle Connectivity Map properties contains the top-level parameters displayed in Table 6.

Table 6 Inbound eWay Connectivity Map Parameter Settings

Name	Description	Required Value
Pollmilliseconds	Polling interval in milliseconds.	A valid numeric value. The default is 5000.

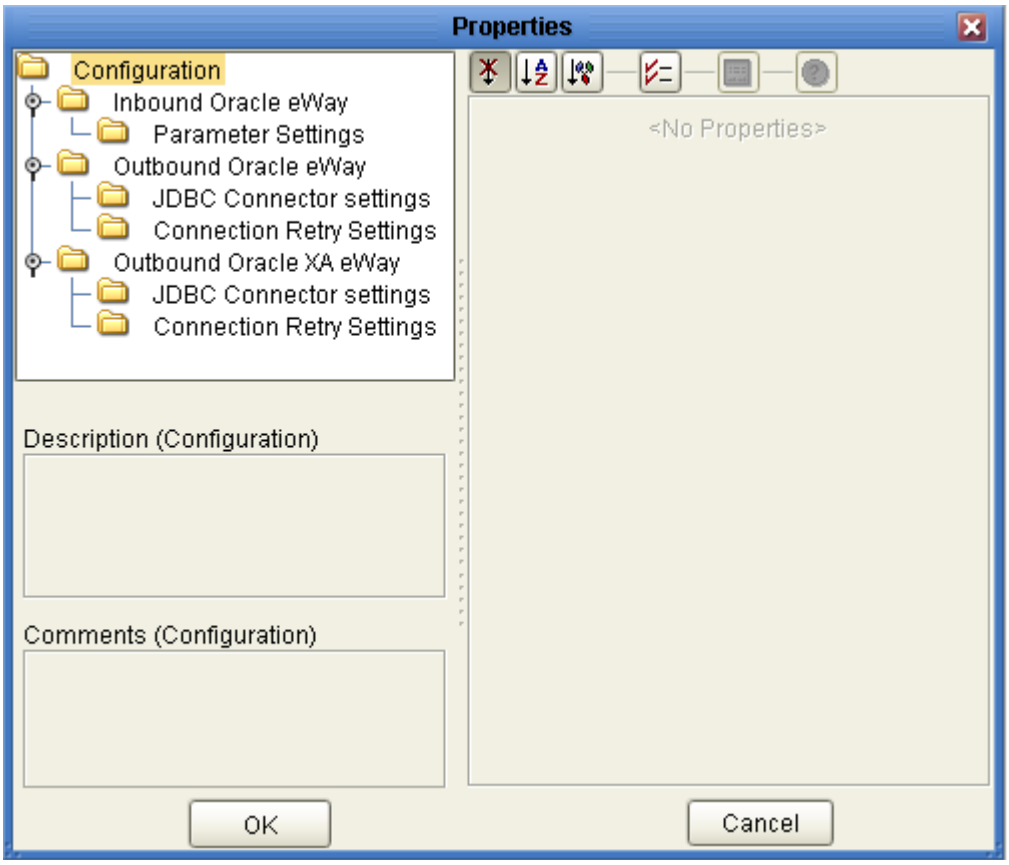
Table 6 Inbound eWay Connectivity Map Parameter Settings

Name	Description	Required Value
PreparedStatement	The Prepared Statement used for polling against the database.	The Prepared Statement must be the same Statement you created using the Database OTD Wizard. Only a SELECT Statement is allowed. Additionally, no place holders should be used. This is a SQL statement that cannot contain any input data (i.e. you cannot use "?" in the Prepared Query).

3.3 Oracle eWay Environment Explorer Properties

The Oracle eWay configuration parameters, accessed from the Environment Explorer tree, are organized into the following sections:

Figure 6 Oracle eWay Environment Configuration



3.3.1 Inbound Oracle eWay Properties

The **Parameter Settings** section of the Inbound Oracle Environment contains the top-level parameters displayed in Table 7.

Table 7 Inbound Oracle eWay Environment Properties

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **Oracle DriverManager**
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1521**.
DatabaseName	Specifies the name of the Oracle SID.	Any valid string.
User	Specifies the user name that the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

3.3.2 Outbound Oracle eWay Properties

The Outbound Oracle eWay properties, accessed from the Environment Explorer tree, are organized into the following sections:

JDBC Connector Settings

The **JDBC Connector Settings** section of the Outbound Oracle Environment contains the top-level parameters displayed in Table 8.

Table 8 Outbound eWay Environment JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **Oracle Connection Pool Datasource**
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1521**.
DatabaseName	Specifies the name of the Oracle SID.	Any valid string.
User	Specifies the user name that the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties (**Optional Setting**)	Use the JDBC driver that is shipped with this eWay. The DataSource implementation may need to execute additional methods to assure a successful run. The additional methods will need to be identified in the Driver Properties. ***Note:*** This field is seldomly used for Oracle.	The delimiter set by the user. For more information, see the Delimiter property below. Valid delimiters are: **"<method-name-1>#<param-1>#<param-2>#……<param-n>##<method-name-2>#<param-1>#<param-2>#……<param-n>##……##".**
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**. See the DriverProperties property above for more information on how the default value is used.

Table 8 Outbound eWay Environment JDBC Connector Settings

Name	Description	Required Value
TNSEntry	Specifies the TNS name for the Oracle instance specified in TNSNAMES.ORA. If a TNS name is specified, then the OCI driver is used, which further requires installation of the Oracle client. If a TNS name is not specified, then the thin driver is used.	A valid TNS name if using the OCI driver; otherwise do not enter any value.
MinPoolSize	Specifies 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 the new connections should be created as needed. If the pool size is too small, you may experience longer connection times due to the existing number of physical connections. A connection that stays in the pool allows transactions to use it via a logical connection which is faster.	A valid numeric value. The default is **0**.
MaxPoolSize	Specifies the maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size depends on the transaction volume and response time. If the pool size is too big, you may end up with too many connections with the database.	A valid numeric value. The default is **10**.
MaxIdleTime	The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value. The default is **0**.

Connection Retry Settings

The **Connection Retry Settings** section of the Outbound Oracle Environment contains the top-level parameters displayed in Table 9.

Table 9 Outbound eWay Environment Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection with the Oracle database upon a failure to acquire one.	an integer indicationg the number of attempts allowed to establish a connection. The configured default is **0**.
ConnectionRetry Interval	Specifies the milliseconds of pause before each attempt to access the database. This setting is used in conjunction with the 'Connection Retries' setting. For example: In the event that the eWay cannot connect to the Database, the eWay will try to reconnect to the database in 5 second intervals, a total of 10 times, when the Connection Retries property is set at 10 and the Connection Retry Interval property is 5000.	An integer indicating the configured length of the time (in milliseconds) before each reattempt to access the destination file. The configured default is **1000** (1 second).

3.3.3 Outbound Oracle eWay Properties with XA support

The Outbound Oracle eWay properties with XA support, accessed from the Environment Explorer tree, are organized into the following sections:

- **JDBC Connector Settings (with XA support)** on page 28
- **Connection Retry Settings (with XA support)** on page 30

JDBC Connector Settings (with XA support)

The **JDBC Connector Settings** section of the Outbound XA Oracle Environment contains the top-level parameters displayed in Table 10.

Table 10 Outbound XA eWay Environment JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **Oracle XA Datasource**
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1521**.
DatabaseName	Specifies the name of the Oracle SID.	Any valid string.
User	Specifies the user name that the eWay uses to connect to the database.	Any valid string.

Table 10 Outbound XA eWay Environment JDBC Connector Settings

Name	Description	Required Value
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties (**Optional Setting**)	Use the JDBC driver that is shipped with this eWay. The DataSource implementation may need to execute additional methods to assure a successful run. The additional methods will need to be identified in the Driver Properties.	The delimiter set by the user. For more information, see the Delimiter property below. Valid delimiters are: **"<method-name-1>#<param-1>#<param-2>#.....<param-n>##<method-name-2>#<param-1>#<param-2>#.....<param-n>##.....##".**
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**. See the DriverProperties property above for more information on how the default value is used.
TNSEntry	Specifies the TNS name for the Oracle instance specified in TNSNAMES.ORA. If a TNS name is specified, then the OCI driver is used, which further requires installation of the Oracle client. If a TNS name is not specified, then the thin driver is used.	A valid TNS name if using the OCI driver; otherwise do not enter any value.
MinPoolSize	Specifies 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 the new connections should be created as needed. If the pool size is too small, you may experience longer connection times due to the existing number of physical connections. A connection that stays in the pool allows transactions to use it via a logical connection which is faster.	A valid numeric value. The default is **0**.

Table 10 Outbound XA eWay Environment JDBC Connector Settings

Name	Description	Required Value
MaxPoolSize	Specifies the maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size depends on the transaction volume and response time. If the pool size is too big, you may end up with too many connections with the database.	A valid numeric value. The default is **10**.
MaxIdleTime	The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value. The default is **0**.

Connection Retry Settings (with XA support)

The **Connection Retry Settings** section of the Outbound XA Oracle Environment contains the top-level parameters displayed in Table 11.

Table 11 Outbound XA eWay Environment Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection with the Oracle database upon a failure to acquire one.	an integer indicating the number of attempts allowed to establish a connection. The configured default is **0**.
ConnectionRetry Interval	Specifies the milliseconds of pause before each attempt to access the database. This setting is used in conjunction with the 'Connection Retries' setting. For example: In the event that the eWay cannot connect to the Database, the eWay will try to reconnect to the database in 5 second intervals, a total of 10 times, when the Connection Retries property is set at 10 and the Connection Retry Interval property is 5000.	An integer indicating the configured length of the time (in milliseconds) before each reattempt to access the destination file. The configured default is **1000** (1 second).

Chapter 4

Using the Oracle eWay OTD Wizard

This chapter describes how to use the Oracle eWay Database Wizard to build OTD's, including using the Statement Builder Wizard.

What's in This Chapter

4.1 Using the Database OTD Wizard

The Database OTD Wizard generates OTDs by connecting to external data sources and creating corresponding Object Type Definitions. The OTD Wizard can create OTDs based on any combination of Tables and Stored Procedures or Prepared SQL Statements.

Field nodes are added to the OTD based on the Tables in the external data source. Java method and parameter nodes are added to provide the appropriate JDBC functionality. For more information about the Java methods, refer to your JDBC developer's reference.

The Oracle eWay also supports Double-Byte Character Set (DBCS) table and column names. The DBCS is a set of characters in which each character is represented by two bytes. Japanese and Korean languages require double-byte character sets.

Note: *Database OTDs are not messagable. For more information on messagable OTDs, see the eGate Integrator User's Guide.*

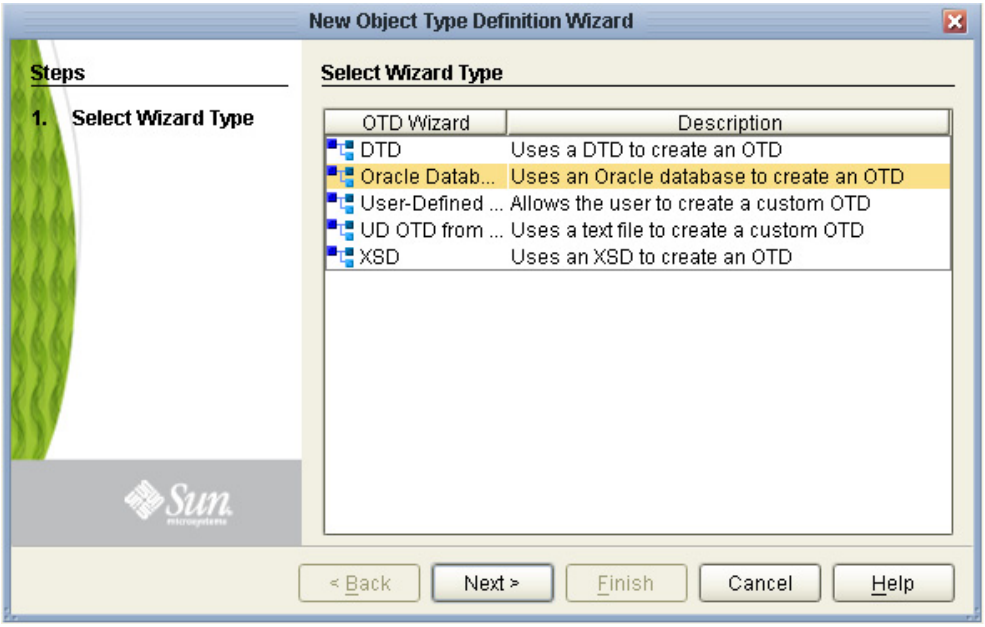
4.2 Creating a New Oracle OTD

The following steps are required to create a new OTD for the Oracle Intelligent Adapter eWay.

4.2.1 Select Wizard Type

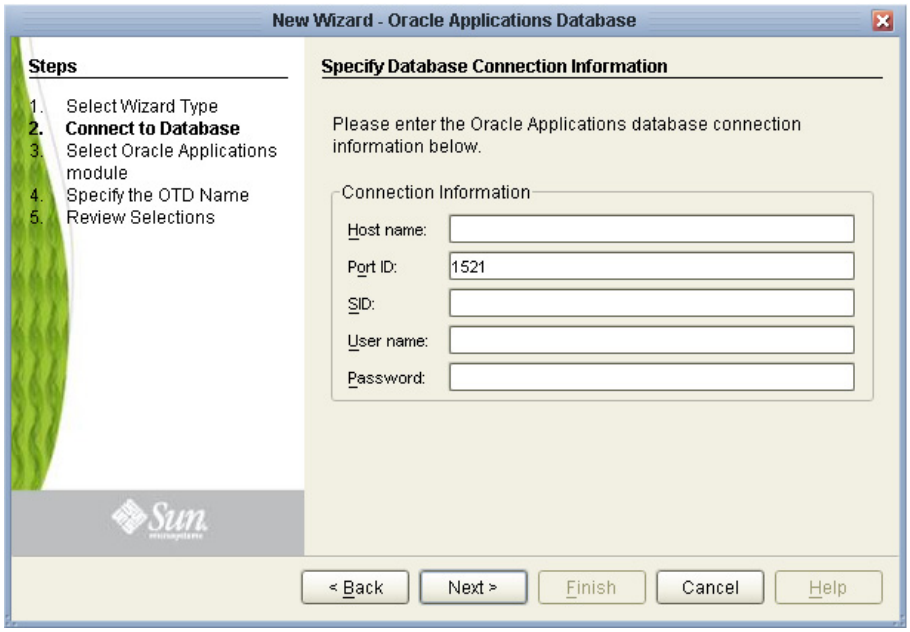
1. On the Enterprise Explorer, right click on the project and select **Create an Object Type Definition** from the shortcut menu.
2. From the OTD Wizard Selection window, select the **Oracle Database** and click **Next** (See Figure 7).

Figure 7 OTD Wizard Selection



4.2.2 Connect to Database

Figure 8 Database Connection Information



1 Specify the applicable connection information for your database including:

- **Host Name** - The server where Oracle resides.
- **Port ID** - The port number of Oracle.
- **SID** - The name of the Oracle instance (equivalent to the database name).
- **User Name** - The user name that the eWay uses to connect to the database.
- **Password** - The password used to access the database.

2 Click **Next**. The Select Database Objects window appears.

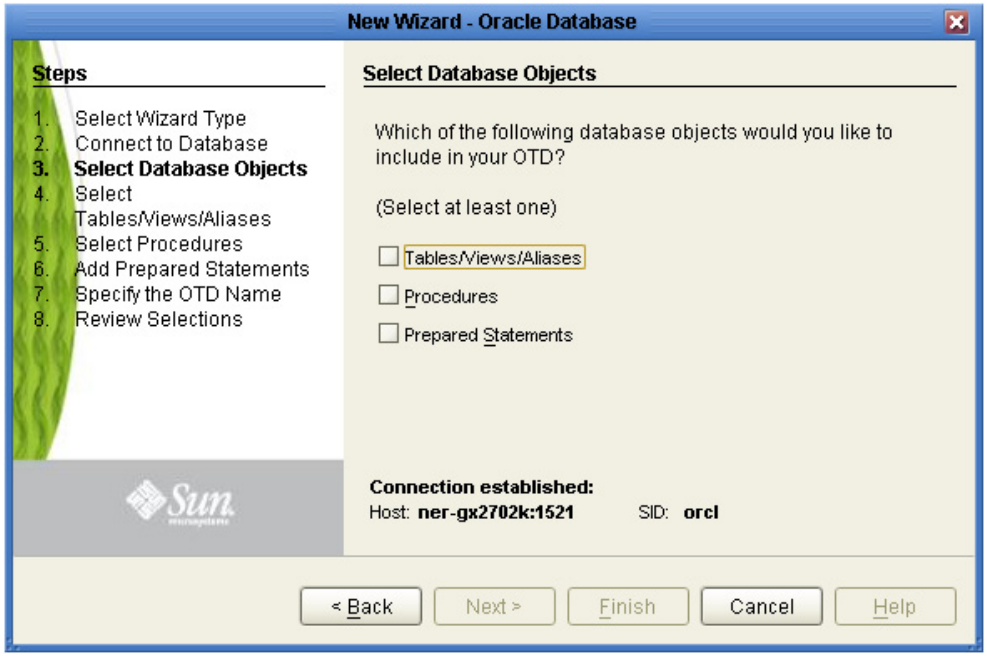
4.2.3 Select Database Objects

Select Database Objects

1 In the **Select Database Objects** dialog box, shown in Figure 9, select **Tables/Views** for this sample. When selecting Database Objects, you can select any combination of **Tables**, **Views**, **Procedures**, or **Prepared Statements** you would like to include in the .otd file. Click **Next** to continue.

Note: *Views are read-only and are for informational purposes only.*

Figure 9 Select Database Objects

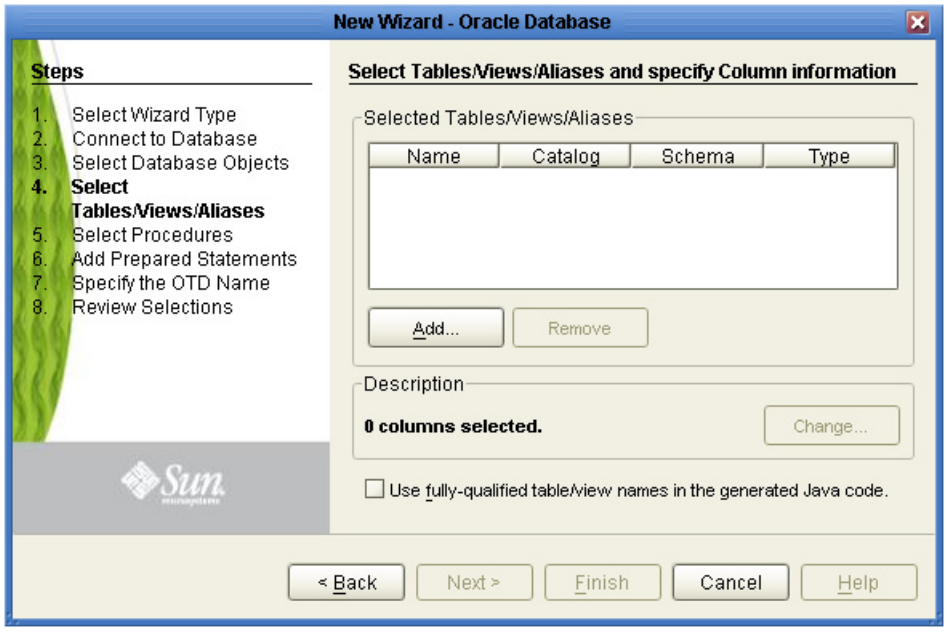


2 Click **Next** to continue. The **Select Tables/Views/Aliases** window appears (depending on your selection).

4.2.4 Select Table/Views/Aliases

1 In the **Select Tables/Views/Aliases** window, click **Add** (see Figure 10).

Figure 10 Select Tables/Views

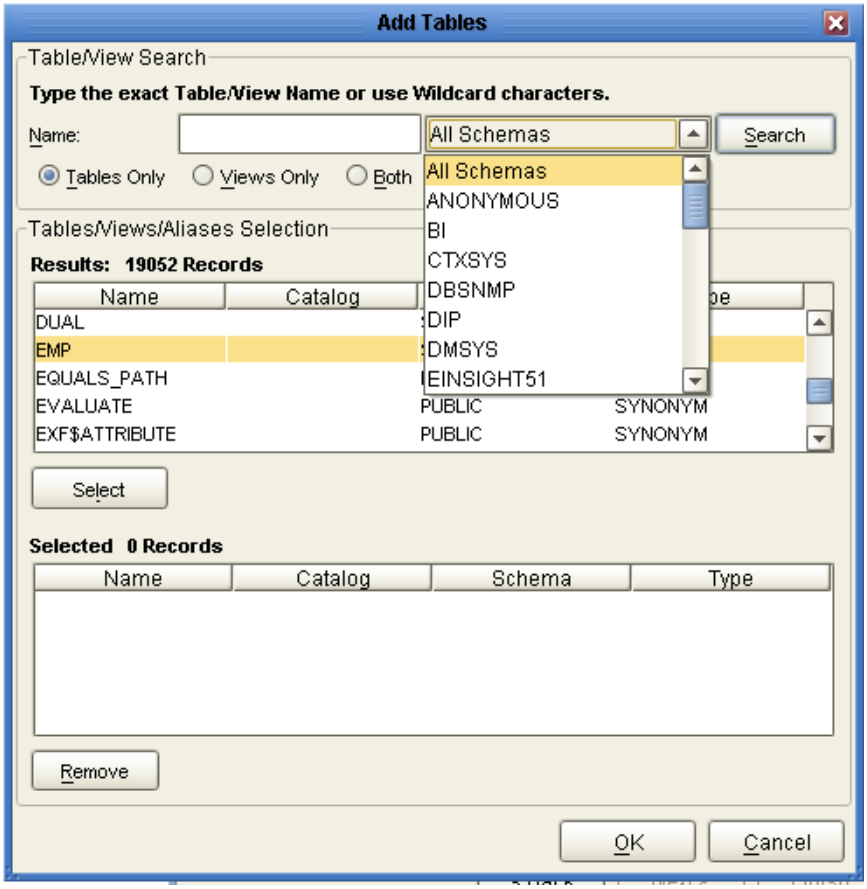


2 In the **Add Tables** window, select the type of criteria to be used for your search, consisting of table data, view only data, or both. You can include system tables in your search by selecting the checkbox.

3 From the **Table/View** Name drop down list, select the location of your database table and click **Search** (see Figure 11). You can search for Table/View Names by entering a table name. The use of wildcard characters of '?', and '*' as part of your Table/View name search allow for greater search capabilities. For example, "AB?CD" or "AB*CD".

4 Select the table of choice and click **OK**. The table selected is added to the Selected window (see Figure 11).

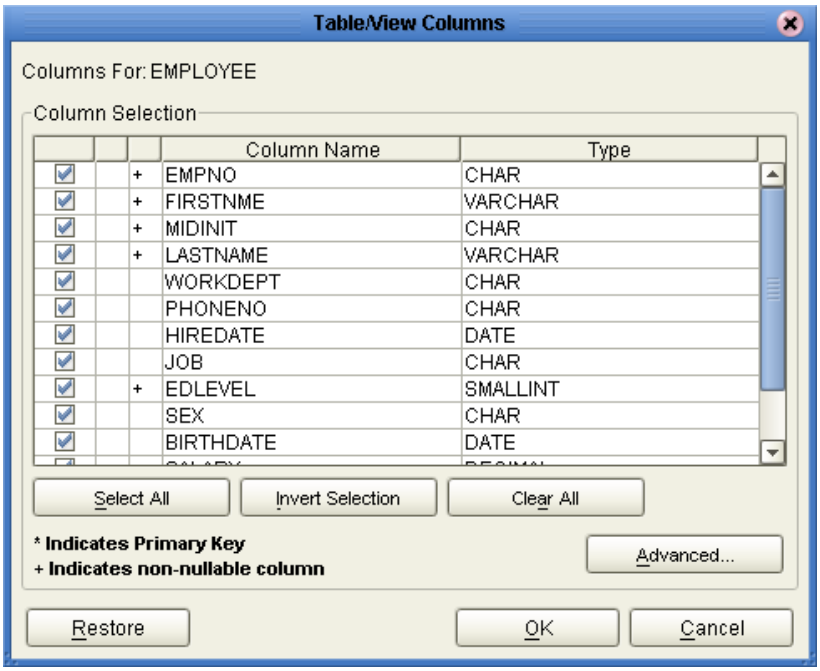
Figure 11 Selected Tables/Views window with a table selected



5 On the **Selected Tables/Views** window, review the table(s) you have selected. To make changes to the selected Table or View, click **Change**. If you do not wish to make any additional changes, click **Next** to continue.

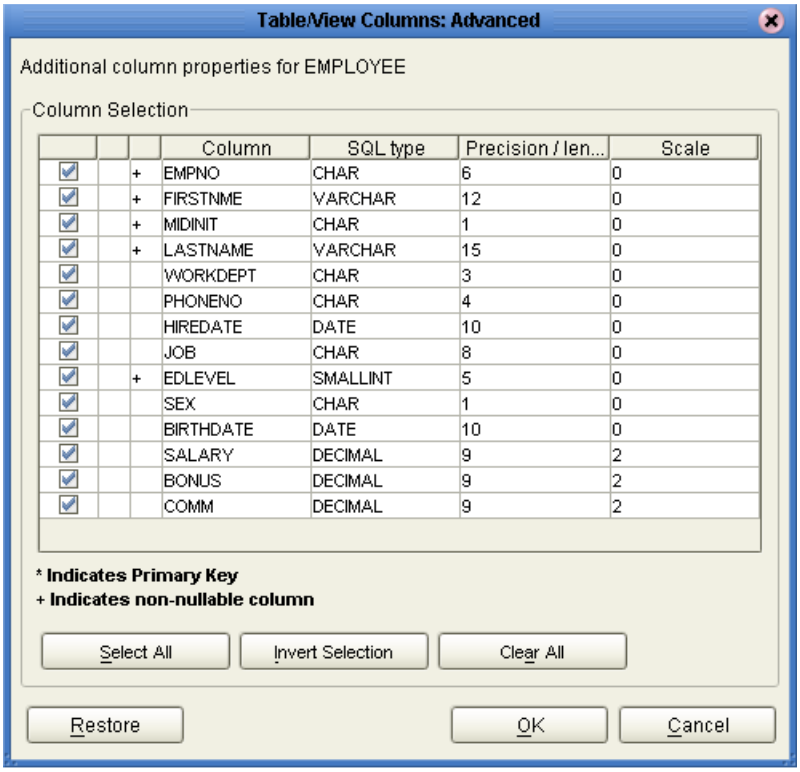
6 If you clicked **Change** on the Selected **Tables/Views** window, you can select or deselect your table columns on the **Table/View Columns** window. You can also change the data type for each table by highlighting the data type and selecting a different one from the drop down (see Figure 12).

Figure 12 Tables/Views Columns



7 Click **Advanced** to change the data type, precision/length, or scale. In general, do not change the precision/length or the scale. Once you have finished your table choices, click **OK** (see Figure 13).

Figure 13 Tables/Views Columns - Advanced

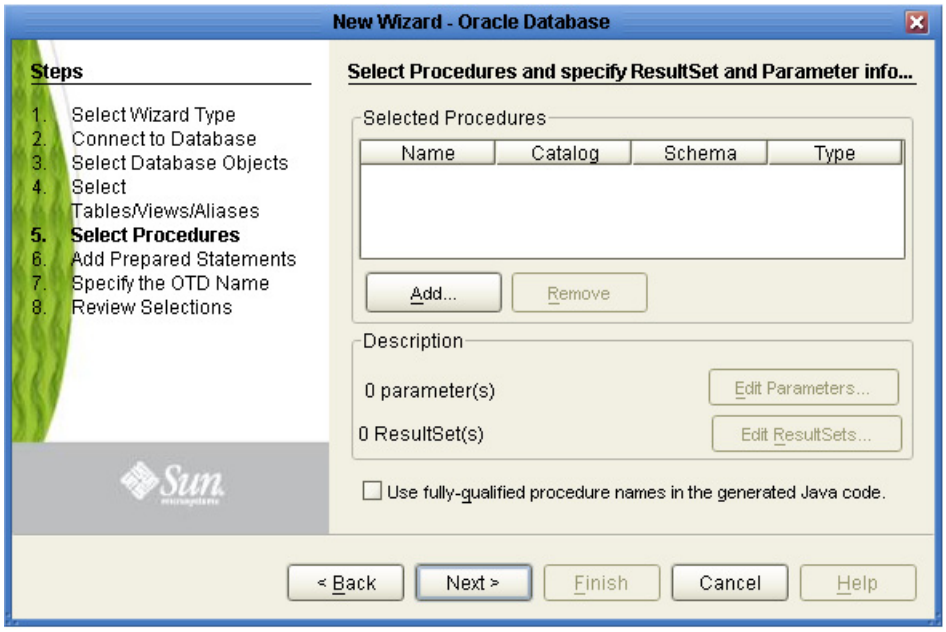


8 When using Prepared Statement packages, select **Use fully qualified table/view names in the generated Java code**.

4.2.5 Select Procedures

1 In the **Select Procedures and specify ResultSet and Parameter Information** dialog box, click **Add.**

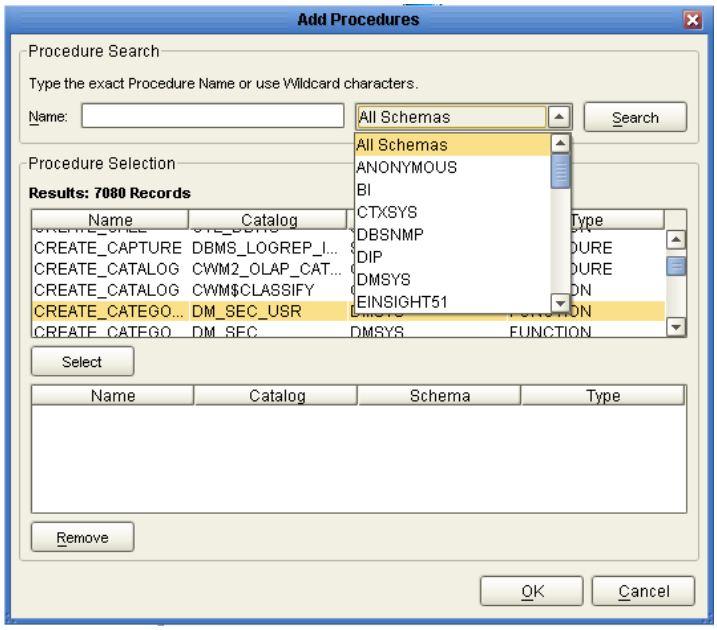
Figure 14 Select Procedures and specify ResultSet and Parameter Information



2 In the **Select Procedures** dialog box, enter the name (case-sensitive) of a Procedure or select a table from the drop-down list. Click **Search**. You can use Wildcard characters.

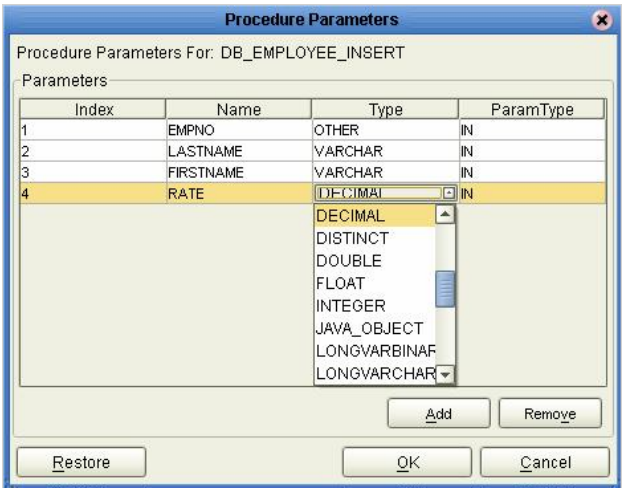
3 In the resulting **Procedure Selection** list box, select a Procedure. Click **OK**.

Figure 15 Add Procedures



4 In the **Select Procedures and specify ResultSet and Parameter Information** dialog box, click **Edit Parameters** to make changes (as needed) to the selected Procedure (see Figure 16).

Figure 16 Procedure Parameters

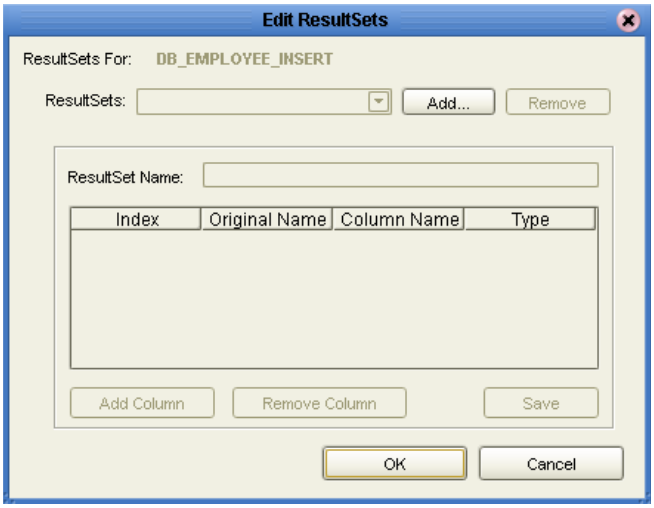


5 To restore the data type, click **Restore**. When you are finished, click **OK**.

6 Click **Edit ResultSets** to have the wizard generate the ResultSet(s) for the Oracle OTD.

7 Select the type of Resultset you want to generate and click **Add**.

Figure 17 Edit ResultSet



The DBWizard provides three different ways to generate the ResultSet nodes of a Stored Procedure. They are the "**By Executing**", "**Manually**", and "**With Assistance**" modes.

By Executing Mode	“By Executing” mode executes the specified Stored Procedure with default values to generate the ResultSet(s). Depending on the business logic of the Stored Procedure, zero or more ResultSets can be returned from the execution. In the case that there are multiple ResultSets and "By Executing" mode does not return all ResultSets, one should use the other modes to generate the ResultSet nodes.
With Assistance Mode	"With Assistance" mode allows users to specify a query and execute it to generate the ResultSet node. To facilitate this operation, the DBWizard tries to retrieve the content of the specified Stored Procedure and display it. However, content retrieval is not supported by all types of Stored Procedures. We can roughly classify Stored Procedures into two types: SQL and external. SQL Stored Procedures are created using CREATE PROCEDURE SQL statements while external Stored Procedures are created using host languages (e.g. Java). Since external Stored Procedures do not store their execution plans in the database, content retrieval is impossible. When using "Assist" mode, highlight the execute statement up to and including the table name(s) before executing the query.

Manually Mode	"Manually" mode is the most flexible way to generate the result set nodes. It allows users to specify the node name, original column name and data type manually. One drawback of this method is that users need to know the original column names and data types. This is not always possible. For example, the column name of 3*C in this query. `SELECT A, B, 3*C FROM table T` is generated by the database. In this case, "With Assistance" mode is a better choice. If you modify the ResultSet generated by the “Execute” mode of the Database Wizard you need to make sure the indexes match the Stored Procedure. This assures your ResultSet indexes are preserved.

8 On the **Select Procedures and specify Resultset and Parameter Information** window click **Next** to continue.

4.2.6 Add Prepared Statements

A Prepared Statement OTD represents a SQL statement that has been compiled. Fields in the OTD correspond to the input values that users need to provide.

Prepared statements can be used to perform insert, update, delete and query operations. A prepared statement uses a question mark (?) as a place holder for input. For example: insert into EMP_TAB (Age, Name, Dept No) values (?, ?, ?)

To execute a prepared statement, set the input parameters and call **executeUpdate()** and specify the input values if any.

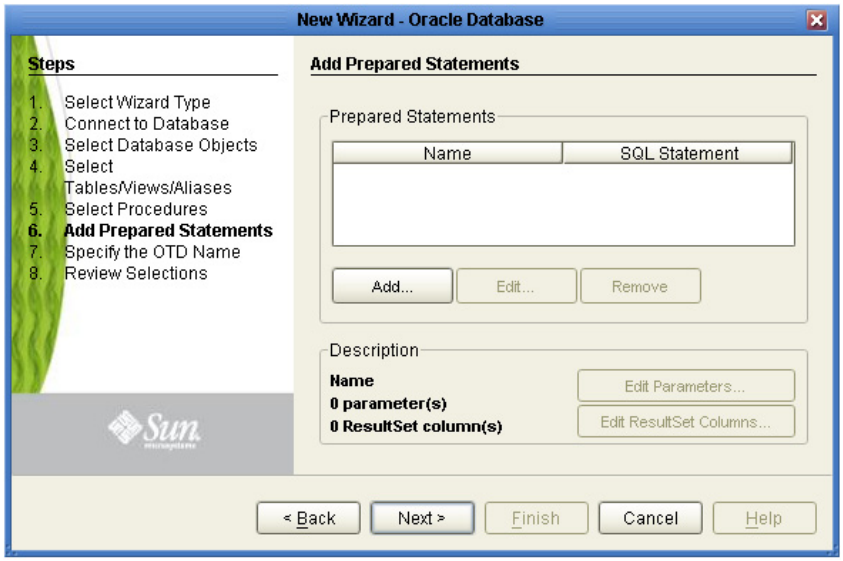
Steps Required to Add Prepared Statements Include:

Note: *When using a Prepared Statement, the ‘ResultsAvailable()’ method will always return true. Although this method is available, you should not use it with a ‘while’ loop. Doing so would result in an infinite loop at runtime and will stop all of the system’s CPU. If it is used, it should only be used with the ‘if’ statement.*

You can process a resultset by looping through the next() method. For more information, see **The Query (Select) Operation** *on page 57.*

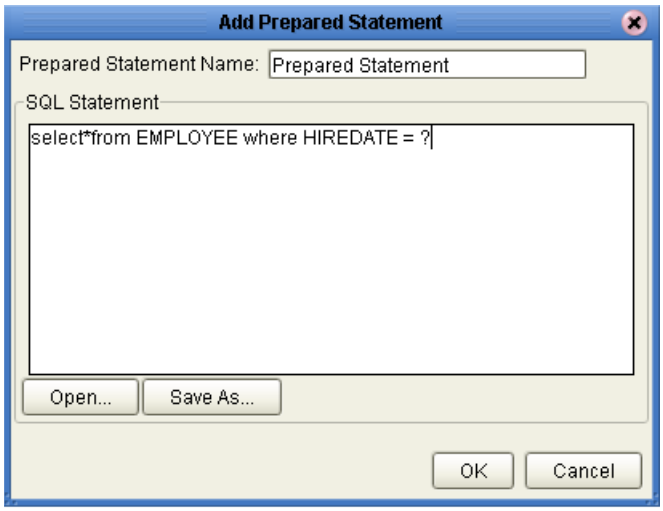
1 In the **Add Prepared Statements** dialog box, click **Add**.

Figure 18 Prepared Statement



2 Enter the name of a Prepared Statement and create a SQL statement by clicking in the **SQL Statement** dialog box or by clicking the **Statement Builder** button. When you are finished creating the statement, click **Save As**, which gives the statement the name you just entered. This name appears as a node in the OTD (see Figure 19). Click **OK**.

Figure 19 Prepared SQL Statement



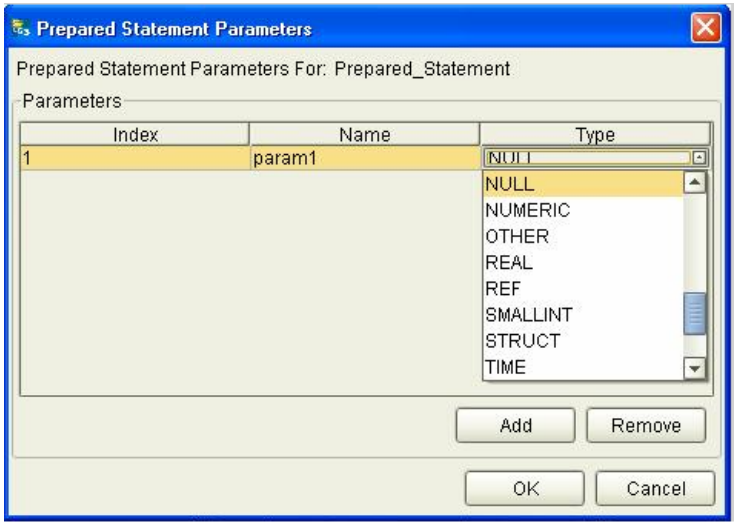
Note: *For more information on creating a Prepared Statement using the **Statement Builder**, refer to* **"Editing Existing OTDs" on page 48**.

3 In the **Add Prepared Statement** dialog box, the name you assigned to the Prepared Statement appears. To edit the parameters, click **Edit Parameters**. You can change the datatype by clicking in the **Type** field and selecting a different type from the list.

4 Click **Add** if you want to add additional parameters to the Statement, or highlight a row and click **Remove** to remove the parameter (see Figure 20). Click **OK**.

Note: *Once you save the Prepared Statement, make sure that the ResultSet Column Name, in the Prepared Statement parameters, is a valid alpha-numeric string with no special characters (e.g. no brackets).*

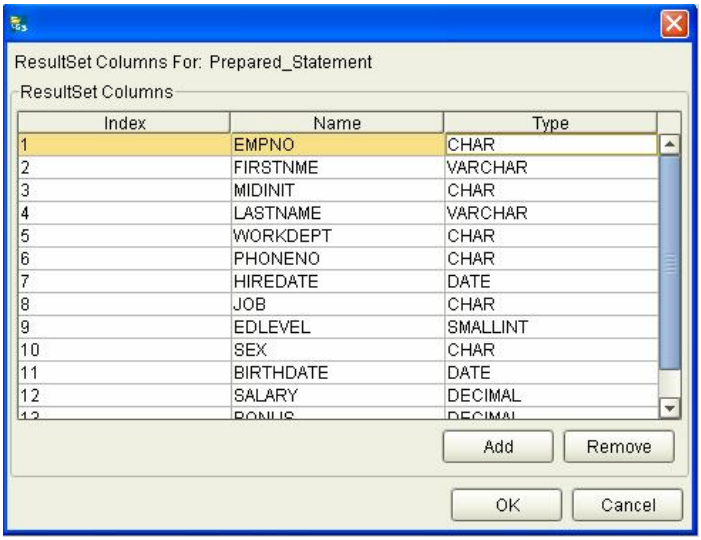
Figure 20 Edit the Prepared Statement Parameters



5 To edit the ResultSet Columns, click **Edit ResultSet Columns** (see Figure 21). Although both the Name and Type are editable, Sun recommends that you do not change the Name because it can cause a loss of integrity between the ResultSet and the Database. Click **OK**.

Note: *The OTD Wizard fails to create OTDs with complex prepared statements that use the same column name in different tables. This problem is resolved by modifying the SQL statement to use column name aliases.*

Figure 21 ResultSet Columns



6 In the **Add Prepared Statements** dialog box, click **OK**.

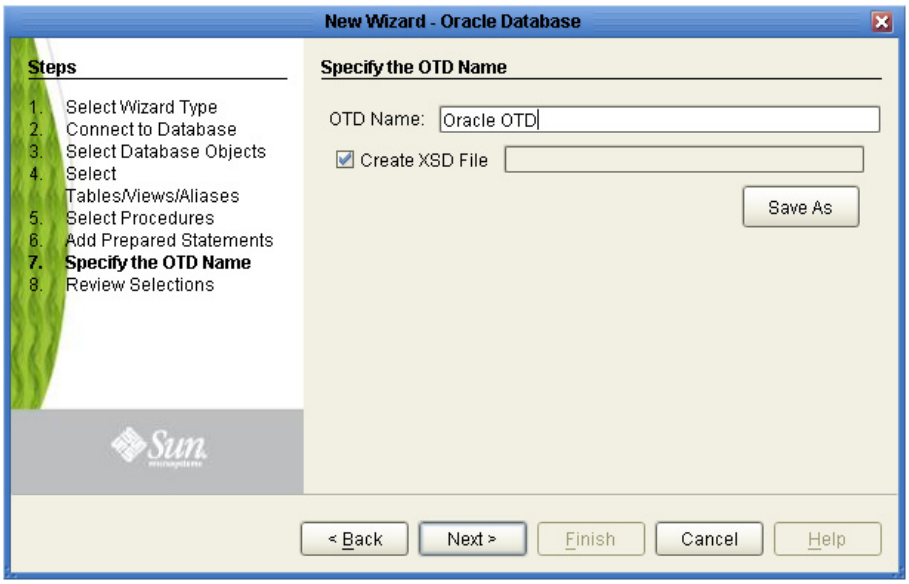
4.2.7 Specify the OTD Name

Specify the name that your OTD will display in the Enterprise Designer Project Explorer.

Steps Required to Specify the OTD Name:

1 Enter a name for the OTD. The OTD contains the selected tables and the package name of the generated classes (see Figure 22).

Figure 22 Naming the OTD



2 By selecting the **Create XSD File** option you can create a XSD schema along with the database OTD. Enter the name of the XSD File to be created and click the **Save As** button to select where to save the file.

The XSD File, which contains database object structures, can then be used to create an XSD OTD File using the XSD OTD Wizard.

3 Click **Next**.

4.2.8 Review Selections

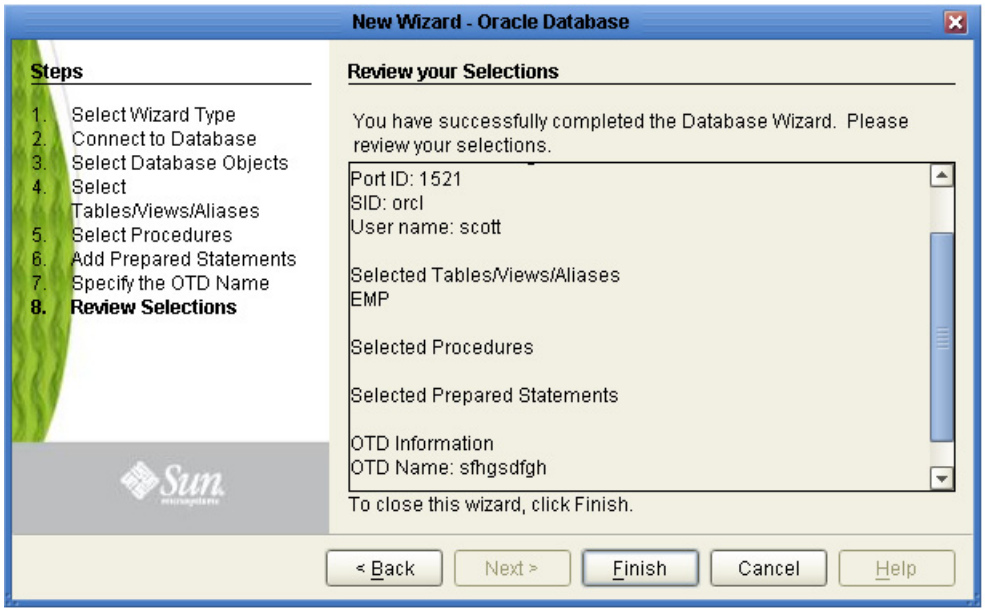
Review the selections made for the new OTD.

Steps Required to Review Your OTD Selections:

1 View the summary of the OTD. If you find you have made a mistake, click **Back** and correct the information.

2 If you are satisfied with the OTD information, click **Finish** to begin generating the OTD (see Figure 23).

Figure 23 Database Wizard - Summary



The resulting OTD appears on the Enterprise Designer's canvas.

4.3 Using the Statement Builder Wizard

Use the Statement Builder wizard to construct a Prepared Statement with point-and-click ease. You can create Prepared Statements, using a graphical user interface, and preview them before incorporating them into an Oracle OTD.

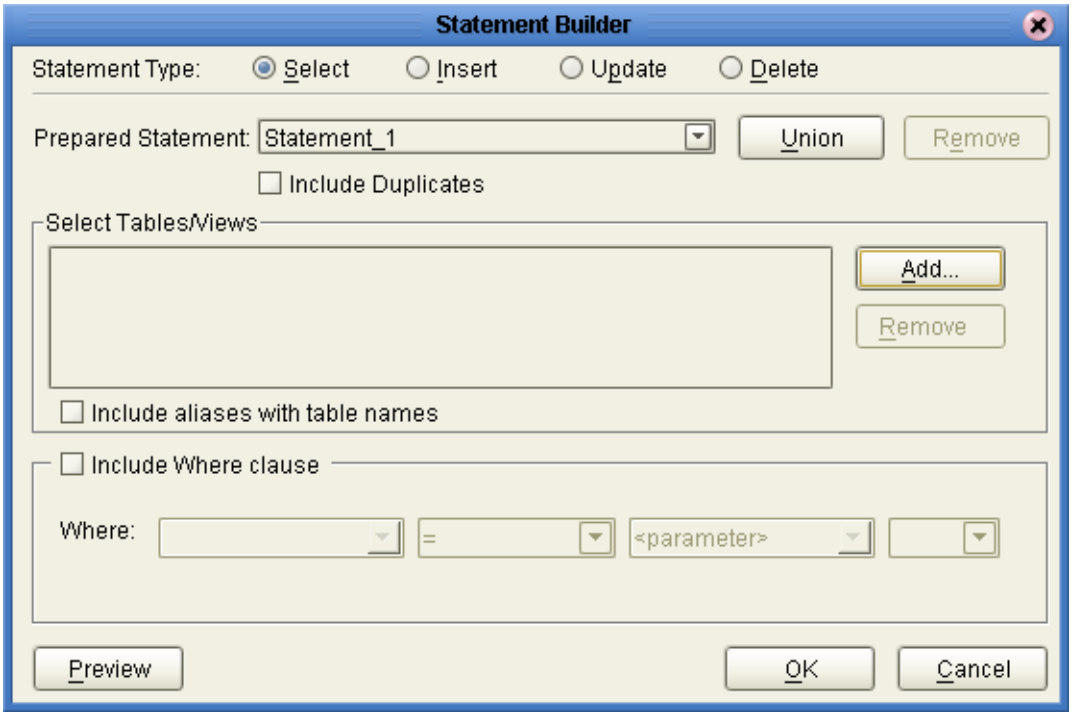
Please note that the Statement Builder wizard:

- is one option for constructing a Prepared Statement, but is not mandatory for building them.
- does not perform data validation.
- does not support the CLOB data type.

Using the Statement Builder

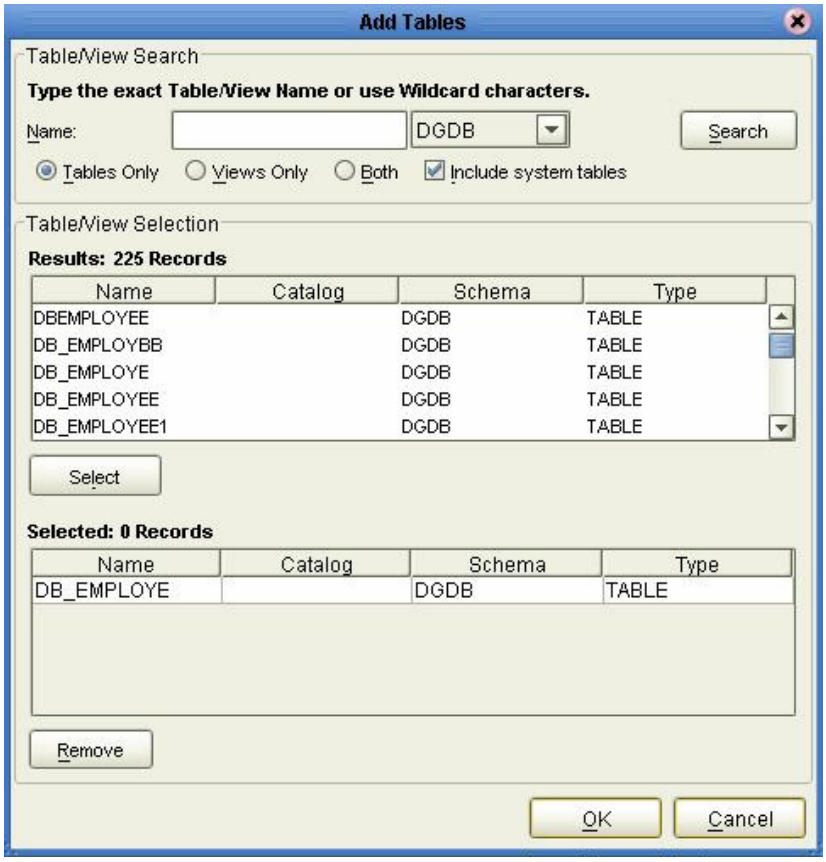
1. Add a Prepared Statement OTD by clicking **Add**, as described in **Add Prepared Statements** on page 40.
2. Enter the name for the Prepared Statement and create a SQL statement by clicking the **Statement Builder** button. The **Statement Builder** dialog box appears (as shown in Figure 24).

Figure 24 Statement Builder



3. Select the type of statement you want to construct and click **Add**. The **Add Tables** dialog box appears (as shown in **Figure 25 on page 46**).

Figure 25 Add Tables

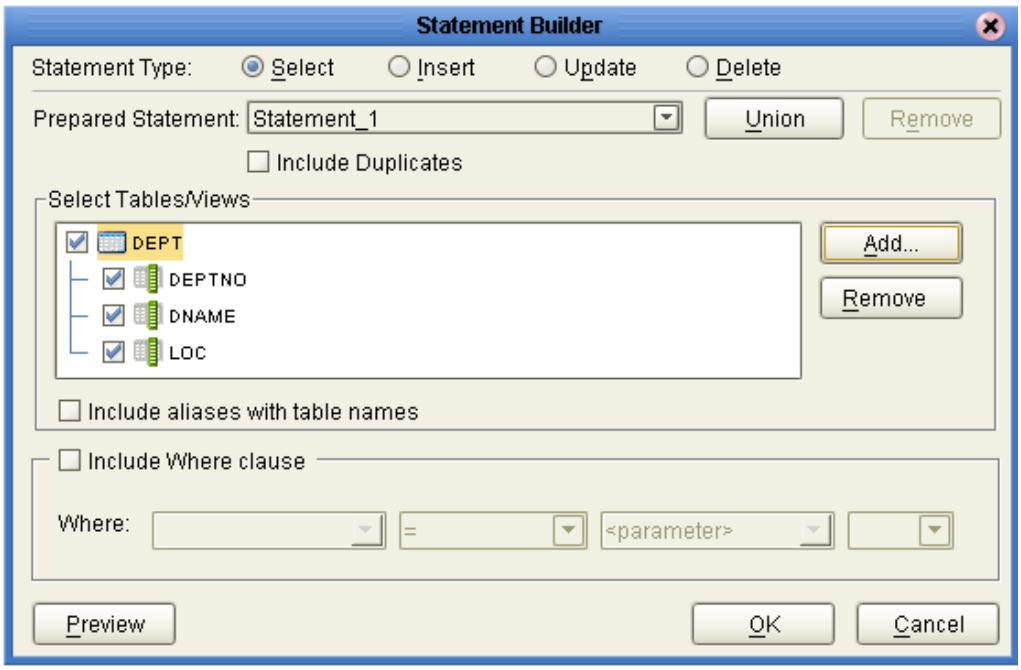


4 In the **Add Tables** dialog box, you can add applicable tables/views to the Statement by searching for a table, highlighting it, and clicking the **Select** button. You can also delete a table/view by highlighting it from the Selected list and clicking **Remove**.

Note: *You can also remove tables/views when you are in the Statement Builder dialog box by checking all applicable boxes and clicking* ***Remove****.*

5 Click **OK** when you have finished selecting the tables/views. The **Statement Builder** dialog box is displayed with all of your selections (as shown in Figure 26).

Figure 26 Statement Builder with Table Selections



6 Select the **Include aliases with table names** checkbox, if applicable.

7 Select the **Include Where clause** checkbox to enable the fields for creating Where clauses.

   A Select the field name to be used in the Where clause.

   B Select the applicable condition (for example +, >, <=, etc.).

   C Set the parameter.

   D Select the **And** or **Or** condition that enables you to create more Where clauses, if needed.

   You can create as many Where clauses to the Statement as needed.

8 Click **OK**. The **Add Prepared Statement** dialog box appears again, displaying the SQL statement you have created. If you need to make any changes to the statement, you can click the **Statement Builder** button again or change the statement in the dialog box.

9 Click **OK**. The Prepared Statement you constructed appears in the Oracle Database wizard. Click the **Edit** button to make changes to the Prepared Statement. If you only need to change parameters or ResultSet Columns, you can use the respective edit buttons.

10 Click the **Next** button. The **Specify the OTD Name** step of the Oracle Database wizard appears.

11 Name the OTD and click **Next**. The **Review your Selections** step appears.

Once you are satisfied with your selections, click **Finish**. Once the OTD is generated, you no longer have the ability to edit the Prepared Statement.

4.4 Editing Existing OTDs

A single OTD can consist of many Database objects. They can be a mixture of **Tables**, **Prepared Statements** and **Stored Procedures**. By using the Database OTD Wizard, the OTD Edit feature allows you to:

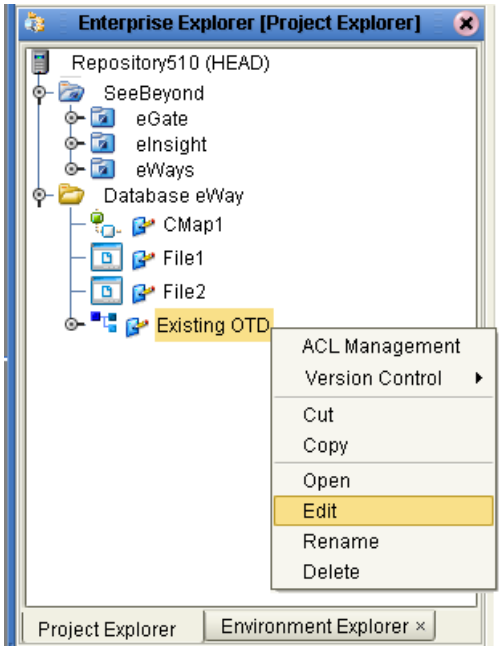
- Add or Remove **Table/Views**.
- Change data types by selecting a different one from a list.
- Add or Remove columns from a **Table** object.
- Add or Remove **Prepared Statement** objects.
- Edit **Prepared Statement** objects.
- Add or Remove **Stored Procedure** objects.
- Edit **Stored Procedure Resultsets**.

To Edit an Existing OTD

When a minor change is needed for an OTD, there is no need to rebuild it from scratch; instead, you can edit the OTD. To edit an OTD, complete the following steps:

1. In the Enterprise Explorer, right-click on the OTD. From the submenu, click **Edit** (see Figure 27). The Database Connection Information Wizard opens.

Figure 27 OTD Edit Menu Item



2. Connect to the Oracle database by entering the applicable information in the wizard. Once the connection is established, the Database Wizard opens, allowing you to make modifications to the OTD.
3. Once you have completed editing the OTD, click the **Finish** button to save the changes.

Caution: *Once the OTD has been edited, you must verify that the changes are reflected in the Collaboration so that no errors occur at runtime. For example, if during the edit process, you delete a database object that is included in a Collaboration, the Collaboration could fail at activation or run-time.*

When editing an OTD, you can connect to another instance of the database under the following conditions:

- The same version of the database should be used unless the newer version is compatible with the older version.
- Tables in the database must be defined with the same definition.
- The stored procedures must be identical.
- For tables/stored procedures built with 'qualified-name', the schema name for the tables/stored procedures must be identical in both database instances.

Chapter 5

Using Oracle Operations

The database operations used in the Oracle eWay are used to access the Oracle database. Database operations are either accessed through Activities in BPEL, or through methods called from a JCD Collaboration.

What's in This Chapter

5.1 Oracle eWay Database Operations (BPEL)

The Oracle eWay uses a number operations to query the Oracle database. Within a BPEL business process, the Oracle eWay uses BPEL Activities to perform basic outbound database operations, including:

- Insert
- Update
- Delete
- SelectOne
- SelectMultiple
- SelectAll

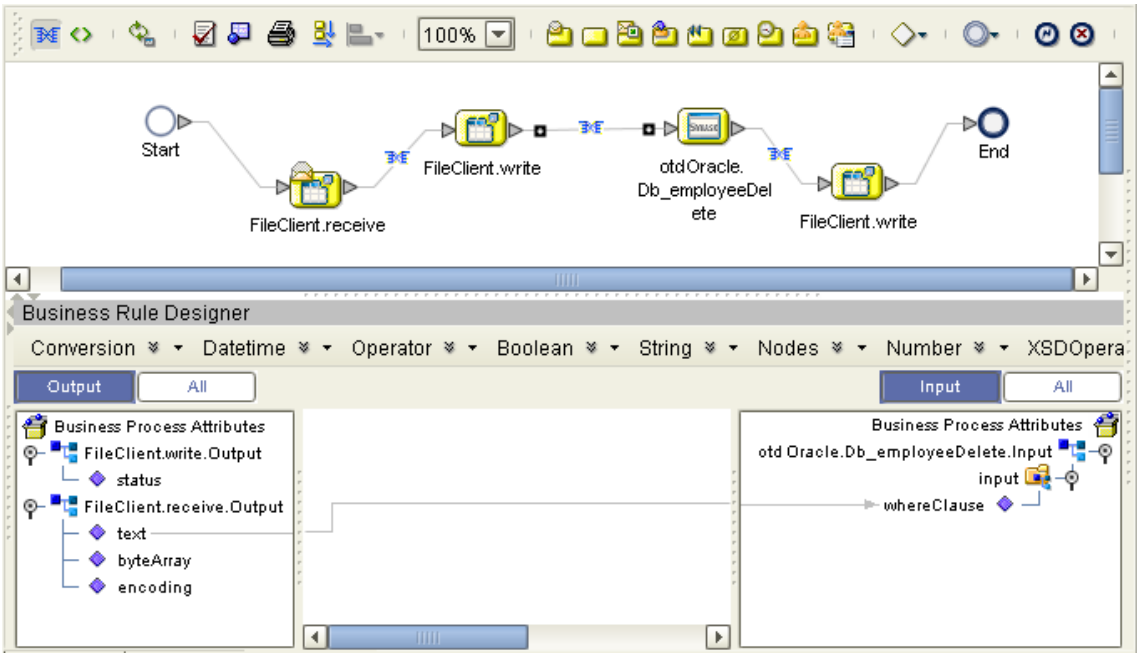
In addition to these outbound operations, the Oracle eWay also employs the inbound Activity **ReceiveOne** within a Prepared Statement OTD.

5.1.1 Activity Input and Output

The Sun SeeBeyond Enterprise Desinger – Business Rules Designer includes Input and Output columns to map and transform data between Activities displayed on the Business Process Canvas.

Figure 28 displays the business rules between the **FileClient.write** and **otdOracle.Db_employeeDelete** Activities. In this example, the **whereClause** appears on the Input side.

Figure 28 Input and Output Between Activities



The following table lists the expected Input and Output of each database operation Activity.

Table 12 Oracle Operations

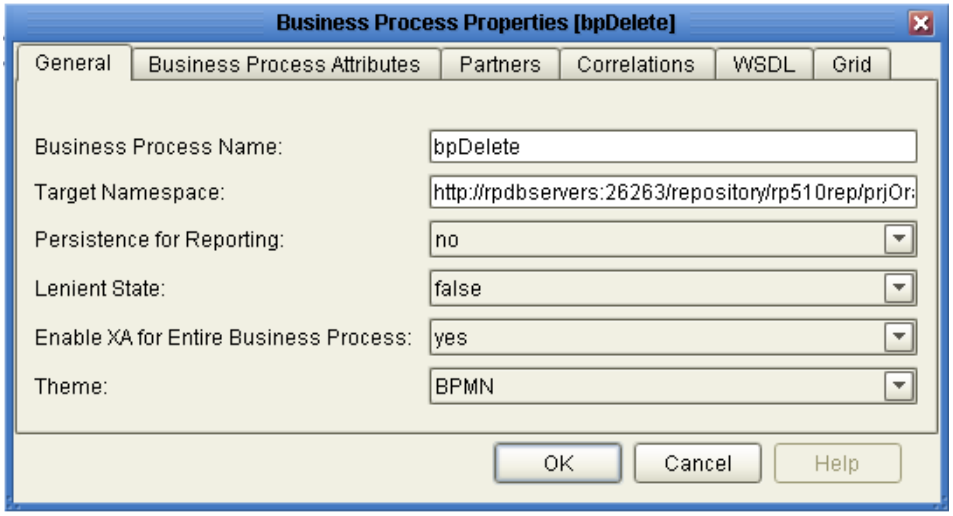
eInsight Operation	Activity Input	Activity Output
SelectAll	where() clause (optional)	Returns all rows that fit the condition of the where() clause
SelectMultiple	number of rows where() clause (optional)	Returns the number of rows specified that fit the condition of the where() clause
SelectOne	where() clause (optional)	Returns the first row that fits the condition of the where() clause
Insert	definition of new item to be inserted	Returns status.
Update	where() clause	Returns status.
Delete	where() clause	Returns status.

5.1.2 Oracle eWay Outbound XA Support for BPEL

To enable XA support for BPEL in the Oracle eWay do the following:

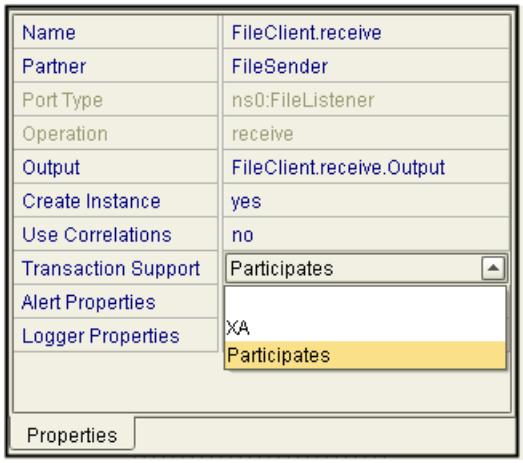
1 In the Business Process properties, set the **Enable XA for the Entire Business Process** field to **Yes** (see Figure 29).

Figure 29 Business Process Properties



2 For all needed activities in the Business Process, set the **Transaction Support** field to **Participates** (see Figure 30).

Figure 30 Transaction Support



Note: *For more information on XA support, refer to the **Sun SeeBeyond eInsight Business Process Manager User's Guide**.*

5.2 Oracle eWay Database Operations (JCD)

The same database operations are also used in the JCD, but appear as methods to call from the Collaboration.

Tables, Views, and Stored Procedures are manipulated through OTDs. Methods to call include:

- insert()

- insertRow()
- update(*String sWhere*)
- updateRow()
- delete(*String sWhere*)
- deleteRow()
- select(*String where*)

Note: *Refer to the Javadoc for a full description of methods included in the Oracle eWay.*

5.2.1 The Table

A table OTD represents a database table. It consists of fields and methods. Fields correspond to the columns of a table while methods are the operations that you can apply to the OTD. This allows you to perform query, update, insert, and delete SQL operations in a table.

By default, the Table OTD has UpdatableConcurrency and ScrollTypeForwardOnly. The type of result returned by the select() method can be specified using:

- SetConcurrencytoUpdatable
- SetConcurrencytoReadOnly
- SetScrollTypetoForwardOnly
- SetScrollTypetoScrollSensitive
- SetScrollTypetoInsensitive

The Query (Select) Operation

To perform a query operation on a table

1. Execute the **select()** method with the "**where**" clause specified if necessary.

Note: *The content of the **input.getText()** file may contain null, meaning it will not have a "**where**" clause or it can contain a "**where**" clause such as **empno > 50**.*

2. Loop through the ResultSet using the **next()** method.
3. Process the return record within a **while()** loop.

For example:

```
package prjOracle_JCDjcdALL;


public class jcdTableSelect
{

    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;
```

```
    public com.stc.codegen.util.CollaborationContext
collabContext;

    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
dtd.otdOutputDTD1325973702.DB_Employee otdOutputDTD_DB_Employee_1,
otdOracle.OtdOracleOTD otdOracle_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
       FileClient_1.setText( "Selectiong records from db_employee
table via Table Select........" );
        FileClient_1.write();
        otdOracle_1.getDb_employee().select( input.getText() );
        while (otdOracle_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.shortToString(
otdOracle_1.getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdOracle_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdOracle_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate(
otdOracle_1.getDb_employee().getRATE().toString() );
            otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdOracle_1.getDb_employee().getLAST_UPDATE(), "yyyy-MM-dd
hh:mm:ss", false, "" ) );
            FileClient_1.setText(
otdOutputDTD_DB_Employee_1.marshalToString() );
            FileClient_1.write();
        }
        FileClient_1.setText( "Table Select Done." );
        FileClient_1.write();
    }

}
```

The Insert Operation

To perform an insert operation on a table

1. Execute the **insert()** method. Assign a field.
2. Insert the row by calling **insertRow()**

   This example inserts an employee record.

```
package prjOracle_JCDjcdALL;


public class jcdInsert
{

    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;

    public com.stc.codegen.util.CollaborationContext collabContext;

    public com.stc.codegen.util.TypeConverter typeConverter;
```

```
    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
otdOracle.OtdOracleOTD otdOracle_1,
dtd.otdInputDTD_1206505729.DB_Employee otdInputDTD_DB_Employee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Inserting records in to db_employee
table......" );
        FileClient_1.write();
        otdInputDTD_DB_Employee_1.unmarshalFromString(
input.getText() );
        otdOracle_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdOracle_1.getDb_employee().setEMP_NO(
typeConverter.stringToShort(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdOracle_1.getDb_employee().setLAST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdOracle_1.getDb_employee().setFIRST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdOracle_1.getDb_employee().setRATE( new
java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1
).getRate() ) );
            otdOracle_1.getDb_employee().setLAST_UPDATE(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdOracle_1.getDb_employee().insertRow();
        }
        FileClient_1.setText( "Insert Done." );
        FileClient_1.write();
    }

}
```

The Update Operation

To perform an update operation on a table

1. Execute the **update()** method.

Note: *The content of the **input.getText()** file may contain null, meaning it will not have a "**where**" clause or it can contain a "**where**" clause such as **empno > 50**.*

2. Using a while loop together with **next()**, move to the row that you want to update.
3. Assign updating value(s) to the fields of the table OTD
4. Update the row by calling **updateRow()**.

```
package prjOracle_JCDjcdALL;


public class jcdUpdate
{

    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;
```

```
    public com.stc.codegen.util.CollaborationContext collabContext;

    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
otdOracle.OtdOracleOTD otdOracle_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Updating the Rate and Last_update
fields .. " );
        FileClient_1.write();
        otdOracle_1.getDb_employee().update( input.getText() );
        while (otdOracle_1.getDb_employee().next()) {
         otdOracle_1.getDb_employee().setLAST_NAME( "Krishna" );
         otdOracle_1.getDb_employee().setFIRST_NAME( "Kishore" );
         otdOracle_1.getDb_employee().updateRow();
        }
        FileClient_1.setText( "Update Done." );
        FileClient_1.write();
    }

}
```

The Delete Operation

To perform a delete operation on a table

1 Execute the **delete()** method.

Note: *The content of the **input.getText()** file may contain null, meaning it will not have a "**where**" clause or it can contain a "**where**" clause such as **empno > 50**.*

In this example DELETE an employee.

```
package prjOracle_JCDjcdALL;

public class jcdDelete
{
    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;

    public com.stc.codegen.util.CollaborationContext
collabContext;

    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
otdOracle.OtdOracleOTD otdOracle_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Deleting record....." );
        FileClient_1.write();
        otdOracle_1.getDb_employee().delete( input.getText() );
        FileClient_1.setText( "Delete Done." );
        FileClient_1.write();
    }
```

```
}
```

5.2.2 The Stored Procedure

A Stored Procedure OTD represents a database stored procedure. Fields correspond to the arguments of a stored procedure while methods are the operations that you can apply to the OTD. It allows you to execute a stored procedure. In the Collaboration Editor you can assign values to the input parameters, execute the call, collect data, and retrieve the values from output parameters.

Executing Stored Procedures

The OTD used in the example below, contains a Stored Procedure with input parameters. These input parameters are generated by the Database OTD Wizard and are displayed in the Collaboration Editor as subnodes of the OTD.

Below are the steps for executing the Stored Procedure:

1. Specify the input values.
2. Execute the Stored Procedure.
3. Retrieve the output parameters if any.

For example:

```
package Storedprocedure;

public class sp_jce
{

    public com.stc.codegen.logger.Logger logger;

    public com.stc.codegen.alerter.Alerter alerter;

    public void receive(
com.stc.connector.appconn.file.FileTextMessage
input,com.stc.connector.appconn.file.FileApplication
FileClient_1,employeedb.Db_employee
employeedb_with_top_db_employee_1,insert_DB.Insert_DBOTD insert_DB_1
)
    throws Throwable
    {
        employeedb_with_top_db_employee_1.unmarshalFromString(
input.getText() );

insert_DB_1.getInsert_new_employee().setEmployee_no(
java.lang.Integer.parseInt(
employeedb_with_top_db_employee_1.getEmployee_no() ) );

        insert_DB_1.getInsert_new_employee().setEmployee_Lname(
employeedb_with_top_db_employee_1.getEmployee_lname() );

        insert_DB_1.getInsert_new_employee().setEmployee_Fname(
employeedb_with_top_db_employee_1.getEmployee_fname() );

        insert_DB_1.getInsert_new_employee().setRate(
java.lang.Float.parseFloat(
employeedb_with_top_db_employee_1.getRate() ) );
```

```
        insert_DB_1.getInsert_new_employee().setUpdate_date(
java.sql.Timestamp.valueOf(
employeedb_with_top_db_employee_1.getUpdate_date() ) );

        insert_DB_1.getInsert_new_employee().execute();

        insert_DB_1.commit();

        FileClient_1.setText( "procedure executed" );

        FileClient_1.write();
    }

}
```

Manipulating the ResultSet and Update Count Returned by Stored Procedure

The following methods are provided for using the ResultSet and Update Count when they are returned by Stored Procedures:

- enableResultSetOnly
- enableUpdateCountsOnly
- enableResultSetandUpdateCounts
- resultsAvailable
- next
- getUpdateCount
- available

Note: *Stored Procedure ResultSets are supported in Java collaborations only.*

Oracle stored procedures do not return records as ResultSets; instead, the records are returned through output reference cursor parameters. Reference Cursor parameters are essentially ResultSets.

The **resultsAvailable()** method, added to the OTD, simplifies the whole process of determining whether any results, whether they are update Counts or ResultSets, are available after a stored procedure has been executed. Although JDBC provides three methods (**getMoreResults()**, **getUpdateCount()**, and **getResultSet()**) to access the results of a stored procedure call, the information returned from these methods can be quite confusing to the inexperienced Java JDBC programmer and they also differ between vendors. You can simply call **resultsAvailable()** and if Boolean true is returned, you can expect either a valid Update Count when get**UpdateCount()** is called and/or the next ResultSet has been retrieved and made available to one of the ResultSet nodes defined for the Stored Procedure OTD, when that node's available() method returns true.

Update Counts information that is returned from Stored Procedures is often insignificant. Process returned ResultSet information and avoid looping through all of the Update Counts. The following three methods control exactly what information is

returned from a stored procedure call. The **enableResultSetsOnly()** method, added to the OTD allows only ResultSets to be returned and thus every **resultsAvailable()** called only returns Boolean true if a ResultSet is available. Likewise, the **enableUpdateCountsOnly()** method causes **resultsAvailable()** to return true only if an Update Count is available. The default case of the **enableResultSetsAndUpdateCount()** method enables both ResultSets and Update Counts to be returned.

Collaboration usability for a Stored Procedure ResultSet

You can use your mouse to drag and drop the Column data of the ResultSets from their OTD nodes to the Business Rules. Below is a code snippet that can be generated by the Collaboration Editor:

```
// resultsAvailable() true if there's an update count and/or a result
set available.
// note, it should not be called indiscriminantly because each time
the results pointer is
// advanced via getMoreResults() call.
while (getSPIn().getSpS_multi().resultsAvailable())
{
    // check if there's an update count
    if (getSPIn().getSpS_multi().getUpdateCount() > 0)
    {
        logger.info("Updated
"+getSPIn().getSpS_multi().getUpdateCount()+" rows");
    }
// each result set node has an available() method (similar to OTD's)
that tells the user
// whether this particular result set is available. note, JDBC does
support access to
// more than one result set at a time, i.e., cannot drag from two
distinct result sets
// simultaneously
    if (getSPIn().getSpS_multi().getNormRS().available())
    {
    while (getSPIn().getSpS_multi().getNormRS().next())
    {
    logger.info("Customer Id =
"+getSPIn().getSpS_multi().getNormRS().getCustomerId());
    logger.info("Customer Name =
"+getSPIn().getSpS_multi().getNormRS().getCustomerName());
    }
    if (getSPIn().getSpS_multi().getDbEmployee().available())
    {
    while (getSPIn().getSpS_multi().getDbEmployee().next())
    {
    logger.info("EMPNO =
"+getSPIn().getSpS_multi().getDbEmployee().getEMPNO());
    logger.info("ENAME =
"+getSPIn().getSpS_multi().getDbEmployee().getENAME());
    logger.info("JOB =
"+getSPIn().getSpS_multi().getDbEmployee().getJOB());
    logger.info("MGR =
"+getSPIn().getSpS_multi().getDbEmployee().getMGR());
    logger.info("HIREDATE =
"+getSPIn().getSpS_multi().getDbEmployee().getHIREDATE());
    logger.info("SAL =
"+getSPIn().getSpS_multi().getDbEmployee().getSAL());
    logger.info("COMM =
"+getSPIn().getSpS_multi().getDbEmployee().getCOMM());
```

```
    logger.info("DEPTNO =
"+getSPIn().getSpS_multi().getDbEmployee().getDEPTNO());
    }
}
```

Note: *resultsAvailable() and available() cannot be indiscriminately called because each time they move ResultSet pointers to the appropriate locations.*

Once the "**resultsAvailable()**" method has been called, the next result (if available) can be either a **ResultSet** or an **UpdateCount**, if the default "**enableResultSetsAndUpdateCount()**" was used.

Because of limitations imposed by some DBMSs, SeeBeyond recommends that for maximum portability, all of the results in a ResultSet object should be retrieved before OUT parameters are retrieved. Therefore, you must retrieve all ResultSet(s) and update counts first, followed by retrieving the OUT type parameters and return values.

The following list includes specific ResultSet behavior that you may encounter:

- The method **resultsAvailable()** implicitly calls **getMoreResults()** when it is called more than once. Do not call both methods in your Java code. If you do, there may be skipped data from one of the ResultSets when more than one ResultSet is present.
- The methods **available()** and **getResultSet()** cannot be used when mulitple ResultSets are open at the same time. Attempting to open more the one ResultSet at the same time closes the previous ResultSet. The recommended working pattern is:
  - Open one Result Set, ResultSet_1 and work with the data until you have completed your modifications and updates. Open ResultSet_2, (ResultSet_1 is now closed) and modify. When you have completed your work in ResultSet_2, open any additional ResultSets or close ResultSet_2.
- If you modify the ResultSet generated by the Execute mode of the Database Wizard, you need to make sure that the indexes match the stored procedure; if you do this, your ResultSet indexes are preserved.
- Generally, you do not need to call getMoreResults; you need to call it only if you do not want to use our enhanced methods and you want to follow the traditional JDBC calls on your own.

5.3 Oracle Table Data Types

Oracle tables support the following data types:

- **Real** - an approximate numeric data type.
- **Float** - a data type where all platforms have values of the least specified minimum precision.
- **CLOB** - a built-in data type that stores a Character Large Object as a column value in a row of a database table.

For all others, use the data types Float, Double, or CLOB and build them using a data type of “Other”.

Note: *The Oracle driver does not support the boolean and PL/SQL RECORD datatypes in the Function and Stored Procedure.*

Long RAW for Prepared Statements and Stored Procedure support:

The following two parameters must be set prior to the Insert/Update/Delete statement.

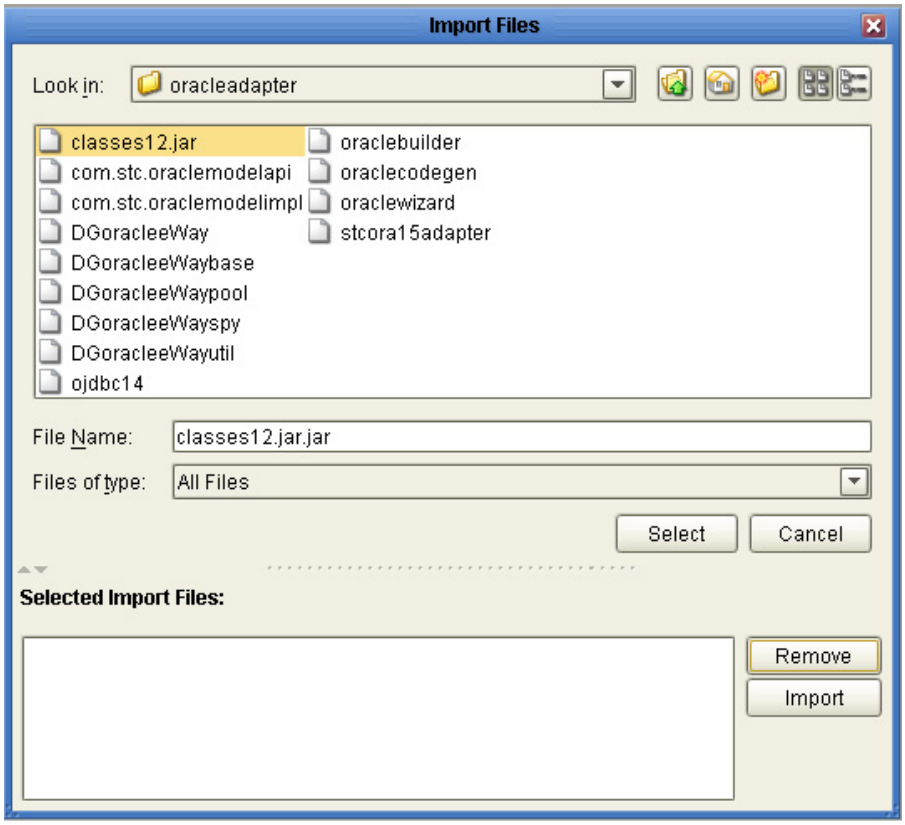
```
setConcurrencyToReadOnly()
setScrollTypeToForwardOnly()
```

5.3.1 Using CLOBs

To use a CLOB in the Oracle eWay, do the following:

1. In the Enterprise Designer, right-click on the project, select **Import**. From the submenu, select **File**. The Import File dialog box appears.
2. From `<Client_eDesigner>\usrdir\modules\ext\oracleadapter`, create a copy of the **`ojdbc14.jar`** in the directory and rename it **`classes12.jar`**.
3. Navigate to the **classes12.jar file**, `<Client_eDesigner>\usrdir\modules\ext\oracleadapter\`**`classes12.jar`** using the Enterprise Designer's Project File Import feature (see Figure 31).

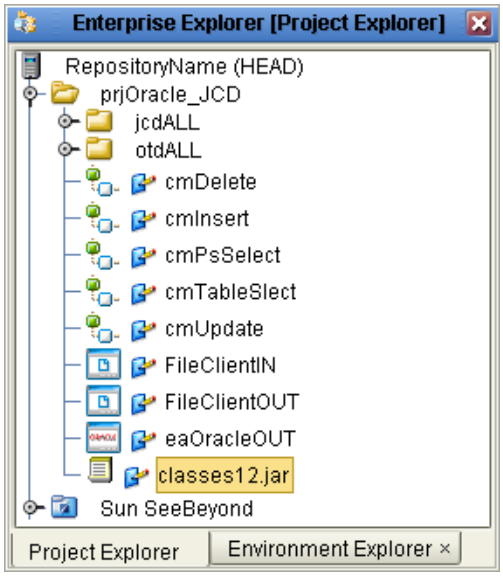
Figure 31 Importing Classes12.jar



4 Click **Select**.

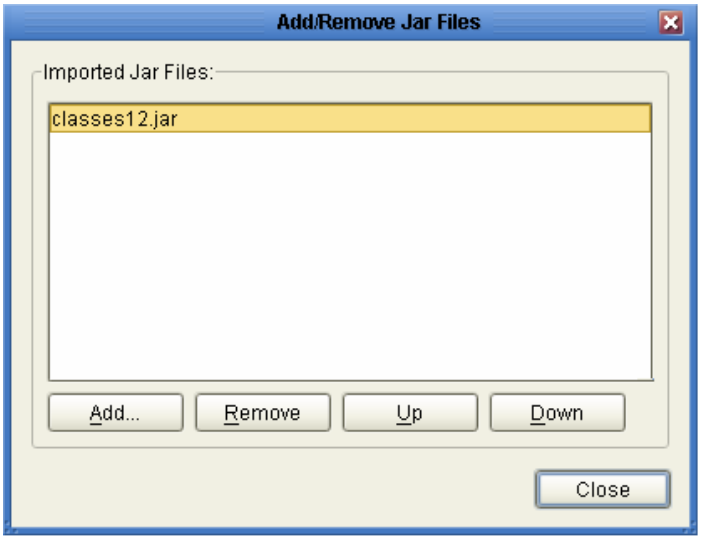
5 Click **Import**. The **classes12.jar** file appears, as shown in Figure 32.

Figure 32 Classes12.jar in a Project



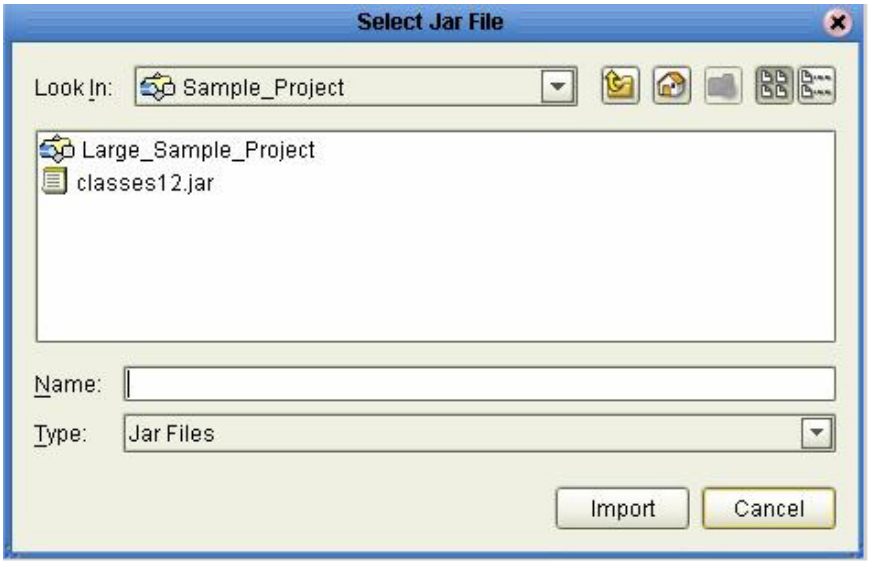
6 To load the **classes12.jar** file into your Java Collaboration, select the **Import JAR File** button. Click **Add** in the **Add/Remove Jar Files** window to add the Jar files (see Figure 33).

Figure 33 Add/Remove Jar Files



7 In the **Select Jar File** window, select the **classes12.jar** file and click **Import** (see Figure 34).

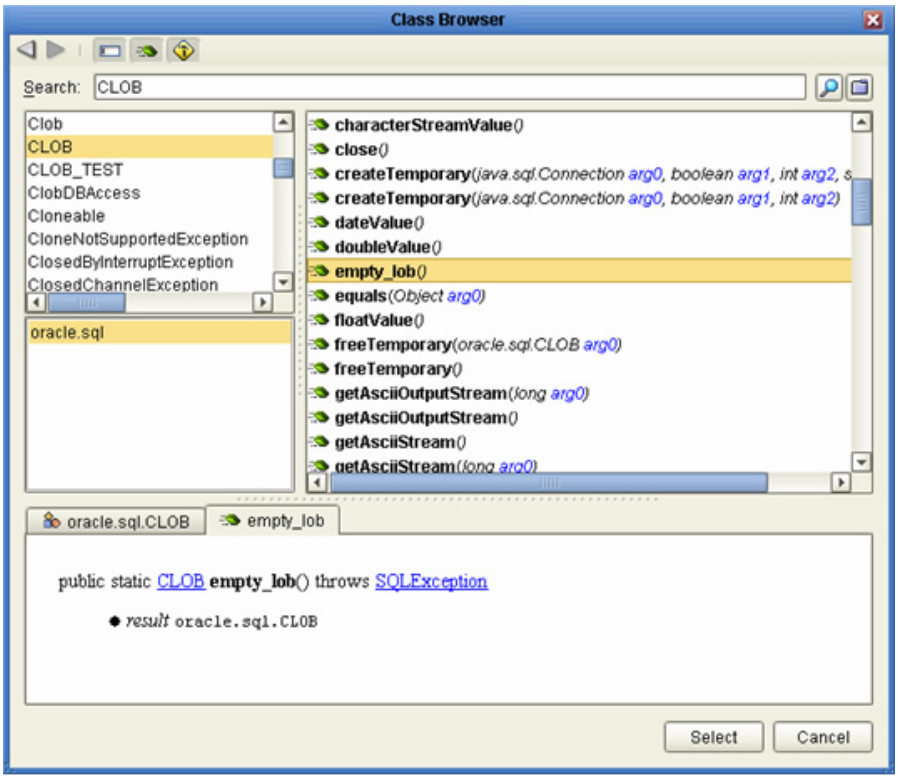
Figure 34 Select Jar File



8 In the **Add/Remove Jar Files** window, click **Close**.

9 In the Business Rules Designer, call the CLOB method by clicking the **Class Browser** button. The **Class Browser** dialog box appears (see Figure 35).

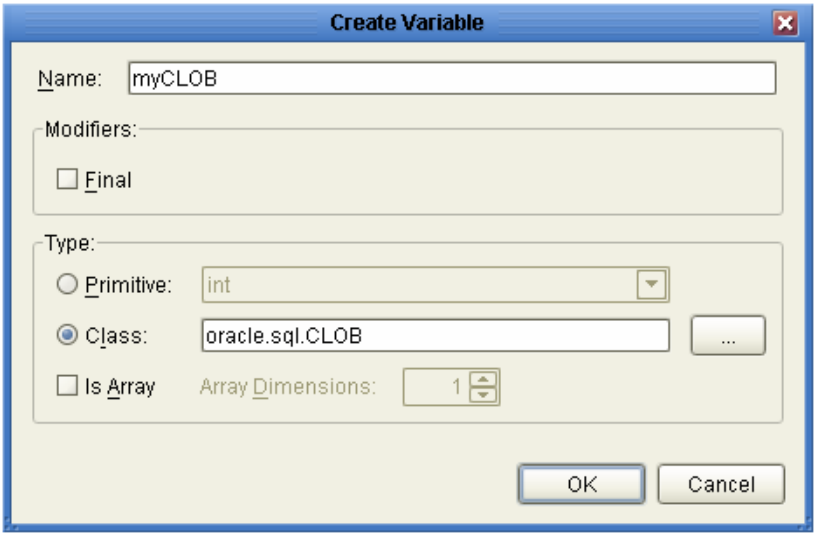
Figure 35 Call the CLOB Method



10 Select **empty_lob** from the list of CLOB variables and click **Select**.

11 Create a local variable by clicking the **Local Variable** button on the Business Rules toolbar. The **Create Variable** dialog box is displayed. (see Figure 36).

Figure 36 Create a Local Variable

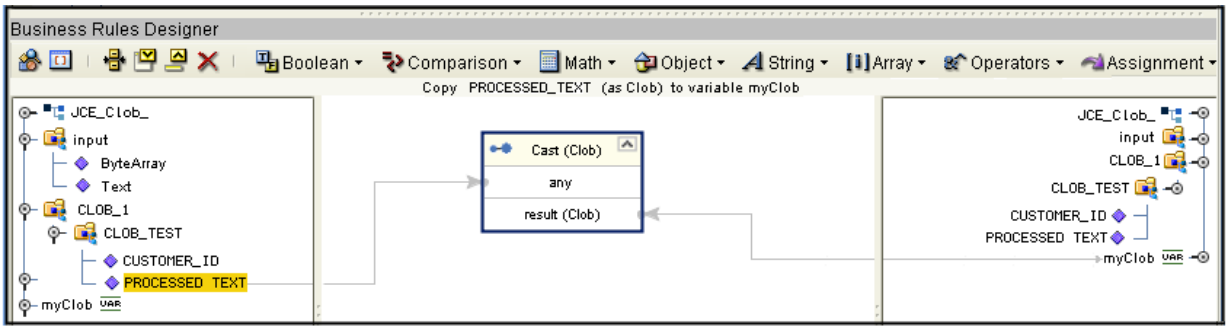


12 Name the variable **myCLOB**, select the Class type, and choose CLOB as the Class type.

13 Click **OK** to create the variable.

14 In the Business Rules Designer, drag the **CLOB** to the Local Variable using the Cast method. Click **Yes** when the incompatible Data Type warning appaers (see Figure 37).

Figure 37 Drag CLOB to the Local Variable



15 Use the CLOB putString method to assign 1 to Arg().

16 In the Java Collaboration Editor, the Java code resembles the following:

```
public void receive( com.stc.connector.appconn.file.FileTextMessage
input,cLOB.CLOBOTD CLOB_1 )
throws Throwable
{
    //@map:CLOB_1.getCLOB_TEST.insert
        CLOB_1.getCLOB_TEST().insert();

        //@map:Copy java.math.BigDecimal.valueOf(100) to CUSTOMER_ID
```

```
        CLOB_1.getCLOB_TEST().setCUSTOMER_ID(
java.math.BigDecimal.valueOf( 100 ) );

        //@map:Copy oracle.sql.CLOB.empty_lob to PROCESSED_TEXT
        CLOB_1.getCLOB_TEST().setPROCESSED_TEXT(
oracle.sql.CLOB.empty_lob() );

        //@map:CLOB_TEST.insertRow
        CLOB_1.getCLOB_TEST().insertRow();

//@map:CLOB_1.getCLOB_TEST.select("customer_id = 100 for update")
        CLOB_1.getCLOB_TEST().select( "customer_id = 100 for update"
);
//If
        if (CLOB_1.getCLOB_TEST().next()) {
            //@map:oracle.sql.CLOB myClob;
            oracle.sql.CLOB myClob;

            //@map:Copy cast PROCESSED_TEXT to oracle.sql.CLOB to
myClob
            myClob = (oracle.sql.CLOB)
CLOB_1.getCLOB_TEST().getPROCESSED_TEXT();

           //@map:myClob.putString(1,Text)
            myClob.putString( 1,input.getText() );

            //@map:CLOB_TEST.updateRow
            CLOB_1.getCLOB_TEST().updateRow();
}
    }
```

Chapter 6

Implementing the Oracle eWay Project

This chapter provides an introduction to the Oracle eWay components, and information on how these components are created and implemented in a Sun Java Composite Application Platform Suite Project.

It is assumed that the reader understands the basics of creating a Project using the Sun SeeBeyond Enterprise Designer. For more information on creating an eGate Project, see the *eGate Tutorial* and the *eGate Integrator User's Guide*.

What's in This Chapter

6.1 About the Oracle eWay Sample Projects

The Oracle eWay **Oracle_eWay_Sample.zip** file contains two sample Projects that provide basic instruction on using Oracle operations in the Java Collaboration Definition (JCD), or the Business Process Execution Language (BPEL) Projects.

- **prjOracle_JCD:** demonstrates how to select, insert, update, and delete data from a Oracle database using JCDs.
- **prjOracle_BPEL:** demonstrates how to select, insert, update, and delete data from a Oracle database using a BPEL business process.

Both the **prjOracle_JCD** and **prjOracle_BPEL** sample Projects demonstrate how to:

- Select employee records from the database using a prepared statement.
- Select employee records from the db_employee table.
- Insert employee records data into the db_employee table.
- Update an employee record in the db_employee table.
- Delete an employee record in the db_employee table.

In addition to sample Projects, the **Oracle510_SAMPLE_projects.zip** file also includes six sample input trigger files and ten sample output files (five per sample).

Sample input files include:

- TriggerDelete.in.~in
- TriggerInsert.in.~in (for JCE projects only)
- TriggerBpInsert.in.~in (for BPEL projects only)
- TriggerPsSelect.in.~in
- TriggerTableSelect.in.~in
- TriggerUpdate.in.~in

Sample output JCD files include:

- JCD_Delete_output0.dat
- JCD_Insert_output0.dat
- JCD_PsSelect_output0.dat
- JCD_TableSelect_output0.dat
- JCD_Update_output0.dat

Sample output BPEL files include:

- BPEL_Delete_output0.dat
- BPEL_Insert_output0.dat
- BPEL_PsSelect_output0.dat
- BPEL_TableSelect_output0.dat
- BPEL_Update_output0.dat

6.1.1 Operations Used in the Oracle Sample Projects

The following database operations are used in both BPEL and JCD sample Projects:

- Insert
- Update
- Delete
- Select (SelectAll as a BPEL Activity)

Assigning Operations in JCD

Database operations are listed as methods in the JCD. Perform the following steps to access these methods:

1. Create a Collaboration that contains a database OTD created from the Oracle database.
2. Right-click the OTD listed in your Collaboration and then select **Select Method to Call** from the shortcut menu.

3 Browse to and select a method to call.

Assigning Operations in BPEL

You can associate an eInsight Business Process Activity with the eWay, both during the system design phase and during run time. To make this association:

1 Select the desired **receive** or **write** operation under the eWay in the Enterprise Explorer.

2 Drag the operation onto the eInsight Business Process canvas.

The operation automatically changes to an Activity with an icon identifying the component that is the basis for the Activity.

At run time, the eInsight engine invokes each step in the order that you defined in the Business Process. Using the engine's Web Services interface, the Activity in turn invokes the eWay. You can open a file specified in the eWay and view its contents before and after the Business Process is executed.

Note: *Inbound database eWays are only supported within BPEL Collaborations.*

6.1.2 About the eInsight Engine and eGate Components

You can deploy an eGate component as an Activity in an eInsight Business Process. Once you have associated the desired component with an Activity, the eInsight engine can invoke it using a Web Services interface.

Examples of eGate components that can interface with eInsight in this way are:

- Object Type Definitions (OTDs)
- An eWay
- Collaborations

Using the eGate Enterprise Designer and eInsight, you can add an Activity to a Business Process, then associate that Activity with an eGate component, for example, an eWay. When eInsight run the Business Process, it automatically invokes that component via its Web Services interface.

6.2 Steps Required to Run the Sample Projects

The following steps are required to run the sample projects that are contained in the **OracleeWayDocs.sar** file.

1 Run the SQL script.

This creates the tables and records required by the sample Project.

2 Import the sample Projects.

3 Build, deploy, and run the sample Projects.

You must do the following before you can run an imported sample Project:

- Create an Environment
- Configure the eWays
- Create a Deployment Profile
- Create and start a domain
- Deploy the Project

4 Check the output.

6.3 Running the SQL Script

The data used for both the **JCD** and **BPEL** sample Projects are contained within a table called **db_employee**. You create this table by using an external tool, such as SQL Plus, to run the SQL statement **Oracle_sample_script.sql** that is included in the sample Project.

Following is the SQL statement designed for the sample Projects.

```
drop table db_employee
go
create table db_employee (
  EMP_NO int,
  LAST_NAME varchar(30),
  FIRST_NAME varchar(30),
  RATE float,
  LAST_UPDATE datetime)
go
```

The sample Projects provided with the Oracle eWay use input files to pass predefined data or conditions into the Collaboration or BPEL business process, which then transforms the database contents, and delivers the result set.

6.4 Importing a Sample Project

Sample eWay Projects are included as part of the installation disk package. To import a sample eWay Project to the Enterprise Designer do the following:

1 Extract the samples from the Enterprise Manager to a local file.

Sample files are uploaded with the eWay's documentation SAR file, and then downloaded from the Enterprise Manager's Documentation tab. The **Oracle_eWay_Sample.zip** file contains the various sample Project ZIP files.

Note: *Make sure you save all unsaved work before importing a Project.*

2 From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import Project** from the shortcut menu. The **Import Manager** appears.

3 Browse to the directory that contains the sample Project ZIP file. Select the sample file and click **Import**.

4 Click **Close** after successfully importing the sample Project.

6.5 Building and Deploying the prjOracle_BPEL Sample Project

The following provides step-by-step instructions for manually creating the **prjOracle_BPEL** sample Project.

Steps required to create the sample project include:

6.5.1 Creating a Project

The first step is to create a new Project in the Sun SeeBeyond Enterprise Designer.

1 Start the Enterprise Designer.

2 From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.

3 Click twice on **Project1** and rename the Project (for this sample, **prjOracle_BPEL**).

6.5.2 Creating the OTDs

The sample Project requires three OTDs to interact with the Oracle eWay. These OTDs include:

- Oracle Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a Oracle Database OTD include:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Object Type Definition**.

   The New Object Type Definition Wizard window appears.
2. Select the **Oracle Database OTD Wizard** from the list of OTD Wizards and click **Next**.
3. Enter the connection information for the Oracle database. Connection fields include:
   - Host name:
   - Port ID:
   - SID:
   - User name:
   - Password:
4. Click **Next**, and select the types of database object you want to include in the sample Project. For this example, select the following:
   - Tables/Views/Aliases
   - Prepared Statements
5. Click **Add** to select tables from the Oracle database. The **Add Tables** window appears.
6. Search for or Type in the name of the database. In this example we use the **DB_EMPLOYEE** table. Click **Select** when the database appears in the Results selection frame. Click **OK** to close the Add Tables window
7. Click **Next the Add Prepared Statements Wizard appears.**
8. Click **Add**, the Add Prepared Statement window appears. Enter the following:
   - Prepared Statement Name: Select_ps
   - SQL Statement:

```
select * from db_employee where emp_no > ? order by emp_no
```

Note: *In our example, the SQL statement includes the* **?** *placeholder for input. This placeholder represents the Where Clause.*

9. Click the **OK** button to close the Prepared Statement window, and then click **Next** on the Prepared Statements Wizard window.
10. Enter an OTD name. In this example, we use **otdOracle**.
11. Click **Next** and review your settings, then click **Finish** to create the OTD.

Steps required to create inbound and outbound DTD OTDs include:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Object Type Definition**.

The New Object Type Definition Wizard window appears.

2 Select **DTD** from the list of OTD Wizards and click **Next**.

3 Browse to and then select a DTD file. For our example, select one of the following DTD files from the sample Project, and then click **Next**.

- otdInputDTD.dtd
- otdOutputDTD.dtd

4 The file you select appears in the Select Document Elements window. Click **Next**.

5 Click **Finish** to complete the DTD based OTD. Repeat this process again to create the second DTD file.

6.5.3 Creating the Business Process

Steps required to create the Business Process include:

- Creating the business process flow
- Configuring the modeling elements

Creating the Business Process Flow

The business process flow contains all the BPEL elements that make up a business process.

Steps to create a business process flow include:

1 Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename **BusinessProcess1** to **bpInsert**.

2 Create four additional business processes and rename them as follows:

- bpUpdate
- bpDelete
- bpPsSelect
- bpTableSelect

3 Add the following activities to the Business Process Designer canvas.

Business Process	Activity
bpInsert	▪ FileClient.Receive ▪ FileClient.Write ▪ FileClient.Write ▪ otdOracle.DB_EMPLOYEEInsert (inside a Scope) ▪ otdInputDTD_DBemployees.unmarshal

Business Process	Activity
bpUpdate	▪ FileClient.receive ▪ FileClient.write ▪ otdOracle.DB_EMPLOYEEUpdate ▪ FileClient.write
bpDelete	▪ FileClient.receive ▪ FileClient.write ▪ otdOracle.DB_EMPLOYEEDelete ▪ FileClient.write
bpPsSelect	▪ FileClient.receive ▪ FileClient.write ▪ otdOracle.Select_psPSSelectAll ▪ Decision ▪ FileClient.write (inside a Scope renamed "No record") ▪ otdInputDTD_DBemployees.marshal (inside a Scope renamed "Records found") ▪ FileClient.write (inside a Scope renamed "Records found") ▪ FileClient.write
bpTableSelect	▪ FileClient.receive ▪ FileClient.write ▪ otdOracle.DB_EMPLOYEESelectAll ▪ otdInputDTD_DBemployees.marshal ▪ FileClient.write ▪ FileClient.write

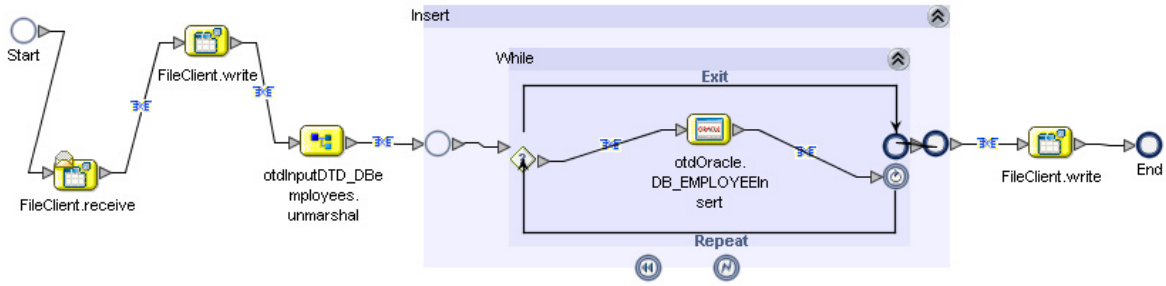
Configuring the bpInsert Modeling Elements

Business Rules, created between the Business Process Activities, allow you to configure the relationships between the input and output Attributes of the Activities using the Business Process Designer's Business Rule Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Insert operation. See Figure 38 for an illustration of how all the modeling elements appear when connected.

Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

Figure 38 bpInsert Business Process

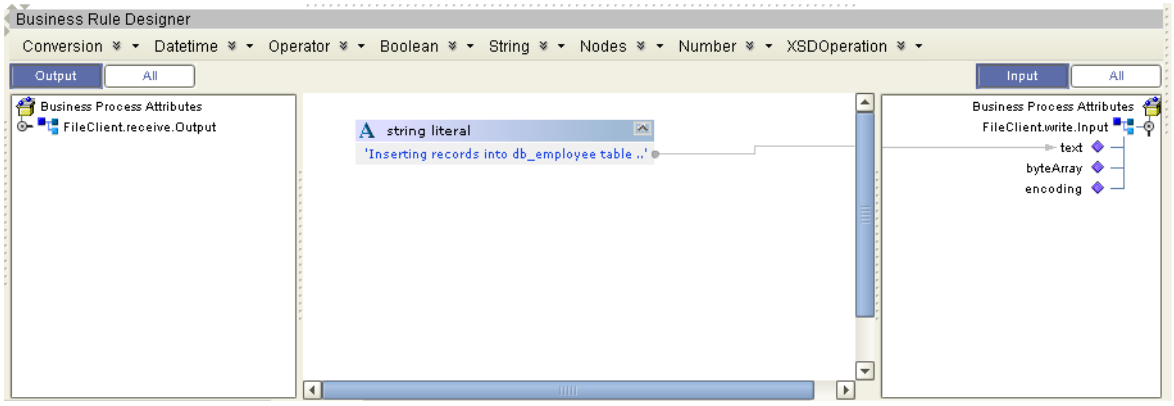


Steps required to configure the bpInsert business process:

Configure the following business rules in the bpInsert business process.

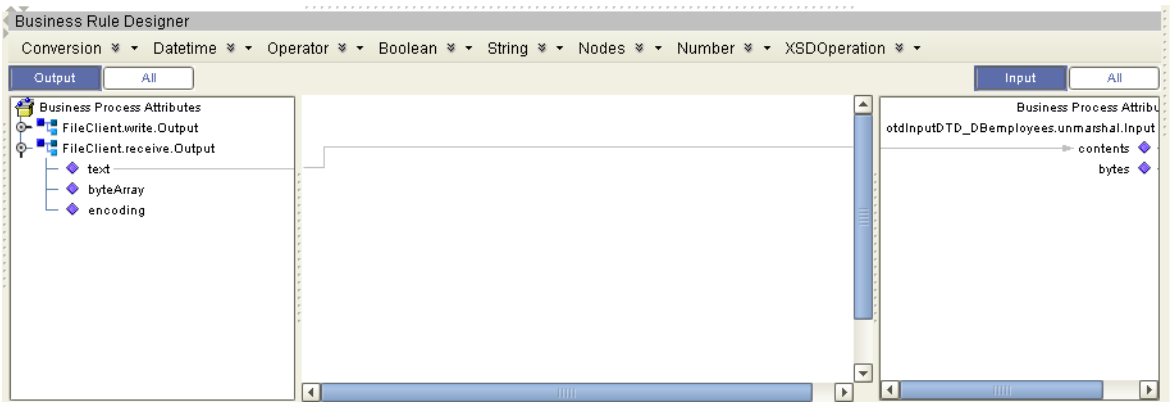
1 Configure the business rule between **FileClient.receive** and **FileClient.write** as seen in Figure 39.

Figure 39 bpInsert Business Rule # 1



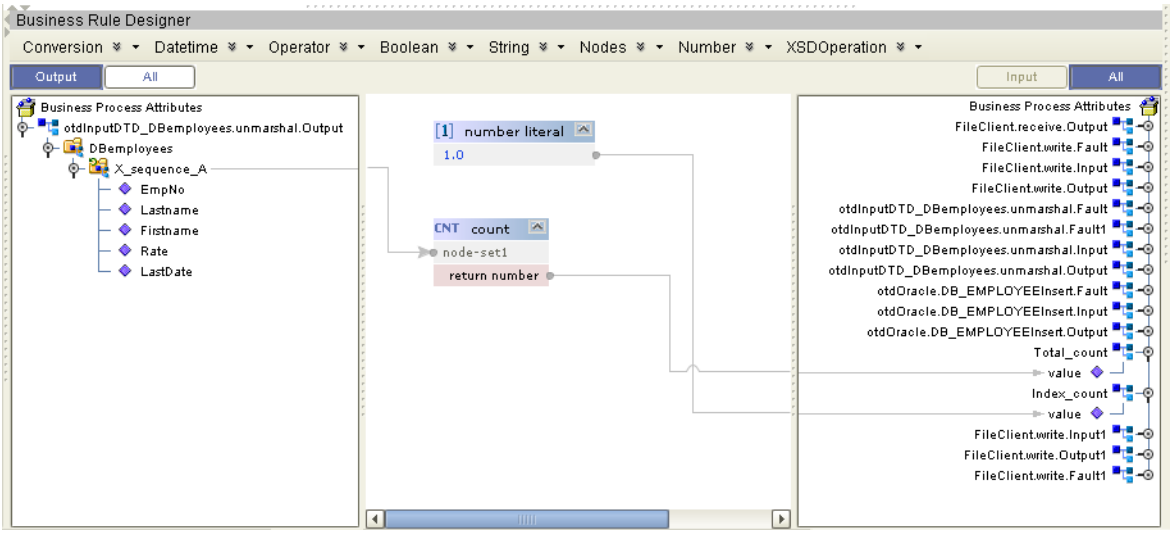
2 Configure the business rule between the **FileClient.write** Activity and **otdInputDTD_DBemployees.unmarshal** Activity as seen in Figure 40.

Figure 40 bpInsert Business Rule # 2



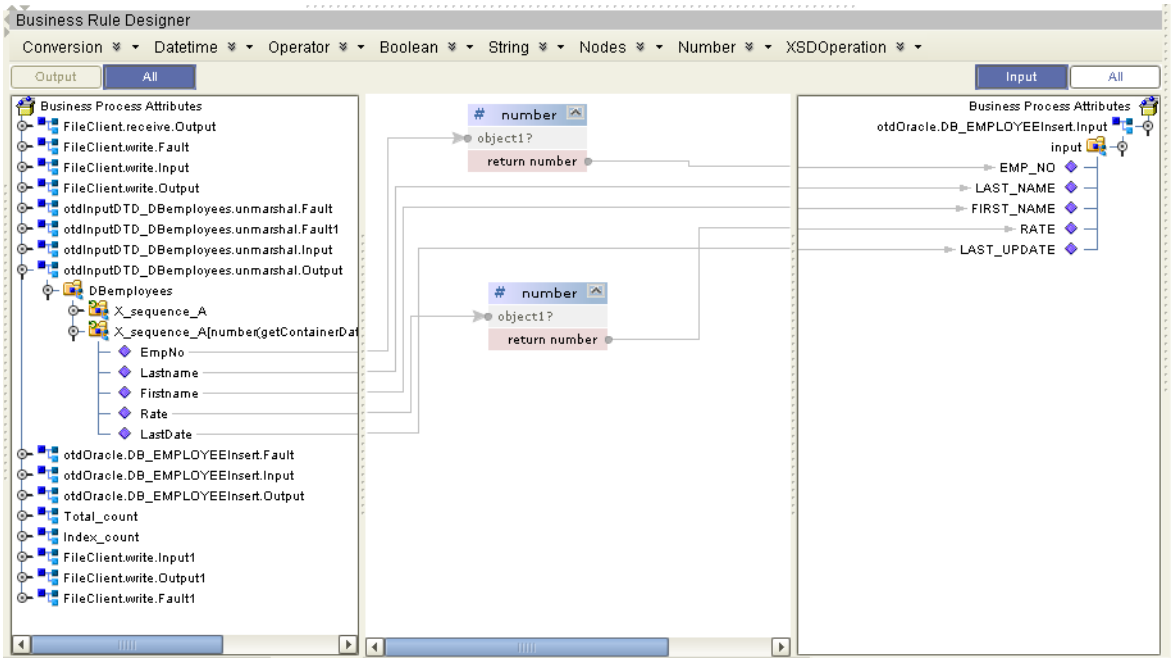
3 Configure the business rule between **otdInputDTD_DBemployees.unmarshal** and the **Insert** (Scope element).

Figure 41 bpInsert Business Rule # 3



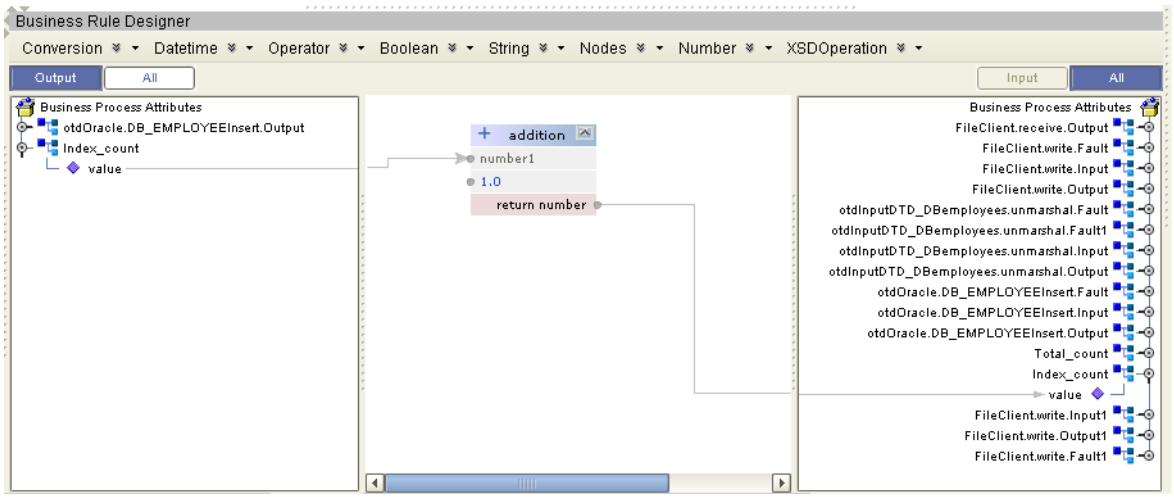
4 Configure the business rule in the **While** statement that connects to the **otdOracle.DB_EMPLOYEEInsert** Activity.

Figure 42 bpInsert Business Rule # 4



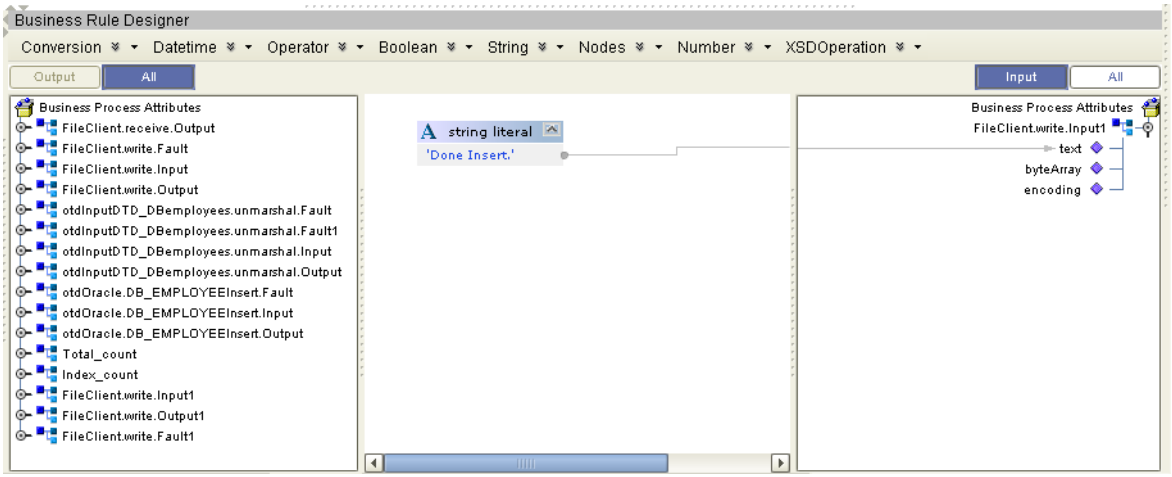
5 Configure the business rule in the **While** statement that connects from the **otdOracle.DB_EMPLOYEEInsert** Activity.

Figure 43 bpInsert Business Rule # 5



6 Configure the business rule from the **Insert** (Scope element) to the **FileClient.write** Activity.

Figure 44 bpInsert Business Rule # 6



Configuring the bpUpdate Modeling Elements

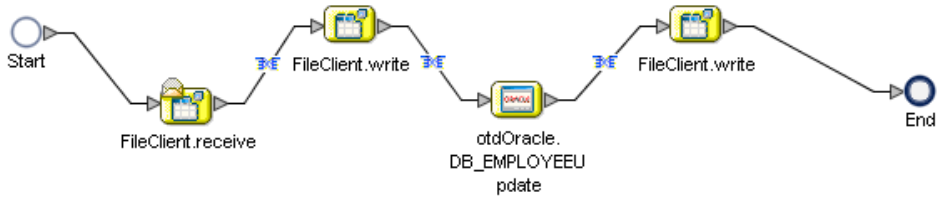
The bpUpdate business process describes how to update a record in the Oracle database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Update operation. Figure 45 illustrates how all the modeling elements appear when connected.

Note: *The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerUpdate.in file is empty.*

Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

Figure 45 bpUpdate Business Process

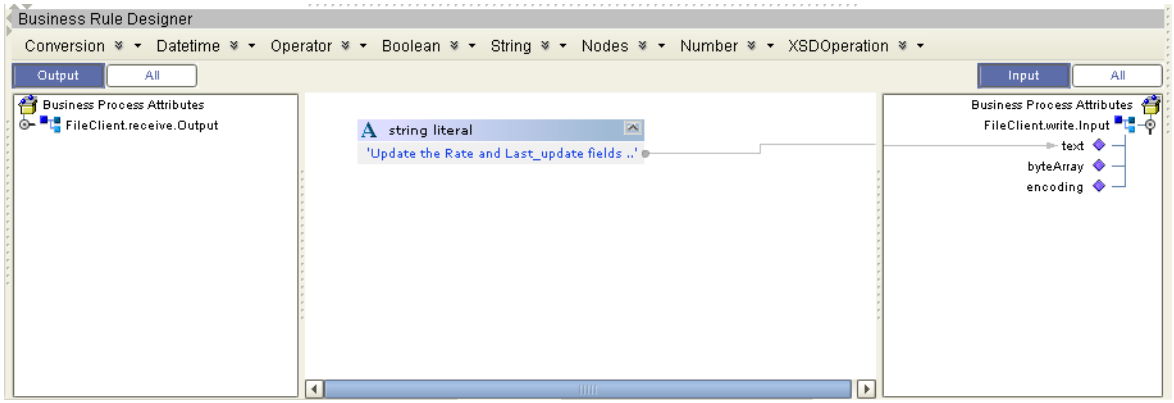


Steps required to configure the bpUpdate business process:

Configure the following business rules in the bpUpdate business process.

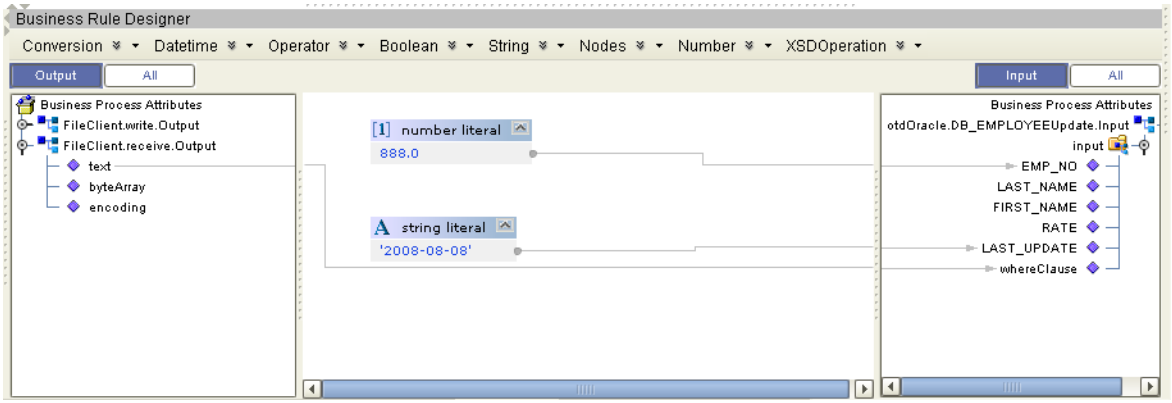
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 46.

Figure 46 bpUpdate Business Rule # 1



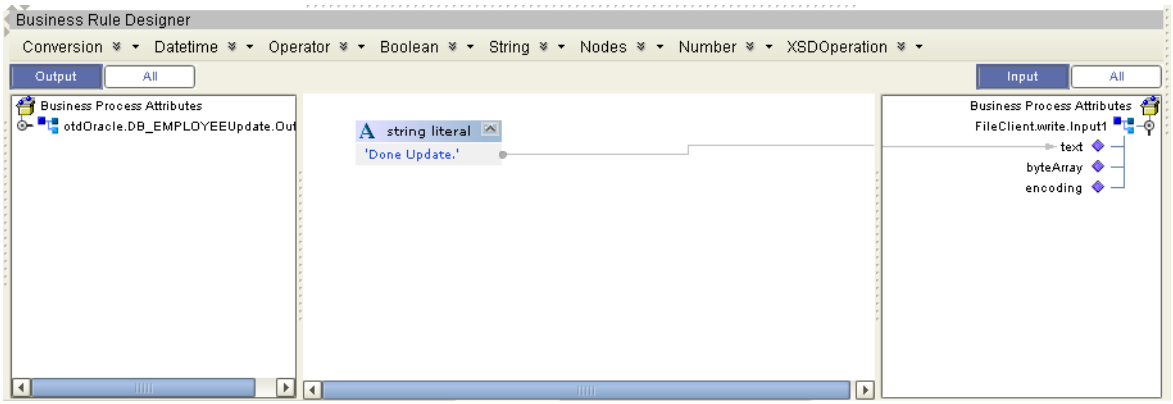
2 Configure the business rule between the **FileClient.write** Activity and **otdOracle.DB_EMPLOYEEUpdate** Activity as seen in Figure 47.

Figure 47 bpUpdate Business Rule # 2



3 Configure the business rule between **otdOracle.DB_EMPLOYEEUpdate** and the **FileClient.write** activity.

Figure 48 bpUpdate Business Rule # 3



4 Configure the business rule in the **While** statement that connects to the **otdOracle.DB_EMPLOYEEInsert** Activity.

5 Configure the business rule from the **Insert** (Scope element) to the **FileClient.write** Activity.

Configuring the bpDelete Modeling Elements

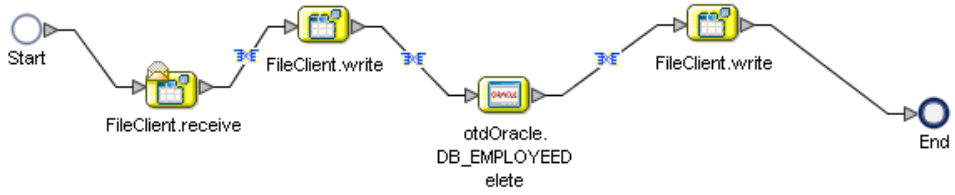
The bpDelete business process describes how to delete a record in the Oracle database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Delete operation. See Figure 49 for an illustration of how all the modeling elements appear when connected.

Note: *The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerDelete.in file is empty.*

Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

Figure 49 bpDelete Business Process

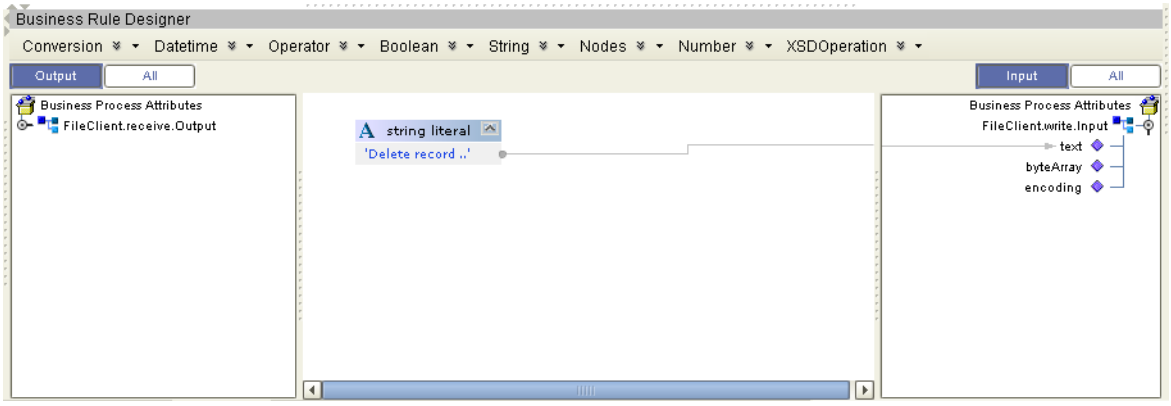


Steps required to configure the bpDelete business process:

Configure the following business rules to the bpDelete business process.

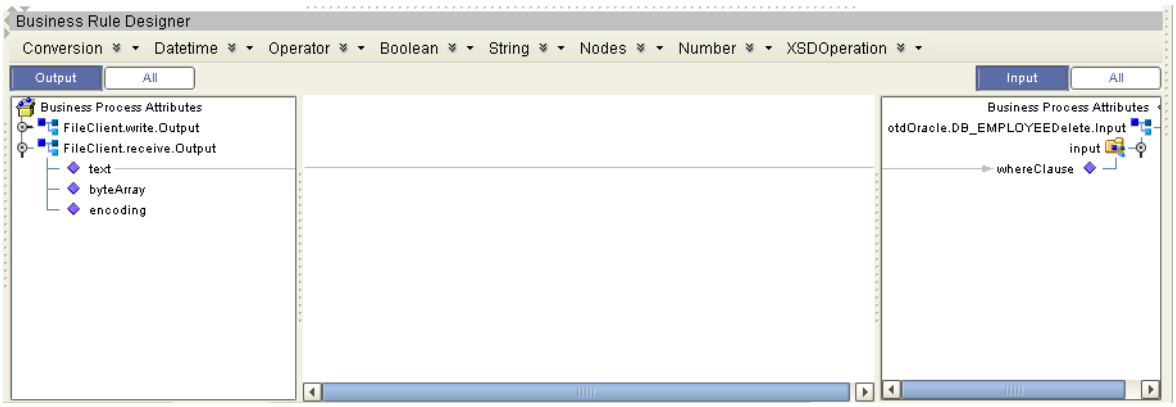
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 50.

Figure 50 bpDelete Business Rule # 1



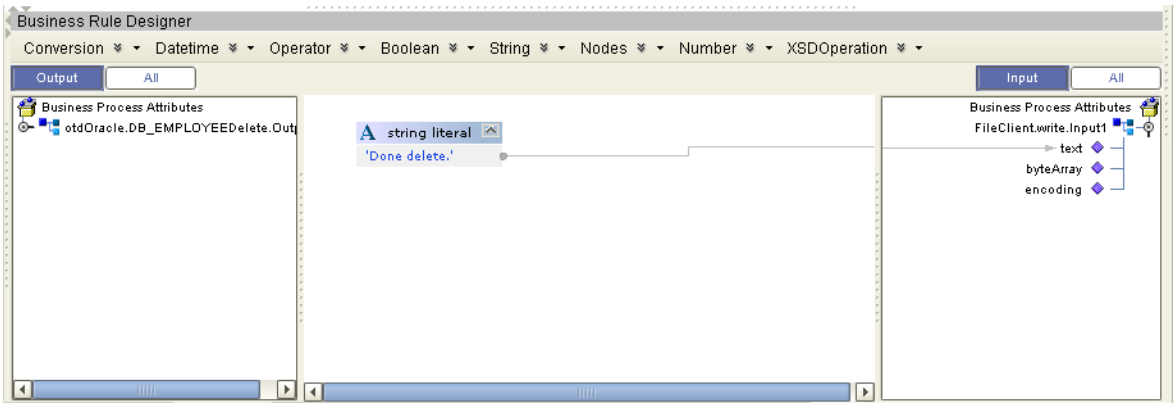
2 Configure the business rule between the **FileClient.write** Activity and **otdOracle.DB_EMPLOYEEDelete** Activity as seen in Figure 51.

Figure 51 bpDelete Business Rule # 2



3 Configure the business rule between the **otdOracle.DB_EMPLOYEEDelete** Activity and the **FileClient.write** Activity as seen in Figure 52.

Figure 52 bpDelete Business Rule # 3



Configuring the bpTableSelect Modeling Elements

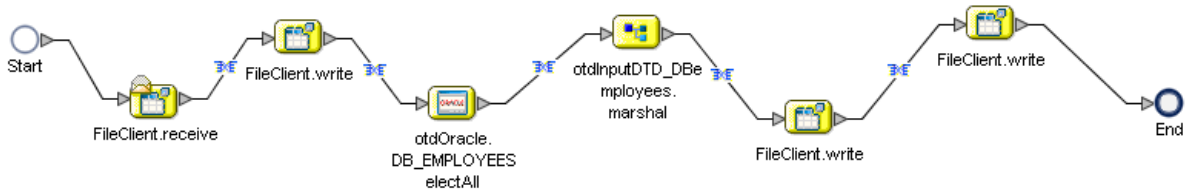
The bpTableSelect business process is describes how to select all records the Oracle database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 53 for an illustration of how all the modeling elements appear when connected.

Note: *The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.*

Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

Figure 53 bpTableSelect Business Process

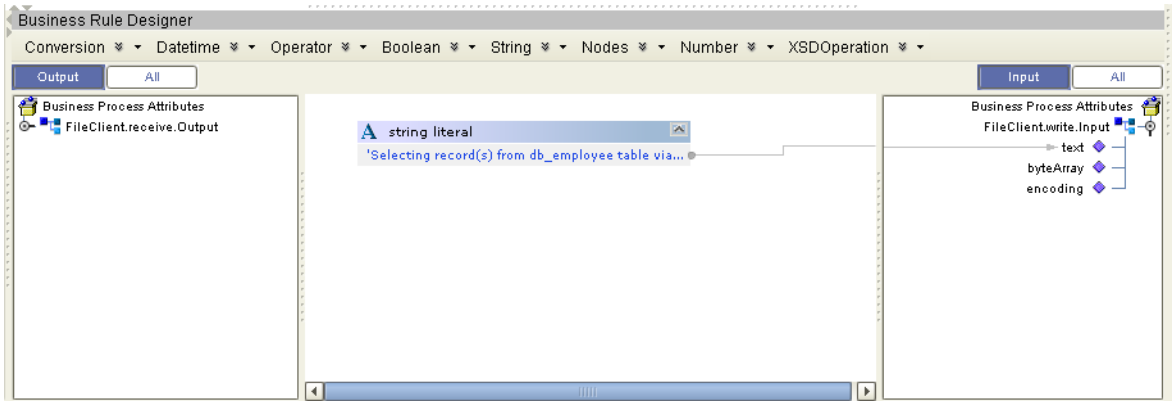


Steps required to configure the bpTableSelect business process:

Configure the following business rules in the bpTableSelect business process.

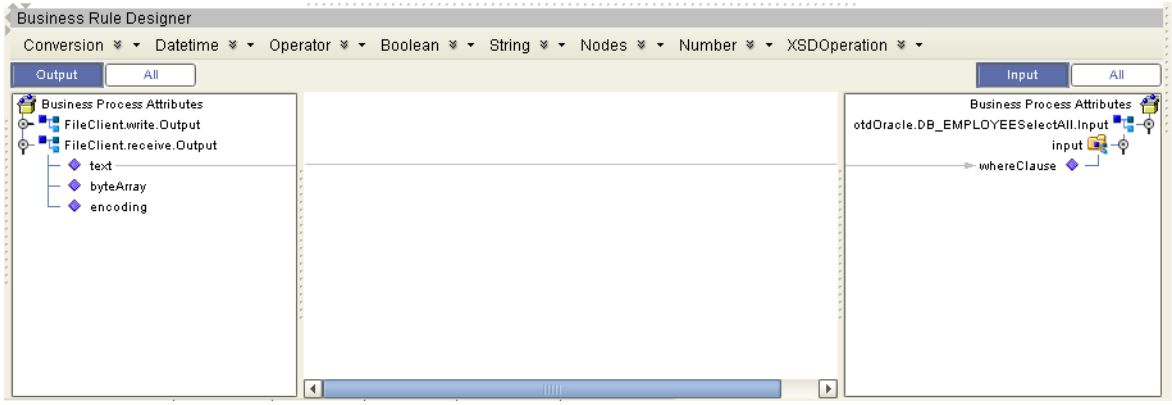
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 54.

Figure 54 bpTableSelect Business Rule # 1



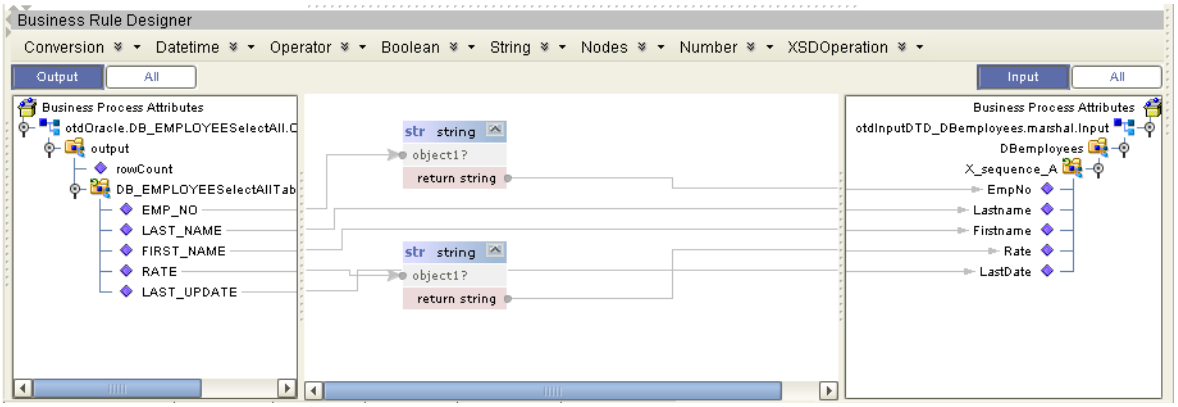
2 Configure the business rule between the **FileClient.write** Activity and **otdOracle.DB_EMPLOYEESelectAll** Activity as seen in Figure 55.

Figure 55 bpTableSelect Business Rule # 2



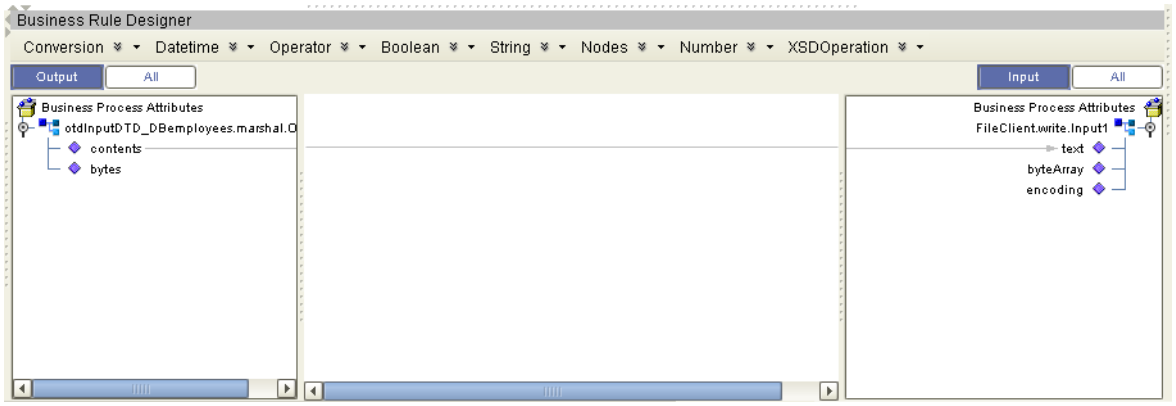
3 Configure the business rule between the **otdOracle.DB_EMPLOYEESelectAll** Activity and the **otdInputDTD_DBemployees.marshal** Activity as seen in Figure 56.

Figure 56 bpSelectTable Business Rule # 3



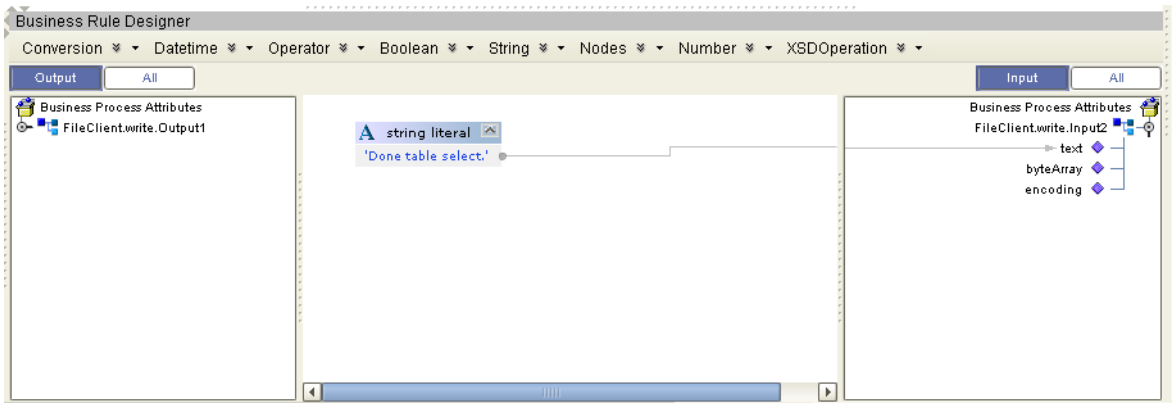
4 Configure the business rule between the **otdInputDTD_DBemployees.marshal** Activity and the **FileClient.write** Activity as seen in Figure 57.

Figure 57 bpTableSelect Business Rule # 3



5 Configure the business rule between the **FileClient.write** Activity and the **FileClient.write** Activity as seen in Figure 58.

Figure 58 bpTableSelect Business Rule # 3



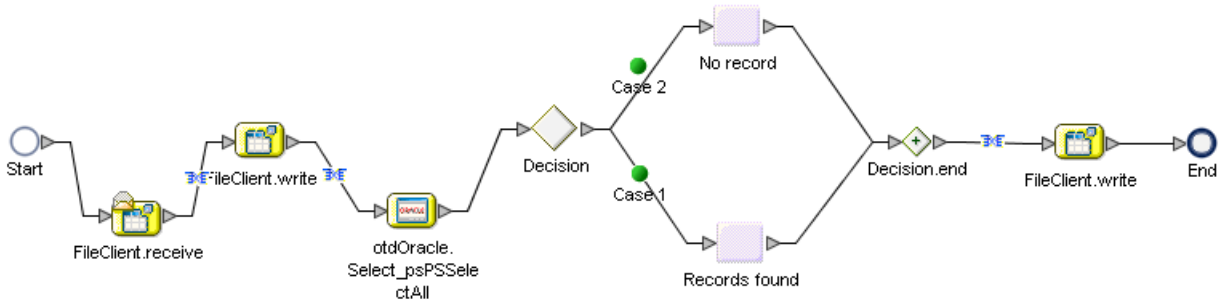
Configuring the bpPsSelect Modeling Elements

The bpPsSelect business process describes how to use a Prepared Statement query to select all records in the Oracle database via the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 59 for an illustration of how all the modeling elements appear when connected.

Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

Figure 59 bpPsSelect Business Process

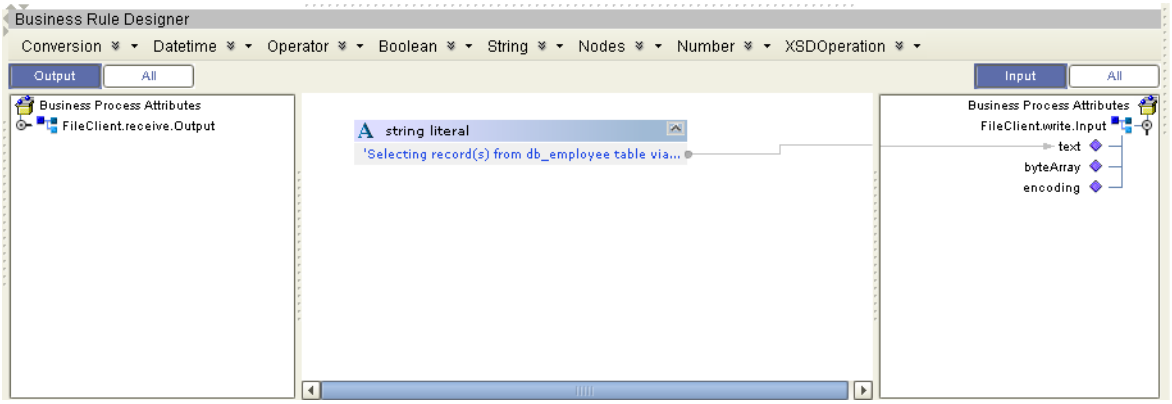


Steps required to configure the bpPsSelect business process:

Configure the following business rules in the bpPsSelect business process.

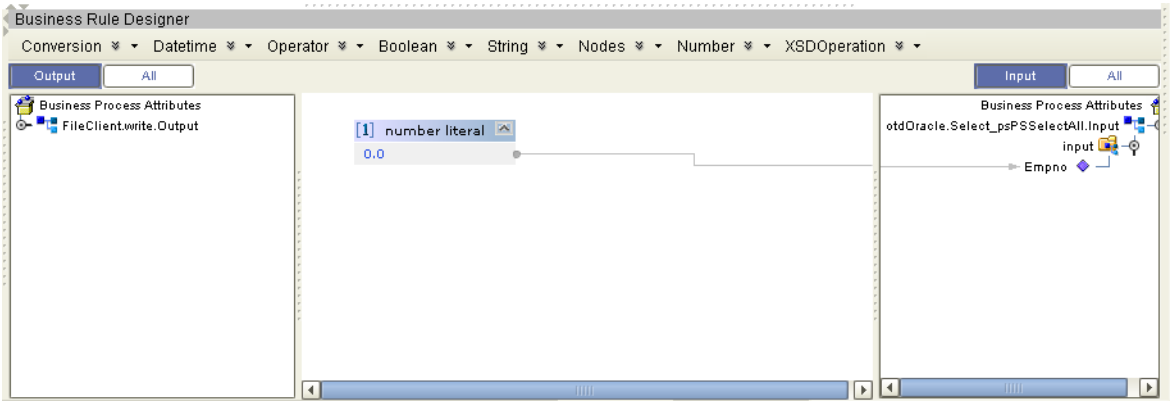
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 54.

Figure 60 bpSelectTable Business Rule # 1



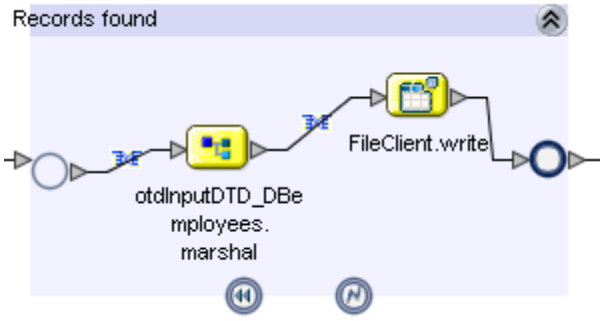
2 Configure the business rule between **FileClient.write** and **otdOracle.Select_psPSSelectAll** as seen in Figure 61.

Figure 61 bpSelectTable Business Rule # 2



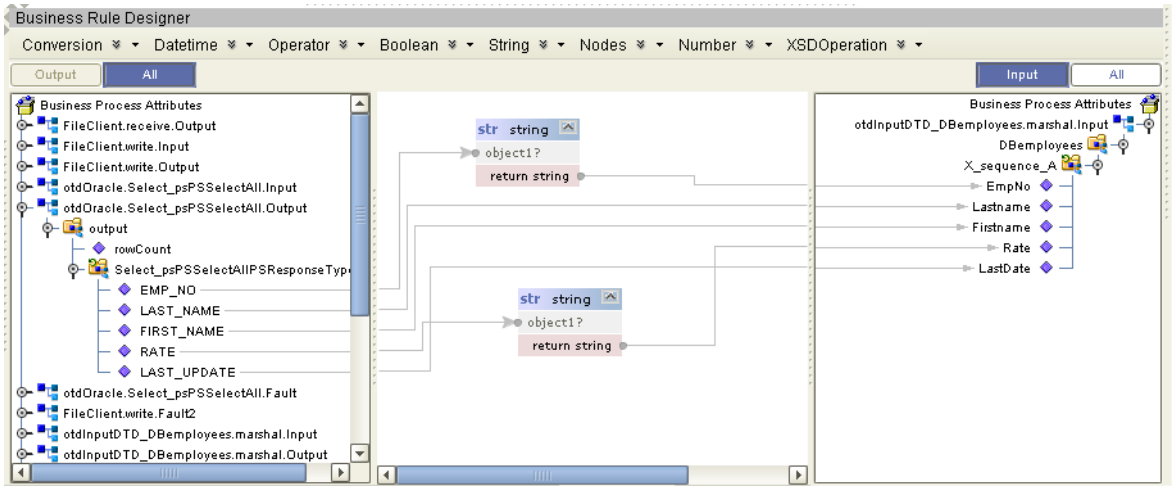
3 Configure **Case 1** of the Decision branching activity. This requires adding business rules between the **otdInputDTD_DBemployees.marshal** and the **FileClient.write** activities within the Scope element.

Figure 62 Activities within Case 1 Scope



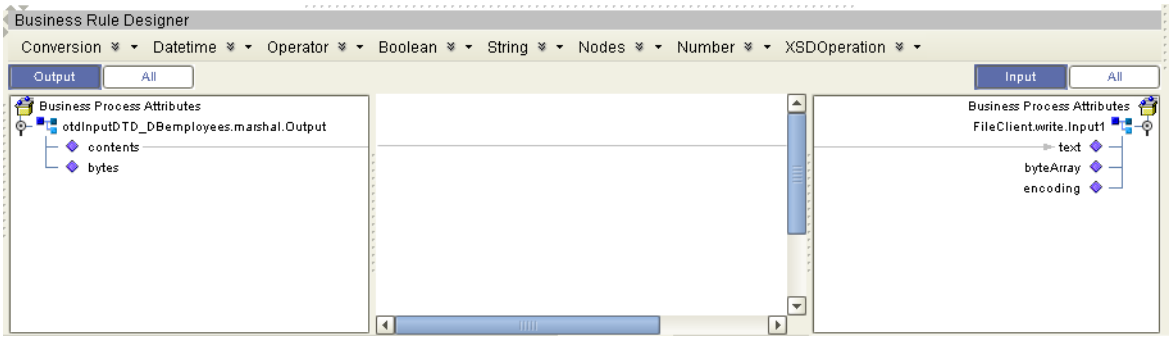
4 Configure the business rule between the start of the Scope element in **Case 1** and the **otdInputDTD_DBemployees.marshal** activity, as seen in Figure 63.

Figure 63 Case 1 Scope Business Rule # 3



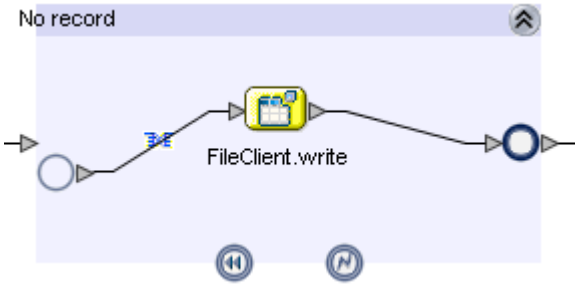
5 Configure the business rule between **otdInputDTD_DBemployees.marshal** and **FileCleint.write** in the Scope element, as seen in Figure 64.

Figure 64 Case 1 Scope Business Rule # 4



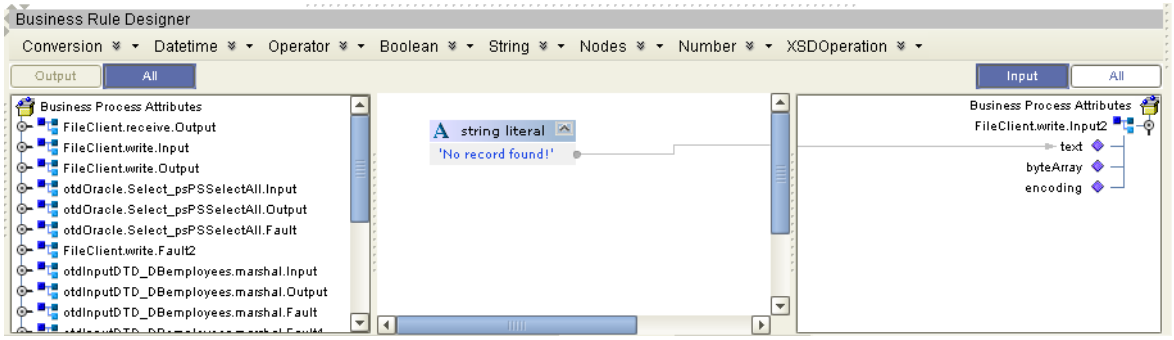
6 Configure Case 2 of the Decision branching activity. This requires adding business rules between the **otdInputDTD_DBemployees.marshal** and the **FileClient.write** activities within the Scope element.

Figure 65 Activities within Case 2 Scope



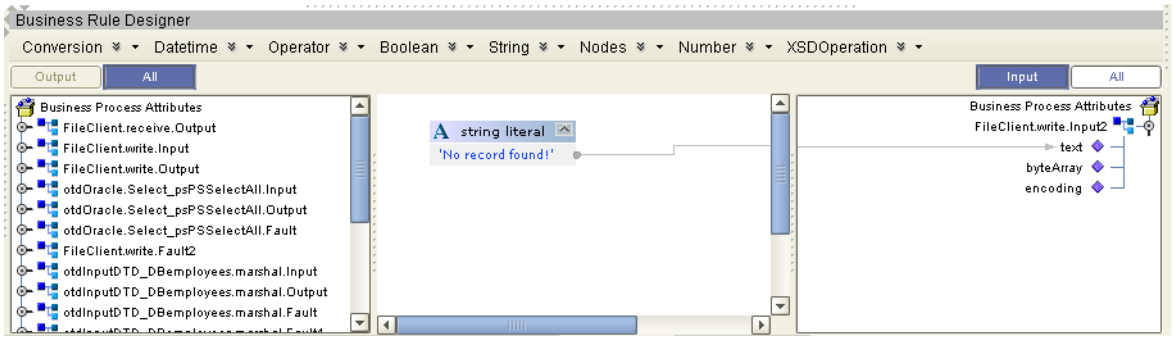
7 Configure the business rule between the start of the Scope element in **Case 2** and the **FileClient.Write** activity, as seen in Figure 66.

Figure 66 Case 2 Scope Business Rule # 5



8 Configure the business rule between the **Decision.end** Element and the **FileClient.write** Activity, as seen in Figure 67.

Figure 67 bpSelectTable Business Rule # 6



6.5.4 Creating the Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

1 From the Project Explorer tree, right-click the new **prjOracle_BPEL** Project and select **New > Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears, and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**.

Create four additional Connectivity Maps—**CMap2**, **CMap3**. **CMap4**, and **CMap5**—and rename them as follows:

- cmDelete
- cmInsert
- cmPsSelect
- cmTableSelect
- cmUpdate

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the **prjOracle_BPEL** sample Project requires the following components:

- File External Application (2)
- Oracle External Application
- Business Process

Any eWay added to the Connectivity Map is associated with an External System. To establish a connection to Oracle, first select Oracle as an External System to use in your Connectivity Map.

To Select a Oracle External System

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the external systems necessary to create your Project (for this sample, Oracle and **File**). Icons representing the selected external systems are added to the Connectivity Map toolbar.
3. Rename the following components and then save changes to the Repository:
   - File1 to FileClientIN
   - File2 to FileClientOUT
   - Oracle1 to eaOracleOUT

To Select a Oracle Business Process

1. Drag a business process from the Enterprise Explorer Project Explorer onto the corresponding Connectivity Map. For example, drag the **dbDelete** business process onto the **cmDelete** Connectivity Map.
2. Save your changes to the Repository

Binding the eWay Components

The final step in creating a Connectivity Map is binding the eWay components together.

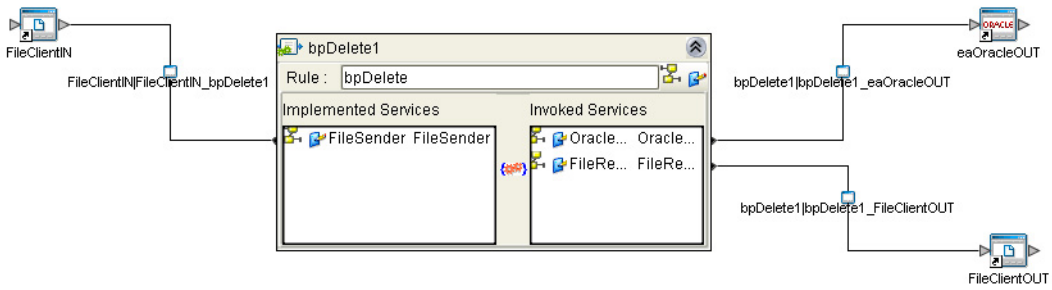
Steps required to bind eWay components together

1. Open one of the Connectivity Maps and double-click a Business Process, for example the **bpDelete** Business Process in the **cmDelete** Connectivity Map. The **bpDelete** Binding dialog box appears.
2. From the **bpDelete** Binding dialog box, map **FileSender** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **bpDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **bpDelete**.

3 From the **bpDelete** Binding dialog box, map **Oracle_otdOracle** (under Invoked Services) to the **eaOracleOUT** External Application.

4 From the **bpDelete** Binding dialog box, map **FileReceiver** to the **FileClientOUT** External Application, as seen in Figure 68.

Figure 68 Connectivity Map - Associating (Binding) the Project's Components



5 Minimize the **bpDelete** Binding dialog box by clicking the chevrons in the upper-right corner.

6 Save your current changes to the Repository, and then repeat this process for each of the other Connectivity Maps.

6.5.5 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Editor.

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **envOracleProj**.

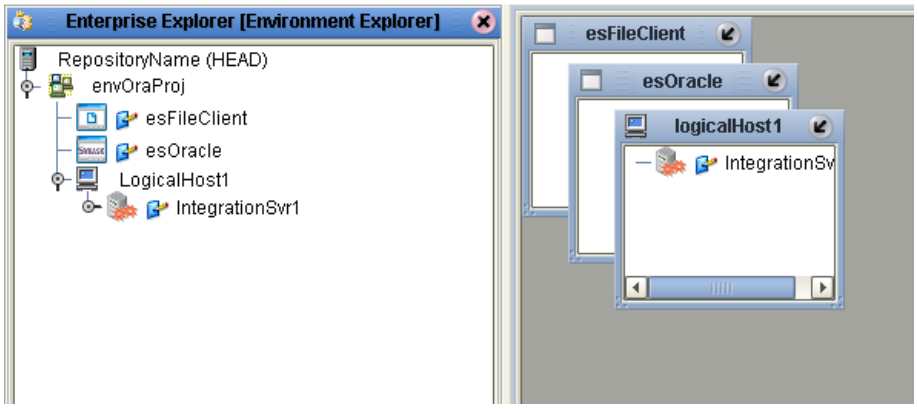
4 Right-click **envOracleProj** and select **New Oracle External System**. Name the External System **esOracle**. Click **OK**. **esOracle** is added to the Environment Editor.

5 Right-click **envOracleProj** and select **New File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.

6 Right-click **envOracleProj** and select **New Logical Host**. The **LogicalHost1** box is added to the Environment, and **LogicalHost1** is added to the Environment Editor tree.

7 Right-click **LogicalHost1** and select **New Sun SeeBeyond Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see Figure 69).

Figure 69 Environment Editor - envOracleProj



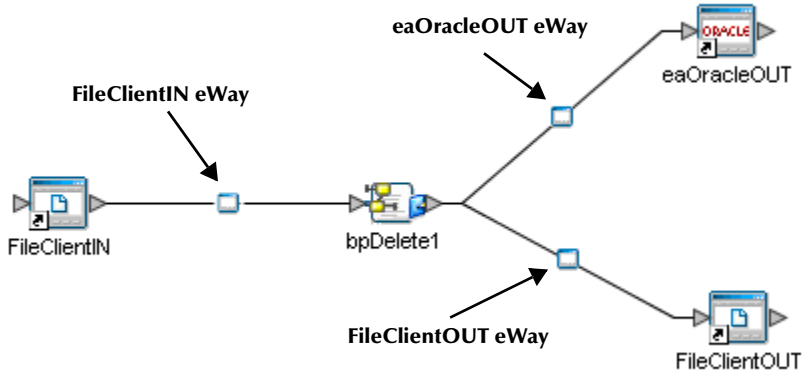
8 Save your current changes to the Repository.

6.5.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the The **prjOracle_BPEL** sample Project use three eWays that are represented as a nodes between the External Applications and the Business Process, as seen in Figure 70.

You must configure eWay properties in both the Connectivity Map and the Environment Explorer.

Figure 70 eWays in the cmDelete Connectivity Map



Steps to Configure the eWay Properties

1 Double-click the **FileClientIN eWay on each of the Connectivity Maps and** modify the properties for your system, as seen in Table 13. Click **OK** to close the Properties Editor.

Table 13 FileClientIN eWay Property Settings

Connectivity Map	Property Name	Required Value
cmDelete	Input file name	TriggerDelete.in
cmInsert	Input file name	TriggerBpInsert.in
cmPsSelect	Input file name	TriggerPsSelect.in
cmTableSelect	Input file name	TriggerTableSelect.in
cmUpdate	Input file name	TriggerUpdate.in

2 Double-click the **FileClientIN eWay on each of the Connectivity Maps and** modify the properties for your system, as seen in Table 14. Click **OK** to close the Properties Editor.

Table 14 FileClientOUT eWay Property Settings

Connectivity Map	Property Name	Required Value
cmDelete	Input file name	BPEL_Delete_output%d.dat
cmInsert	Input file name	BPEL_Insert_output%d.dat
cmPsSelect	Input file name	BPEL_PsSelect_output%d.dat
cmTableSelect	Input file name	BPEL_TableSelect_output%d.dat.in
cmUpdate	Input file name	BPEL_Update_output%d.dat

Steps to Configure the Environment Explorer Properties

1 From the **Environment Explorer** tree, right-click the File External System (**esOracle** in this sample), and select **Properties**. The Properties Editor opens to the Oracle eWay Environment configuration.

2 Modify the Oracle eWay Environment configuration properties for your system, as seen in Table 15, and click **OK**.

Table 15 Oracle eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound Oracle eWay > JDBC Connector settings	ServerName	Enter the name of the database server being used.
	DatabaseName	Enter the name of the Oracle SID.
	User	Enter the user account name for the database.
	Password	Enter the user account password for the database.

3 From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the Oracle eWay Environment configuration.

4 Modify the File eWay Environment configuration properties for your system, as seen in Table 16, and click **OK**.

Table 16 File eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound File eWay > Parameter Settings	Directory	Enter the directory that contains the input files (trigger files included in the sample Project). Trigger files include: ▪ TriggerBpInsert.in.~in ▪ TriggerDelete.in.~in ▪ TriggerPsSelect.in.~in ▪ TriggerTableSelect.in.~in ▪ TriggerUpdate.in.~in
Configuration > Outbound File eWay > Parameter Settings	Directory	Enter the directory where output files are written. In this sample Project, the output files include: ▪ BPEL_Delete_output0.dat ▪ BPEL_Insert_output0.dat ▪ BPEL_PsSelect_output0.dat ▪ BPEL_TableSelect_output0.dat ▪ BPEL_Update_output0.dat

6.5.7 Configuring the Integration Server

You must set your Sun SeeBeyond Integration Server Password property before deploying your Project.

1 From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.

2 Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.

3 Click the ellipsis. The **Password Settings** dialog box appears. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.

4 Click **OK** to accept the new property and close the Properties Editor.

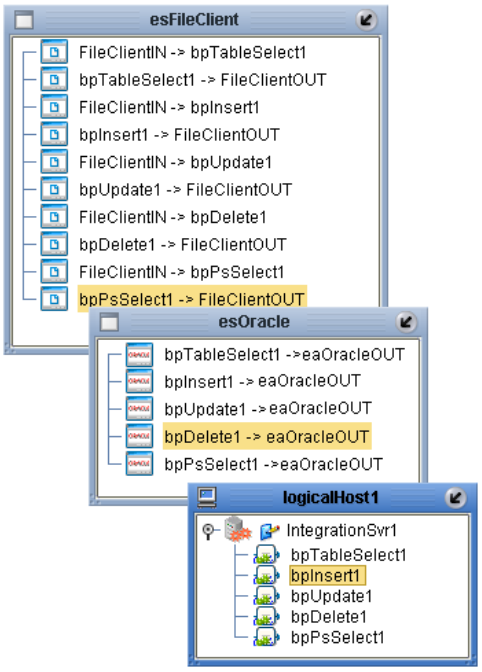
For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

6.5.8 Creating the Deployment Profile

A Deployment Profile is used to assign services and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1. From the Enterprise Explorer's Project Explorer, right-click the **prjOracle_BPEL** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpOracle_BPEL**). Select **envOracleProj** as the Environment and click **OK**.
3. From the Deployment Editor toolbar, click the **Automap** icon. The Project's components are automatically mapped to their system windows, as seen in Figure 71.

Figure 71 Deployment Profile



6.5.9 Creating and Starting the Domain

To build and deploy your Project, you must first create a domain. A domain is an instance of a Logical Host. After the domain is created, the Project is built and then deployed.

Note: *You are only required to create a domain once when you install the Java Composite Application Platform Suite.*

Steps to create and start the domain include:

1. Navigate to your **<caps51>\logicalhost** directory (where <caps51> is the location of your Sun Java Composite Application Platform Suite installation.

2 Double-click the **domainmgr.bat** file. The **Domain Manager** appears.

3 If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

4 If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.

5 Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

Note: *For more information about creating and managing domains see the eGate Integrator System Administration Guide.*

6.5.10 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

Build the Project

1 From the Deployment Editor toolbar, click the **Build** icon.

2 If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.

3 After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1 From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.

2 A message appears when the project is successfully deployed. You can now test your sample.

Deploying a Project to an HP NonStop Server

To deploy a project on an HP NonStop Server, please see the "eGate Integrator User's Guide".

6.5.11 Running the Sample Project

Perform the following steps to run your deployed sample Project:

1 Rename one of the trigger files included in the sample Project from **<filename>.in.~in** to **<filename>.in** to run the corresponding operation.

The File eWay polls the directory every five seconds for the input file name (as defined in the Inbound File eWay Properties window). The Business Process then

transforms the data, and the File eWay sends the output to an Output file name (as defined in the outbound File eWay Properties window).

The Where Clause defined in the business rule recognizes the trigger as a placeholder for input, allowing a set condition, such as emp_no = 100, to determine the type of output data.

You can modify the following input files to view different output.

- TriggerTableSelect.in
- TriggerDelete.in
- TriggerUpdate.in

Having no content in these files causes the operation to read all records.

2 Verify the output data by viewing the sample output files. See **About the Oracle eWay Sample Projects** on page 66 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.

6.6 Creating the prjOracle_JCD Sample Project

The following provides step-by-step instructions for manually creating the prjOracle_JCD sample Project.

Steps required to create the sample project include:

6.6.1 Creating a Project

The first step is to create a new Project in the Sun SeeBeyond Enterprise Designer.

1 Start the Enterprise Designer.

2 From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.

3 Click twice on **Project1** and rename the Project (for this sample, **prjOracle_BPEL**).

6.6.2 Creating the OTDs

The sample Project requires three OTDs to interact with the Oracle eWay. These OTDs include:

- Oracle Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a Oracle Database OTD include:

1 Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Object Type Definition**.

The New Object Type Definition Wizard window appears.

2 Select the **Oracle Database OTD Wizard** from the list of OTD Wizards and click **Next**.

3 Enter the connection information for the Oracle database. Connection fields include:

- Host name:
- Port ID:
- SID:
- User name:
- Password:

4 Click **Next**, and select the types of database object you want to include in the sample Project. For our example, select the following:

- Tables/Views/Aliases
- Prepared Statements

5 Click **Add** to select tables from the Oracle database. The **Add Tables** window appears.

6 Search for or Type in the name of the database. In this example we use the **DB_EMPLOYEE** table. Click **Select** when the database appears in the Results selection frame. Click **OK** to close the Add Tables window

7 Click **Next the Add Prepared Statements Wizard appears.**

8 Click **Add**, the Add Prepared Statement window appears. Enter the following:

- Prepared Statement Name: Select_ps
- SQL Statement:

```
select * from db_employee where emp_no > ? order by emp_no
```

Note: *In this example, the SQL statement includes the* **?** *placeholder for input. This placeholder represents the Where Clause.*

9 Click the **OK** button to close the Prepared Statement window, and then click **Next** on the Prepared Statements Wizard window.

10 Enter an OTD name. In this example, use **otdOracle**.

11 Click **Next** and review your settings, then click **Finish** to create the OTD.

Steps required to create inbound and outbound DTD OTDs include:

1 Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Object Type Definition**.

The New Object Type Definition Wizard window appears.

2 Select **DTD** from the list of OTD Wizards and click **Next**.

3 Browse to and then select a DTD file. For this example, select one of the following DTD files from the sample Project, and then click **Next**.

- otdInputDTD.dtd
- otdOutputDTD.dtd

4 The file you select appears in the Select Document Elements window. Click **Next**.

5 Click **Finish** to complete the DTD based OTD. Repeat this process again to create the second DTD file.

6.6.3 Creating a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a Project's components.

1 From the Project Explorer tree, right-click the new **prjOracle_JCD** Project and select **New > Connectivity Map** from the shortcut menu.

2 The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree labeled **CMap1**.

Create four additional Connectivity Maps—**CMap2**, **CMap3**. **CMap4**, and **CMap5**— and rename them as follows:

- cmDelete
- cmInsert
- cmPsSelect
- cmTableSelect
- cmUpdate

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the **prjOracle_JCD** sample Project requires the following components:

- File External Application (2)
- Oracle External Application
- Service

Any eWay added to the Connectivity Map is associated with an External System. To establish a connection to Oracle, first select Oracle as an External System to use in your Connectivity Map.

Steps required to select an Oracle External System

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the external systems necessary to create your Project (for this sample, **Oracle** and **File**). Icons representing the selected external systems are added to the Connectivity Map toolbar.
3. Rename the following components and then save changes to the Repository:
   - File1 to FileClientIN
   - File2 to FileClientOUT
   - Oracle1 to eaOracleOUT
4. Rename each Connectivity Map Service to match the intended operation, as for example:
   - jcdDelete
   - jcdInsert
   - jcdPsSelect
   - jcdTableSelect
   - jcdUpdate

6.6.4 Creating the Collaboration Definitions (Java)

The next step is to create Collaborations using the **Collaboration Definition Wizard (Java)**. Since the sample Project includes five database operations, you must create five separate Collaboration Definitions (Java), or JCDs. Once you create the Collaboration Definitions, you can write the Business Rules of the Collaborations using the Collaboration Editor.

JCDs required for the **prjOracle_JCD** sample include:

- jcdDelete
- jcdInsert

- jcdPsSelect
- jcdTableSelect
- jcdUpdate

jcdDelete Collaboration

The Steps to Create the jcdDelete Collaboration include:

1. From the Project Explorer, right-click the sample Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
2. Enter a Collaboration Definition name (for this sample **jcdDelete**) and click **Next**.
3. For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.
4. For Step 3 of the wizard, from the Select OTDs selection window, double-click **prjOracle_JCD** > **otdALL** > **otdOracle**. The **otdOracle** OTD is added to the Selected OTDs field.
5. Click the **Up One Level** button to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.
6. Click **Finish**. The Collaboration Editor with the new **jcdDelete** Collaboration appears in the right pane of the Enterprise Designer.

jcdInsert Collaboration

Steps required to create the jcdInsert Collaboration:

1. From the Project Explorer, right-click the sample Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
2. Enter a Collaboration Definition name (for this sample **jcdInsert**) and click **Next**.
3. For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.
4. For Step 3 of the wizard, from the Select OTDs selection window, double-click **prjOracle_JCD** > **otdALL** > **otdOracle**. The **otdOracle** OTD is added to the Selected OTDs field.
5. In the same window, double-click **otdInputDTD_DBemployees**. The **otdInputDTD_DBemployees** OTD is added to the Selected OTDs field.

Note: *The otdOutputDTD_DBemployees OTD is created from the otdInputDTD.dtd that is included in the Sample Project.*

6 Click the **Up One Level** button to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.

7 Click **Finish**. The Collaboration Editor with the new **jcdInsert** Collaboration appears in the right pane of the Enterprise Designer.

jcdPsSelect Collaboration

Steps required to create the jcdPsSelect Collaboration:

1 From the Project Explorer, right-click the sample Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

2 Enter a Collaboration Definition name (for this sample **jcdPsSelect**) and click **Next**.

3 For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.

4 For Step 3 of the wizard, from the Select OTDs selection window, double-click **prjOracle_JCD** > **otdALL** > **otdOracle**. The **otdOracle** OTD is added to the Selected OTDs field.

5 In the same window, double-click **otdOutputDTD_DBemployee**. The **otdOutputDTD_DBemployee** OTD is added to the Selected OTDs field.

Note that the otdOutputDTD_DBemployee OTD is created from the otdOutputDTD.dtd that is included in the Sample Project.

6 Click the **Up One Level** button to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.

7 Click **Finish**. The Collaboration Editor with the new **jcdPsSelect** Collaboration appears in the right pane of the Enterprise Designer.

jcdTableSelect Collaboration

Steps required to create the jcdTableSelect Collaboration:

1 From the Project Explorer, right-click the sample Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

2 Enter a Collaboration Definition name (for this sample **jcdTableSelect**) and click **Next**.

3 For Step 2 or the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.

4 For Step 3 of the wizard, from the Select OTDs selection window, double-click **prjOracle_JCD** > **otdALL** > **otdOracle**. The **otdOracle** OTD is added to the Selected OTDs field.

5 In the same window, double-click **otdOutputDTD_DBemployee**. The **otdOutputDTD_DBemployee** OTD is added to the Selected OTDs field.

Note: *The otdOutputDTD_DBemployee OTD is created from the otdOutputDTD.dtd that is included in the Sample Project.*

6 Click the **Up One Level** button to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.

7 Click **Finish**. The Collaboration Editor with the new **jcdTableSelect** Collaboration appears in the right pane of the Enterprise Designer.

jcdUpdate Collaboration

Steps required to create the jcdUpdate Collaboration:

1 From the Project Explorer, right-click the sample Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

2 Enter a Collaboration Definition name (for this sample **jcdUpdate**) and click **Next**.

3 For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient** > **receive**. The File Name field now displays **receive.** Click **Next**.

4 For Step 3 of the wizard, from the Select OTDs selection window, double-click **prjOracle_JCD > otdALL > otdOracle**. The **otdOracle** OTD is added to the Selected OTDs field.

5 Click the **Up One Level** button to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.

6 Click **Finish**. The Collaboration Editor with the new **jcdUpdate** Collaboration appears in the right pane of the Enterprise Designer.

6.6.5 Create the Collaboration Business Rules

The next step in the sample is to create the Business Rules of the Collaboration using the Collaboration Editor.

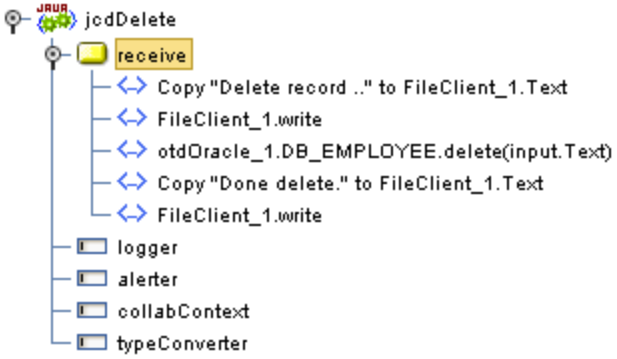
Creating the jcdDelete Business Rules

The **jcdDelete** Collaboration implements the Input Web Service Operation to read the **TriggerDelete.in** file and then delete the record **emp_no = 500**. The Collaboration also writes a message to **JCD_Delete_output0.dat** to confirm a deleted record.

Note: *The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerDelete.in file is empty.*

The **jcdDelete** Collaboration contains the Business Rules displayed in Figure 72.

Figure 72 jcdDelete Business Rules



Creating the jcdInsert Business Rules

The **jcdInsert** Collaboration implements the Input Web Service Operation to read the **TriggerInsert.in**. file. It then unmarshals data from the input data into the **otdInputDTD_DBEmployees** OTD, calls the **otdOracle** OTD, and inserts records into the database via a For Loop. The Collaboration also writes a message to **JCD_Insert_output0.dat** to confirm an inserted record.

The **jcdInsert** Collaboration contains the Business Rules displayed in Figure 73.

Figure 73 jcdInsert Business Rules

Sample code from the jcdInsert Includes:

```
package prjOracle_JCDjcdALL;
```

```
public class jcdInsert
{
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
otdOracle.OtdOracleOTD otdOracle_1,
dtd.otdInputDTD_1206505729.DB_Employee otdInputDTD_DB_Employee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

\\ Writes out a message stating records are being inserted.

       FileClient_1.setText( "Inserting records in to db_employee
table......" );
        FileClient_1.write();

\\ Unmarshals data from the input XML data into the
otdInputDTD_DBEmployees OTD.

        otdInputDTD_DB_Employee_1.unmarshalFromString(
input.getText() );

\\ Calls the otdOracle OTD, and inserts multiple records into the
database via a For Loop. The first insert() method opens the table
result set for insert operations, while the insertRow() method
inserts records into the table result set.

        otdOracle_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdOracle_1.getDb_employee().setEMP_NO(
typeConverter.stringToShort(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdOracle_1.getDb_employee().setLAST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdOracle_1.getDb_employee().setFIRST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdOracle_1.getDb_employee().setRATE( new
java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1
).getRate() ) );
            otdOracle_1.getDb_employee().setLAST_UPDATE(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdOracle_1.getDb_employee().insertRow();

\\ Writes a message to confirm an inserted records.

        }
        FileClient_1.setText( "Insert Done." );
        FileClient_1.write();
    }
}
```

Creating the jcdPsSelect Business Rules

The **jcdPsSelect** Collaboration implements the Input Web Service Operation to read the **TriggerPsSelect.in** file. It then copies the database resultset (as noted in the prepared statement query) into the **otdInputDTD_DBEmployee** OTD and selects all available records from the database. The Collaboration also writes a message to **JCD_PsSelect_output0.dat** to confirm when records are selected, or when no records are available.

The **jcdPsSelect** Collaboration contains the Business Rules displayed in Figure 74.

Figure 74 jcdPsSelect

Sample code from the jcdPsSelect Includes:

```
package prjOracle_JCDjcdALL;

public class jcdPsSelect
{

    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;
    public void receive(

com.stc.connector.appconn.file.FileTextMessage input,
otdOracle.OtdOracleOTD otdOracle_1,
dtd.otdOutputDTD1325973702.DB_Employee otdOutputDTD_DB_Employee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

\\ Writes out a message stating records are being selected

        FileClient_1.setText( "Selecting records from db_employee
table via Prepared Statement select...." );

\\ Copies the database resultset into the otdInputDTD_DBEmployee OTD
and selects all available records from the database. The
executeQuery() method executes the prepared statement query, while
```

```
the resultsAvailable() method ensures all rows are retrieved in the
while loop.

        FileClient_1.write();
        otdOracle_1.getSelect_ps().setEMPNO(
typeConverter.stringToShort( "0", "#", false, 0 ) );
        otdOracle_1.getSelect_ps().executeQuery();
        if (otdOracle_1.getSelect_ps().resultsAvailable()) {
            while
(otdOracle_1.getSelect_ps().get$Select_psResults().next()) {
                otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.intToString(
otdOracle_1.getSelect_ps().get$Select_psResults().getEMP_NO(), "#",
false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastname(
otdOracle_1.getSelect_ps().get$Select_psResults().getLAST_NAME() );
                otdOutputDTD_DB_Employee_1.setFirstname(
otdOracle_1.getSelect_ps().get$Select_psResults().getFIRST_NAME() );
                otdOutputDTD_DB_Employee_1.setRate(
typeConverter.doubleToString(
otdOracle_1.getSelect_ps().get$Select_psResults().getRATE(),
"#.000000;-#.000000", false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdOracle_1.getSelect_ps().get$Select_psResults().getLAST_UPDATE(),
"yyyy-MM-dd hh:mm:ss", false, "" ) );
                FileClient_1.setText(
otdOutputDTD_DB_Employee_1.marshalToString() );
                FileClient_1.write();
            }
        }

\\ Writes a message to JCD_PsSelect_output0.dat to confirm when
records are selected, or when no records are available.

        FileClient_1.setText( "Select Done." );
        FileClient_1.write();
    }
}
```

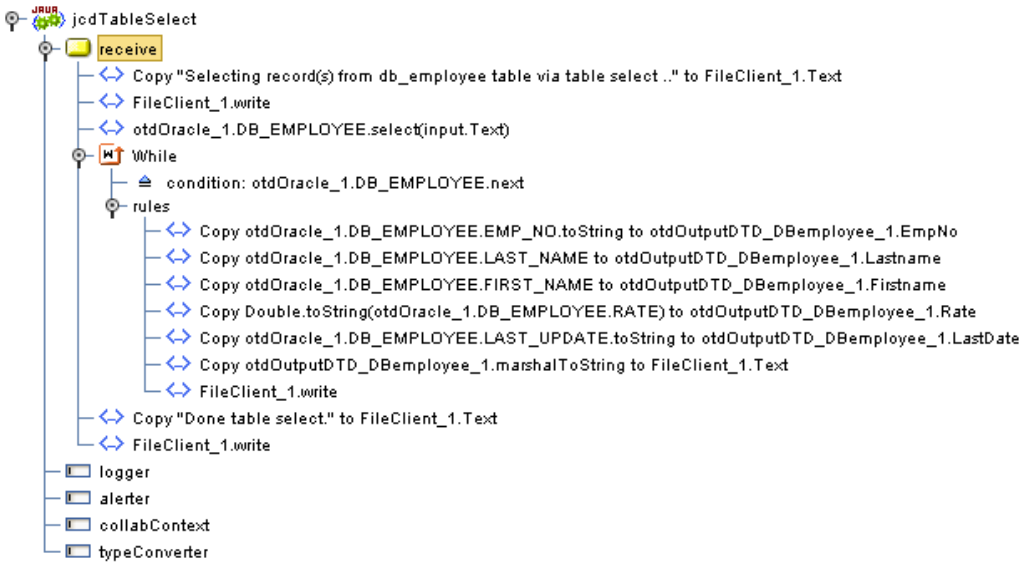
Creating the jcdTableSelect Business Rules

The **jcdTableSelect** Collaboration implements the Input Web Service Operation to read the **TriggerTableSelect.in** file. It then copies the database resultset into the **otdInputDTD_DBEmployee** OTD and selects all available records from the database that meet the criteria **emp_no = 100**. The Collaboration also writes a message to **JCD_TableSelect_output0.dat** to confirm when records are selected, or when no records are available.

Note: *The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.*

The **jcdTableSelect** Collaboration contains the Business Rules displayed in Figure 75.

Figure 75 jcdTableSelect



Sample code from the jcdTableSelect Includes:

```
package prjOracle_JCDjcdALL;
public class jcdTableSelect
{
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

public void receive( com.stc.connector.appconn.file.FileTextMessage
input, dtd.otdOutputDTD1325973702.DB_Employee
otdOutputDTD_DB_Employee_1, otdOracle.OtdOracleOTD otdOracle_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

\\ Writes out a message stating records are being selected.

FileClient_1.setText( "Selecting records from db_employee table via
Table Select........" );
        FileClient_1.write();

\\ Copies the database resultset into the otdInputDTD_DBEmployee (XML
OTD) and selects all available records from the database that meet the
criteria emp_no = 100. Checking the next() method ensures all rows are
retrieved in the while loop.

        otdOracle_1.getDb_employee().select( input.getText() );
        while (otdOracle_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.shortToString( otdOracle_1.
getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdOracle_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdOracle_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate(
otdSOracle_1.getDb_employee().getRATE().toString() );
```

```
            otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdOracle_1.getDb_employee().getLAST_UPDATE(), "yyyy-MM-dd hh:mm:ss",
false, "" ) );

\\ marshals XML data from the output data into the
otdOutputDTD_DB_Employee_1.marshallToString() method.

            FileClient_1.setText(
otdOutputDTD_DB_Employee_1.marshalToString() );
            FileClient_1.write();
        }

\\ Writes a message to confirm when records are selected, or when no
records are available.

        FileClient_1.setText( "Table Select Done." );
        FileClient_1.write();
    }
}
```

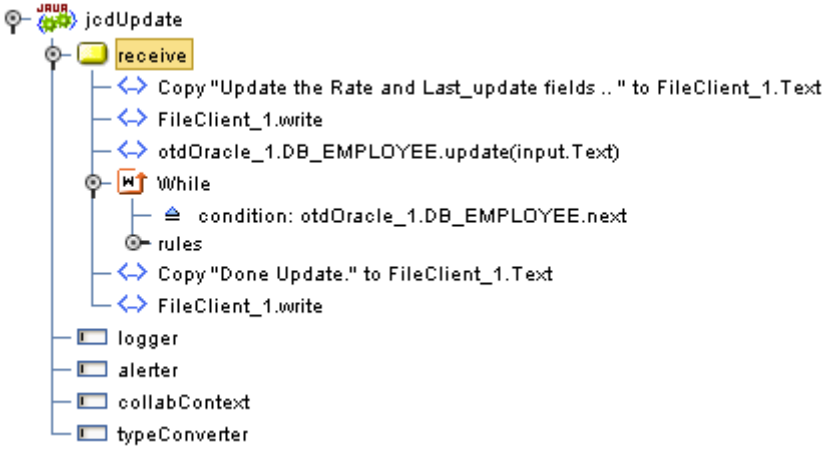
Creating the jcdUpdate Business Rules

The **jcdUpdate** Collaboration implements the Input Web Service Operation to read the **TriggerUpdate.in**. file and then update the record **emp_no = 300**. The Collaboration also writes a message to **JCD_Update_output0.dat** to confirm an updated record.

Note: *The where clause in the business rule reads the trigger value as a placholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerUpdate.in file is empty.*

The **jcdUpdate** Collaboration contains the Business Rules displayed in Figure 76.

Figure 76 jcdTableUpdate



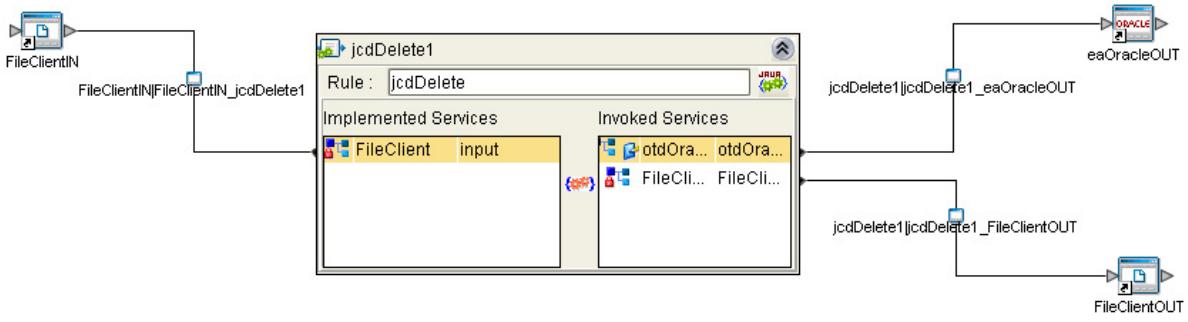
6.6.6 Binding the eWay Components

The final step in creating a Connectivity Map is binding the eWay components together.

Steps to bind eWay components together:

1. Double-click a Connectivity Map—in this example **cmDelete**—in the Project Explorer tree. The **cmDelete** Connectivity Map appears in the Enterprise Designers canvas.
2. Drag and drop the **jcdDelete** Collaboration from the Project Explorer to the **jcdDelete** Service. The Service icon "gears" change from red to green.
3. Double-click the **jcdDelete** Service. The **jcdDelete** Binding dialog box appears.
4. Map the input **FileClient** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **bpDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **bpDelete**.
5. From the **bpDelete** Binding dialog box, map **otdOracle_1** (under Invoked Services) to the **eaOracleOUT** External Application.
6. From the **bpDelete** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 68.

Figure 77 Connectivity Map - Associating (Binding) the Project's Components



7. Minimize the **jcdDelete** Binding dialog box by clicking the chevrons in the upper-right corner.
8. Save your current changes to the Repository, and then repeat this process for each of the other Connectivity Maps.

6.6.7 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Editor.

Steps required to create an Environment:

1. From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **envOracleProj**.

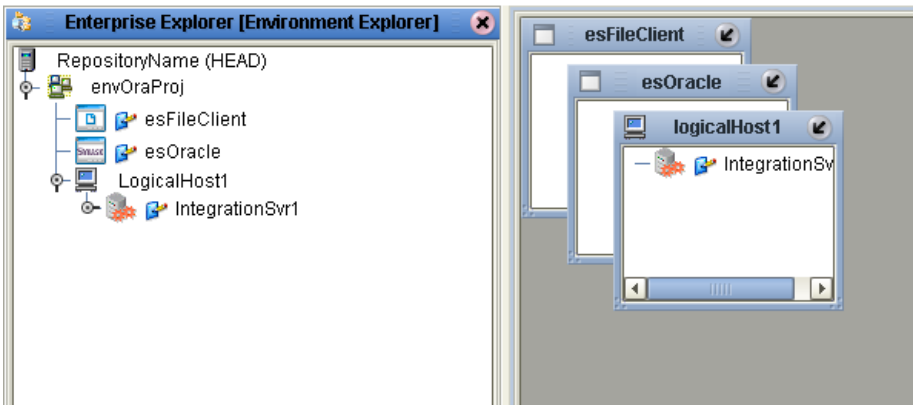
4 Right-click **envOracleProj** and select **New Oracle External System**. Name the External System **esOracle.** Click **OK**. **esOracle** is added to the Environment Editor.

5 Right-click **envOracleProj** and select **New File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.

6 Right-click **envOracleProj** and select **New Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

7 Right-click **LogicalHost1** and select **New Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see Figure 69).

Figure 78 Environment Editor - envOracleProj



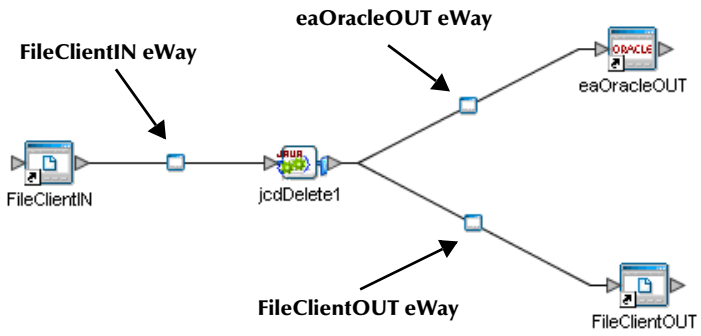
8 Save your current changes to the Repository.

6.6.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjOracle_BPEL** sample Project uses three eWays that are represented as nodes between the External Applications and the Business Process, as seen in Figure 70.

You must configure eWay properties in both the Connectivity Map and the Environment Explorer.

Figure 79 eWays in the cmDelete Connectivity Map



Configuring the eWay Properties

Steps required to configure the eWay properties:

1 Double-click the **FileClientIN eWay on each of the Connectivity Maps and** modify the properties for your system, as seen in Table 17. Click **OK** to close the Properties Editor.

Table 17 FileClientIN eWay Property Settings

Connectivity Map	Property Name	Required Value
cmDelete	Input file name	TriggerDelete.in
cmInsert	Input file name	TriggerInsert.in
cmPsSelect	Input file name	TriggerPsSelect.in
cmTableSelect	Input file name	TriggerTableSelect.in
cmUpdate	Input file name	TriggerUpdate.in

2 Double-click the **FileClientIN eWay on each of the Connectivity Maps and** modify the properties for your system, as seen in Table 18. Click **OK** to close the Properties Editor.

Table 18 FileClientOUT eWay Property Settings

Connectivity Map	Property Name	Required Value
cmDelete	Input file name	JCD_Delete_output%d.dat
cmInsert	Input file name	JCD_Insert_output%d.dat
cmPsSelect	Input file name	JCD_PsSelect_output%d.dat
cmTableSelect	Input file name	JCD_TableSelect_output%d.dat.in
cmUpdate	Input file name	JCD_Update_output%d.dat

Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

1. From the **Environment Explorer** tree, right-click the File External System (**esOracle** in this sample), and select **Properties**. The Properties Editor opens to the Oracle eWay Environment configuration.
2. Modify the Oracle eWay Environment configuration properties for your system, as seen in Table 19, and click **OK**.

Table 19 Oracle eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound Oracle eWay > JDBC Connector settings	ServerName	Enter the name of the database server being used.
	DatabaseName	Enter the name of the Oracle SID.
	User	Enter the user account name for the database.
	Password	Enter the user account password for the database.

3. From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the Oracle eWay Environment configuration.
4. Modify the File eWay Environment configuration properties for your system, as seen in Table 20, and click **OK**.

Table 20 File eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound File eWay > Parameter Settings	Directory	Enter the directory that contains the input files (trigger files included in the sample Project). Trigger files include: ▪ TriggerDelete.in.~in ▪ TriggerInsert.in.~in ▪ TriggerPsSelect.in.~in ▪ TriggerTableSelect.in.~in ▪ TriggerUpdate.in.~in

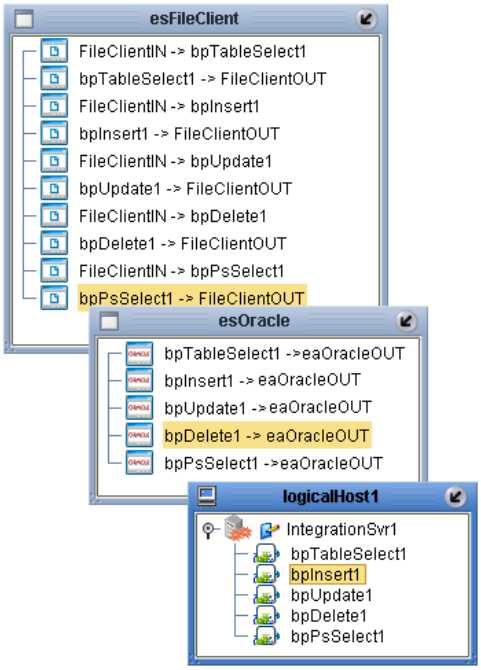
Section	Property Name	Required Value
Configuration > Outbound File eWay > Parameter Settings	Directory	Enter the directory where output files are written. In this sample Project, the output files include: ▪ JCD_Delete_output0.dat ▪ JCD_Insert_output0.dat ▪ JCD_PsSelect_output0.dat ▪ JCD_TableSelect_output0.dat ▪ JCD_Update_output0.dat

6.6.9 Creating the Deployment Profile

A Deployment Profile is used to assign services and message destinations to the integration server and message server. Deployment profiles are created using the Deployment Editor.

1. From the Enterprise Explorer's Project Explorer, right-click the **prjOracle_BPEL** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpOracle_BPEL**). Select **envOracleProj** as the Environment and click **OK**.
3. From the Deployment Editor toolbar, click the **Automap** icon. The Project's components are automatically mapped to their system windows, as seen in Figure 71.

Figure 80 Deployment Profile



6.6.10 Creating and Starting the Domain

To build and deploy your Project, you must first create a domain. A domain is an instance of a Logical Host. After the domain is created, the Project is built and then deployed.

Steps to create and start the domain include:

1. Navigate to your **<caps51>\logicalhost** directory (where <caps51> is the location of your Sun Java Composite Application Platform Suite installation.
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.
5. Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

For more information about creating and managing domains see the *eGate Integrator System Administration Guide*.

6.6.11 Building and Deploying the Project

The Build process compiles and validates the Project's Java files and creates the Project EAR file.

Build the Project

1. From the Deployment Editor toolbar, click the **Build** icon.
2. If there are any validation errors, a **Validation Errors** pane will appear at the bottom of the Deployment Editor and displays information regarding the errors. Make any necessary corrections and click **Build** again.
3. After the Build has succeeded you are ready to deploy your Project.

Deploy the Project

1. From the Deployment Editor toolbar, click the **Deploy** icon. Click **Yes** when the **Deploy** prompt appears.
2. A message appears when the project is successfully deployed. You can now test your sample.

6.6.12 Running the Sample

Perform the following steps to run your deployed sample Project:

1. Rename one of the trigger files included in the sample Project from **<filename>.in.~in** to **<filename>.in** to run the corresponding operation.

The File eWay polls the directory every five seconds for the input file name (as defined in the Inbound File eWay Properties window). The JCD then transforms the data, and the File eWay sends the output to an Output file name (as defined in the outbound File eWay Properties window).

The Where Clause defined in the business rule recognizes the trigger as a placeholder for input, allowing a set condition, such as emp_no = 100, to determine the type of output data.

You can modify the following input files to view different output.

- TriggerTableSelect.in
- TriggerDelete.in
- TriggerUpdate.in

Having no content in these files causes the operation to read all records.

2 Verify the output data by viewing the sample output files. See **About the Oracle eWay Sample Projects** on page 66 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.

Appendix A

Oracle eWay Data Types

This section lists conversions between the Oracle and OTD/Java datatypes, the types of methods to use for each datatype, and the value or size of the data element that can be used.

What's in this Appendix:

A.1 Supported Data Types

The Oracle eWay supports the following data types:

Table 1 Standard Data Types Supported by the Oracle eWay

Data Type	Description	Column Length
Number	Variable-length numeric data. Maximum precision p and/or scale s is 38.	Variable for each row. The maximum space required for a given column is 21 bytes per row.
VarChar2	Variable-length character data, with maximum length size bytes or characters.	Variable for each row, up to 4000 bytes per row. Consider the character set (single-byte or multi-byte) before setting size. You must specify a maximum size.
Char	Fixed-length character data of length size bytes or characters.	Fixed for every row in the table (with trailing blanks); maximum size is 2000 bytes per row, default size is one byte per row. Consider the character set (single-byte or multi-byte) before setting size.
Raw	Variable-length raw binary data.	Variable for each row in the table, up to 2000 bytes per row. You must specify a maximum size. Provided for backward compatibility.

Data Type	Description	Column Length
Long	Variable-length character data.	Variable for each row in the table, up to 2^{32} - one bytes, or two gigabytes, per row. Provided for backward compatibility.
Clob	Built-in data type that stores a Character Large Object as a column value in a row of a database table.	Up to 2^{32} - one bytes, or 64k.
Date	Fixed-length date and time data, ranging from Jan. 1, 4712 B.C.E. to Dec. 31, 4712 C.E.	Fixed at seven bytes for each row in the table. Default format is a string (such as DD-MON-RR) specified by the NLS_DATE_FORMAT parameter. Oracle expects a format of: "YYYY-MM-DD; hh:mm:ss.x".

A.2 Converting Data Types in the Oracle eWay

When working with data in the Oracle eWay OTD's, you may need to do a data conversion. The following tables show input/output data for data conversion:

Table 2 Insert and Update Operations Datatype Conversions (Text/String input data)

Oracle Data Type	OTD/Java Data Type	Java Method or New Constructor to Use (Default: Java Method)	Sample Data
Int	BigDecimal	**Call a New Constructor BigDecimal**: java.math.BigDecimal(String)	123
Smallint	BigDecimal	**Call a New Constructor BigDecimal**: java.math.BigDecimal(String)	123
Number	BigDecimal	**Call a New Constructor BigDecimal**: java.math.BigDecimal(String)	123
Decimal*	BigDecimal	**Call a New Constructor BigDecimal**: java.math.BigDecimal(String)	147.78
Real	Double	**Double**: java.lang.Double.parseDouble(String)	147.78
Float	Double	**Double**: java.lang.Double.parseDouble(String)	147.78
Double	Double	**Double**: java.lang.Double.parseDouble(String)	147.78

Oracle Data Type	OTD/Java Data Type	Java Method or New Constructor to Use (Default: Java Method)	Sample Data
Date	TimeStamp	**TimeStamp**: java.sql.TimeStamp.valueOf(String)	2003-08-11 11:47:39.0
TimeStamp	TimeStamp	**TimeStamp**: java.sql.TimeStamp.valueOf(String) **Note**: To use the TimeStamp data type, (allowing milliseconds to be stored) you must use the Oracle 10g driver (Oracle JDBC version 10.1.0.2.0).	11:47:39.001
Varchar2	String	Direct Assign	Any character
Char	String	Direct Assign	Any character
Long Char	String	Direct Assign	Any character
Raw	Byte[]	**String**: java.lang.String.getBytes()	Any character
Long Raw	Byte[]	**String**: java.lang.String.getBytes()	Any character
CLOB	Clob	See Appendix	Any character

Table 3 Select Operation Datatype Conversion (Text/String output data)

Oracle Data Type	OTD/Java Data Type	Methods To Use	Sample Data
Int	BigDecimal	**BigDecimal**: java.math.toString()	123
SmallInt	BigDecimal	**BigDecimal**: java.math.toString()	123
Number	BigDecimal	**BigDecimal**: java.math.toString()	123
Decimal	BigDecimal	**BigDecimal**: java.math.toString()	123.67
Real	Double	**Double**: java.lang.Double.toString(double)	123
Float	Double	**Double**: java.lang.Double.toString(double)	123
Double	Double	**Double**: java.lang.Double.parseDouble(String)	123

Oracle Data Type	OTD/Java Data Type	Methods To Use	Sample Data
Date	TimeStamp	**TimeStamp**: java.sql.TimeStamp.valueof()	2003-08-11 11:47:39.0
TimeStamp	TimeStamp	**TimeStamp**: java.sql.TimeStamp.valueOf(String) **Note**: To use the TimeStamp data type, (allowing milliseconds to be stored) you must use the Oracle 10g driver (Oracle JDBC version 10.1.0.2.0).	11:47:39.001
Varchar2	String	Direct Assign	Any characters
Char	String	Direct Assign	Any Characters
Raw	Byte[]	N/A	N/A
Long Raw	Byte[]	N/A	N/A
CLOB	Clob	N/A	N/A

Index

Q

S

T

U

V