SUN SEEBEYOND

eWay™ JDBC/ODBC Adapter USER'S GUIDE

Release 5.1.1

Sun microsystems

Version 20060614130417

Contents

Chapter 1

Introducing the JDBC/ODBC eWay 7

About Java Database Connectivity (JDBC) 7

JDBC Drivers 7

Type I: JDBC-ODBC Bridge 8

Type One Driver 8

Type II: Partial Java driver 9

Type Two Driver 9

Pure Java driver for database middleware 10

Type Three Driver 10

Type Four Driver: Direct-to-database pure Java driver 11

About the JDBC/ODBC eWay 13

What's New in This Release 13

About This Document 14

JDBC eWay Javadoc 14

Scope 14

Intended Audience 14

Text Conventions 15

Related Documents 15

Sun Microsystems, Inc. Web Site 15

Documentation Feedback 15

Chapter 2

Installing the JDBC/ODBC eWay 16

Installing the JDBC eWay 16

Installing the JDBC eWay on an eGate supported system 17

Adding the eWay to an Existing Sun Java Composite Application Platform Suite Installation 17

After Installation 18

Extracting the Sample Projects and Javadocs 18

ICAN 5.0 Project Migration Procedures 18

Installing Enterprise Manager eWay Plug-Ins 20

Viewing Alert Codes 21

Chapter 3

Setting Properties of the JDBC/ODBC eWay 24

Creating and Configuring a JDBC eWay 24

Configuring the eWay Connectivity Map Properties 24

Transaction Support Levels Between Different Versions 26

Configuring the eWay Environment Properties 28

eWay Connectivity Map Properties 29

Connectivity Map Outbound eWay Properties 30

Connectivity Map Outbound non-Transactional eWay Properties 30

Connectivity Map Outbound XA eWay Properties 30

eWay Environment Properties 30

Inbound JDBC eWay Properties 31

Outbound JDBC eWay Properties 31

JDBC Connector Settings 32

Connection Retry Settings 33

Outbound non-Transactional JDBC eWay Properties 34

JDBC Connector Settings 34

Connection Retry Settings 36

Outbound XA JDBC eWay Properties 36

JDBC Connector Settings 37

Connection Retry Settings 38

Chapter 4

Using the JDBC/ODBC eWay Database Wizard 40

About the Database OTD Wizard 40

Creating a New JDBC OTD 40

Select Wizard Type 41

Connect To Database 41

Select Database Objects 42

Select Tables/Views/Aliases 43

Select Procedures 47

Add Prepared Statement 51

Specify the OTD Name 54

Review Selections 54

Steps to Edit an Existing JDBC OTD 55

Chapter 5

Using JDBC/ODBC Operations 56

JDBC eWay Database Operations (BPEL) 56

Activity Input and Output 56

JDBC eWay Database Operations (JCD) 58

The Table 59

The Query (Select) Operation 59

The Insert Operation 60

The Update Operation 61

The Delete Operation 62

The Stored Procedure 62

Executing Stored Procedures 63

Manipulating the ResultSet and Update Count Returned by Stored Procedure 64

Prepared Statement 66

Batch Operations 66

Chapter 6

Implementing the JDBC/ODBC eWay Sample Projects 68

About the JDBC eWay Sample Projects 68

Sample Project Data 70

Operations Used in the JDBC Sample Projects 70

Assigning Operations in JCD 70

Assigning Operations in BPEL 70

About the eInsight Engine and eGate Components 71

Sample Projects Drivers 71

Running the Sample Projects 71

Running the SQL Script 72

Importing a Sample Project 72

Building and Deploying the prjJDBC_JCD Sample Project 73

Creating a Project 74

Creating the OTDs 74

Creating a Connectivity Map 75

Populating the Connectivity Map 76

Creating the Collaboration Definitions (Java) 77

jcdDelete Collaboration 77

jcdInsert Collaboration 78

jcdPsSelect Collaboration 78

jcdTableSelect Collaboration 79

jcdPsUpdate Collaboration 79

Create the Collaboration Business Rules 80

Creating the jcdDelete Business Rules 80

Creating the jcdInsert Business Rules 80

Creating the jcdPsSelect Business Rules 82

Creating the jcdTableSelect Business Rules 84

Creating the jcdUpdate Business Rules 86

Binding the eWay Components 86

Creating an Environment 87

Configuring the eWays 88

Configuring the eWay Properties 89

Configuring the Environment Explorer Properties 90

Configuring the Integration Server 91

Creating the Deployment Profile 91

Creating and Starting the Domain 92

Building and Deploying the Project 93
Running the Sample 93

Building and Deploying the prjJDBC_BPEL Sample Project 94
Creating a Project 94
Creating the OTDs 95
Creating the Business Process 96
Creating the Business Process Flow 96
Configuring the bpelPsInsert Modeling Elements 97
Configuring the bpelUpdate Modeling Elements 99
Configuring the bpelDelete Modeling Elements 101
Configuring the bpelTableSelect Modeling Elements 103
Creating the Connectivity Map 106
Populating the Connectivity Map 106
Binding the eWay Components 107
Creating an Environment 108
Configuring the eWays 109
Configuring the eWay Properties 110
Configuring the Environment Explorer Properties 110
Configuring the Integration Server 112
Creating the Deployment Profile 112
Creating and Starting the Domain 113
Building and Deploying the Project 114
Running the Sample Project 114

Appendix A

JDBC/ODBC Drivers 116

Index 130

Chapter 1

Introducing the JDBC/ODBC eWay

Welcome to the *Sun SeeBeyond eWay™ JDBC/ODBC Adapter User's Guide.* This document includes information about installing, configuring, and using the Sun Java Composite Application Platform Suite JDBC/ODBC eWay™ Adapter, referred to as the JDBC eWay throughout this guide.

This chapter provides an overview of database connectivity (JDBC) and open database connectivity (ODBC) APIs. This chapter also introduces the JDBC/ODBC eWay.

What's In This Chapter

- **About Java Database Connectivity (JDBC)** on page 7
- **About the JDBC/ODBC eWay** on page 13
- **What's New in This Release** on page 13
- **About This Document** on page 14
- **Related Documents** on page 15
- **Sun Microsystems, Inc. Web Site** on page 15
- **Documentation Feedback** on page 15

1.1 About Java Database Connectivity (JDBC)

Java Database Connectivity (JDBC) is an implementation of the Java programming language that dictates how databases communicate with each other. Through a standardized application programming interface (API), connectivity from database management systems (DBMS) to a wide range of SQL databases is accomplished. By deploying database drivers laced with JDBC technology, it is possible to connect to any database -- even in a heterogeneous environment -- and access tables, tabular data, flat files and more. When using JDBC, Java programmers have the ability to request connections to a database, send queries to the database using SQL statements, and receive results for advanced processing.

1.1.1 JDBC Drivers

To connect with individual databases, JDBC requires drivers for each database. Those drivers come in four varieties. Driver types 1 and 2 are typically intended for programmers that write applications. Driver types 3 and 4 are typically used by database and middleware vendors. The various driver types are described in the following sections:

- **"Type I: JDBC-ODBC Bridge" on page 8**
- **"Type II: Partial Java driver" on page 9**
- **"Pure Java driver for database middleware" on page 10**
- **"Type Four Driver: Direct-to-database pure Java driver" on page 11**

Type I: JDBC-ODBC Bridge

This combination provides JDBC access via ODBC drivers. ODBC binary code--and in many cases, database client code--must be loaded on each client machine that uses a JDBC-ODBC Bridge. A product called SequeLink from Data Direct Technologies provides a driver that supports some ODBC drivers (for example Microsoft Access).

Type one drivers provide JDBC access via one or more Open Database Connectivity (ODBC) drivers. ODBC, which predates JDBC, is widely used by developers to connect to databases in a non-Java environment.

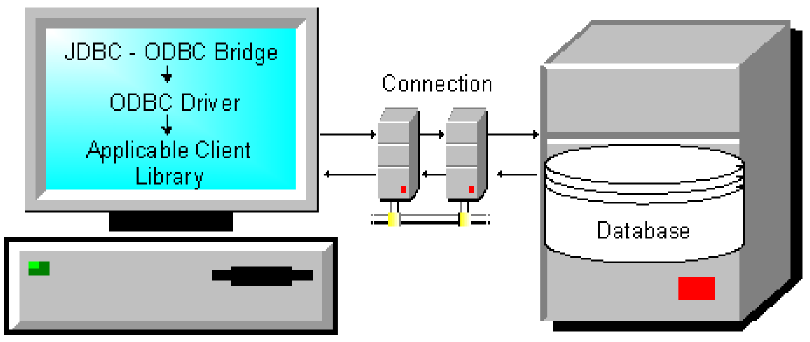
Pros: A good approach for learning JDBC. May be useful for companies that already have ODBC drivers installed on each client machine — typically the case for Windows-based machines running productivity applications. May be the only way to gain access to some low-end desktop databases.

Cons: Not for large-scale applications. Performance suffers because there's some overhead associated with the translation work to go from JDBC to ODBC. Doesn't support all the features of Java. User is limited by the functionality of the underlying ODBC driver.

Type One Driver

A JDBC/ODBC bridge provides JDBC API access through one or more ODBC drivers. Some ODBC native code and in many cases native database client code must be loaded on each client machine that uses this type of driver.

Figure 1 Typical Type 1 Driver Configuration



The pros and cons for using this type of driver are as follows:

Pros

- Allows access to almost any database since the database ODBC drivers are readily available

Cons

- Performance is degraded since the JDBC call goes through the bridge to the ODBC driver then to the native database connectivity interface. The results are then sent back through the reverse process
- Limited Java feature set
- May not be suitable for a large-scale application

Type II: Partial Java driver

This type of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine.

This type of driver converts the calls that a developer writes to the JDBC application programming interface into calls that connect to the client machine's application programming interface for a specific database, such as IBM, Informix, Oracle or Sybase.

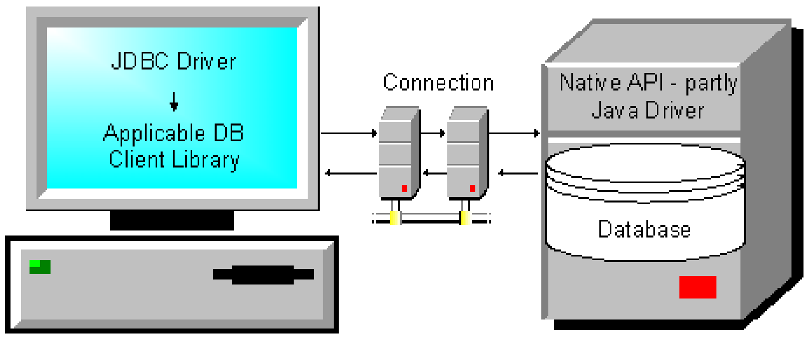
Pros: Performance is better than that of Type 1, in part because the Type 2 driver contains compiled code that's optimized for the back-end database server's operating system.

Cons: User needs to make sure the JDBC driver of the database vendor is loaded onto each client machine. Must have compiled code for every operating system that the application will run on. Best use is for controlled environments, such as an intranet.

Type Two Driver

A native-API partly Java technology-enabled driver converts JDBC calls into calls on the client API for DBMSs. Like the bridge driver, this style of driver requires that some binary code be loaded on each client machine. An example of this type of driver is the Oracle Thick Driver, which is also called OCI (see **JDBC/ODBC Drivers** on page 116 regarding JDBC eWay support of Oracle drivers).

Figure 2 Typical Type 2 Driver Configuration



The pros and cons for using this type of driver are as follows:

Pros

- Allows access to almost any database since the databases ODBC drivers are readily available
- Offers significantly better performance than the JDBC/ODBC Bridge
- Limited Java feature set

Cons

- Applicable Client library must be installed
- Type 2 driver shows lower performance than type 3 or 4

Pure Java driver for database middleware

This style of driver translates JDBC calls into the middleware vendor's protocol, which is then translated to a DBMS protocol by a middleware server. The middleware provides connectivity to many different databases.

This driver translates JDBC calls into the middleware vendor's protocol, which is then converted to a database-specific protocol by the middleware server software.

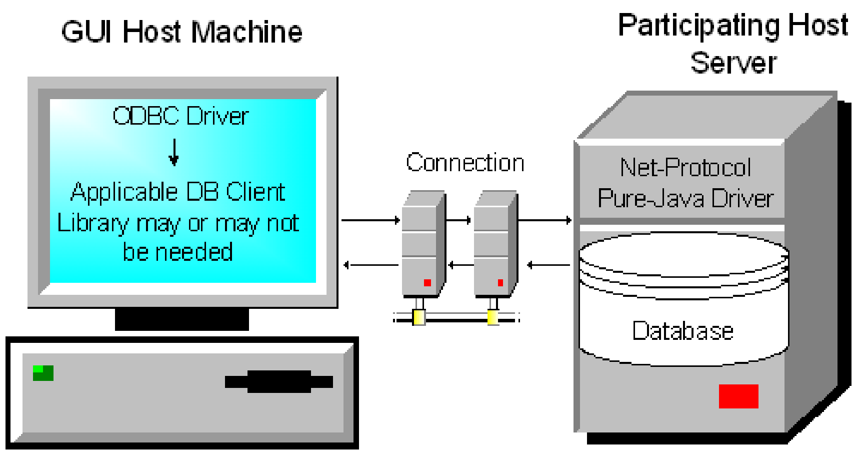
Pros: Better performance than Types 1 and 2. Can be used when a company has multiple databases and wants to use a single JDBC driver to connect to all of them. Server-based, so no need for JDBC driver code on client machine. For performance reasons, the back-end server component is optimized for the operating system on which the database is running.

Cons: Needs some database-specific code on the middleware server. If the middleware must run on different platforms, a Type 4 driver might be more effective.

Type Three Driver

A net-protocol fully Java-enabled driver translates JDBC API calls into a DBMS-independent net protocol which is then translated to a DBMS protocol by a server. This net server middleware is able to connect all of its Java technology-based clients to many different databases. Many mainframe legacy non-relational databases use this kind of driver.

Figure 3 Typical Type 3 Middleware Driver Configuration


GUI Host Machine
ODBC Driver
Applicable DB Client Library may or may not be needed
Connection
Participating Host Server
Net-Protocol Pure-Java Driver
Database

The pros and cons for using this type of driver are as follows:

Pros

- Allows access to almost any database since the databases ODBC drivers are readily available
- Offers significantly better performance than the JDBC/ODBC Bridge and Type 2 Drivers
- Advanced Java feature set
- Scalable
- Caching
- Advanced system administration
- Does not require applicable database client libraries

Cons

- Requires a separate JDBC middleware server to translate specific native-connectivity interface.

Type Four Driver: Direct-to-database pure Java driver

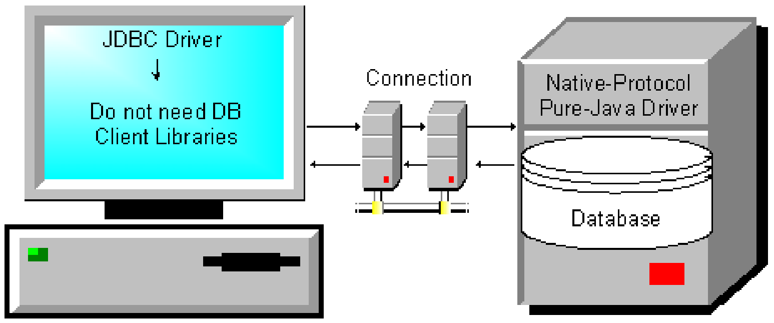
This style of driver converts JDBC calls into a network protocol that sends the converted packets--in a proprietary format--to be used directly by DBMSs, thus allowing a direct call from the client machine to the DBMS server and providing a practical solution for intranet access. This type of driver has become very popular recently and is supported by most database software vendors. All JDBC drivers from Data Direct Technologies (driver vendor) are Type 4 drivers.

Pros: Better performance than Types 1 and 2. No need to install special software on client or server.

Cons: Not optimized for server operating system, so the driver can't take advantage of operating system features. (The driver is optimized for the database and can take advantage of the database vendor's functionality.) User needs a different driver for each different database.

A native-protocol fully Java technology-enabled driver converts JDBC technology calls into the network protocol used by DBMSs directly. This allows a direct call from the client machine to the DBMS server.

Figure 4 Typical Type 4 Driver Configuration


JDBC Driver
Do not need DB Client Libraries
Connection
Native-Protocol Pure-Java Driver
Database

The pros and cons for using this type of driver are as follows:

Pros

- Allows access to almost any database since the databases ODBC drivers are readily available
- Offers significantly better performance than the JDBC/ODBC Bridge and Type 2 Drivers
- Scalable
- Caching
- Advanced system administration
- Superior performance
- Advance Java feature set
- Does not require applicable database client libraries

Cons

- Each database will require a driver

1.2 About the JDBC/ODBC eWay

This document describes how to install and configure the JDBC/ODBC eWay. The JDBC/ODBC eWay enables the eGate system to exchange data with external databases. The Sun Java Composite Application Platform Suite (Java CAPS) contains many database eWays. You should use those eWays to interface with the databases they support. The JDBC/ODBC eWay should only be used when you have a driver or a database that is not supported by those eWays.

The JDBC/ODBC eWay uses Java Collaborations to interact with one or more external databases. By using a Java Collaboration Service it is possible for eGate components such as eWay Adapters to connect to external databases and execute business rules.

1.3 What's New in This Release

The JDBC eWay includes the following changes and new features:

- 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 to read configuration parameters from LDAP at runtime.
- 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.
- Relaunchable OTD Support: An OTD can be rebuilt and saved (under the same name) then relaunched back to the same Java Collaboration or BPEL. This allows you to change the metadata in an OTD without having to completely recreate the business logic from scratch.
- Editable OTD Support: An existing OTD can be edited and saved using the OTD Wizard. This allows you to make minor changes to an OTD without having to completely recreate the OTD from scratch. The OTD is then rebuilt, saved, and then relaunched back to the same Java Collaboration or BPEL.
- 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 Administration Guide*.

1.4 About This Document

This document includes the following chapters:

- **Chapter 1 "Introducing the JDBC/ODBC eWay"**: Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 "Installing the JDBC/ODBC eWay"**: Describes the system requirements and provides instructions for installing the JDBC eWay.
- **Chapter 3 "Setting Properties of the JDBC/ODBC eWay"**: Provides instructions for configuring the eWay to communicate with JDBC drivers.
- **Chapter 4 "Using the JDBC/ODBC eWay Database Wizard"**: Provides instructions for creating Object Type Definitions to be used with the JDBC eWay.
- **Chapter 5 "Using JDBC/ODBC Operations"**: Provides instructions on using JDBC database eWay operations in BPEL and JCD.
- **Chapter 6 "Implementing the JDBC/ODBC eWay Sample Projects"**: Provides instructions for installing and running the sample Projects.
- **Appendix A "JDBC/ODBC Drivers"**: Provides instructions for installing and running database drivers.

JDBC eWay Javadoc

A JDBC eWay Javadoc is also provided that documents the Java methods available with the JDBC eWay. The Javadoc is uploaded with the eWay's documentation file (**JDBCeWayDocs.sar**) and downloaded from the **Documentation** tab of the Sun Java Composite Application Platform Suite Installer. To access the full Javadoc, extract the Javadoc to an easily accessible folder, and double-click the **index.html** file.

1.4.1 Scope

This user's guide provides a description of the JDBC eWay Adapter. It includes directions for installing the eWay, configuring the eWay properties, and implementing the eWay's sample Projects. This document is also intended as a reference guide, listing available properties, functions, and considerations. For a reference of available JDBC eWay Java methods, see the associated Javadoc.

1.4.2 Intended Audience

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

1.4.3 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 15
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.5 Related Documents

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

- *Sun SeeBeyond 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 JDBC/ODBC eWay

This chapter describes how to install the JDBC/ODBC eWay.

What's in this Chapter

- **Installing the JDBC eWay** on page 16
- **ICAN 5.0 Project Migration Procedures** on page 18
- **Installing Enterprise Manager eWay Plug-Ins** on page 20

2.1 Installing the JDBC eWay

The Java Composite Application Platform Suite Installer, referred to throughout this guide as the Suite Installer, is a web-based application that is used to select and upload core products, composite applications, and add-on files (eWays) 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

The JDBC eWay Readme is uploaded with the eWay's documentation file (**JDBCeWayDocs.sar**) and can be accessed from the **Documentation** tab of the Suite Installer.

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

2.1.1 Installing the JDBC eWay on an eGate supported system

Follow the directions for installing the Sun Java Composite Application Platform Suite in the *Sun Java Composite Application Platform Suite Installation Guide.* After you have installed Core Products, do the following:

1. From the Sun Java Composite Application Platform Suite Installer's **Select Sun Java Composite Application Platform Suite Products Installed** table (Administration tab), click the **Click to install additional products** link.
2. Expand the **eWay** option.
3. Select the products for your Sun Java Composite Application Platform Suite and include the following:
   - **File eWay** (the File eWay is used by most sample Projects)
   - **JDBCeWay**

   To upload the JDBC eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, expand the **Documentation** option and select **JDBCeWayDocs**.
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**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.
6. Once your product's installation is finished, continue installing the Sun Java Composite Application Platform Suite as instructed in the *Sun Java Composite Application Platform Suite Installation Guide*.

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

If you are adding the eWay to an existing Sun Java Composite Application Platform Suite installation, do the following:

1. Complete steps 1 through 4 above.
2. Once your product's installation is complete, 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.**

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

After Installation

Once you install the eWay, it must then be incorporated into a Project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

2.1.2 Extracting the Sample Projects and Javadocs

The JDBC 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 Sun Java Composite Application Platform Suite Installer, then click the Add-ons tab.

2 Click the **JDBC eWay Adapter** link. Documentation for the JDBC 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 Sun Java Composite Application Platform Suite Installer, then click the **Add-ons** tab.

2 Click the **JDBC eWay Adapter** link. Documentation for the JDBC eWay appears in the right pane.

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

Refer to **"Importing a Sample Project" on page 72** for instructions on importing the sample Project into your repository via the Enterprise Designer.

2.2 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0.x Projects to the Sun Java Composite Application Platform Suite 5.1.1. To migrate your ICAN 5.0.x Projects to the Sun Java Composite Application Platform Suite 5.1.1, do the following:

Export the Project

1 Before you export your Projects, save your current ICAN 5.0.x Projects to your Repository.

2. From the Project Explorer, right-click your Project and select **Export** from the shortcut menu. The Export Manager appears.
3. Select the Project that you want to export in the left pane of the Export Manager and move it to the Selected Projects field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.
4. In the same manner, select the Environment that you want to export in the left pane of the Export Manager and move it to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.
5. Browse to select a destination for your Project ZIP file and enter a name for your Project in the **ZIP file** field.
6. Click **Export** to create the Project ZIP file in the selected destination.

Install Java CAPS 5.1.1

1. Install **Java CAPS 5.1.1**, including all eWays, libraries, and other components used by your ICAN 5.0 Projects.
2. Start the Java CAPS 5.1.1 Enterprise Designer.

Import the Project

1. From the Java CAPS 5.1.1 Enterprise Designer's Project Explorer tree, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.
2. Browse to and select your exported Project file.
3. Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository when the Project was created that are not installed on the Java CAPS 5.1.1 Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.
4. Close the Import Manager after the Project is successfully imported.

Deploy the Project

1. A new Deployment Profile must be created for each of your imported Projects. When a Project is exported, the Project's components are automatically *"checked in"* to Version Control to write-protected each component. These protected components appear in the Explorer tree with a red padlock in the bottom-left corner of each icon. Before you can deploy the imported Project, the Project's components must first be *"checked out"* of Version Control from both the Project Explorer and the Environment Explorer. To *"check out"* all of the Project's components, do the following:

   A. From the Project Explorer, right-click the Project and select **Version Control > Check Out** from the shortcut menu. The Version Control - Check Out dialog box appears.

   B. Select **Recurse Project** to specify all components, and click **OK**.

C Select the Environment Explorer tab, and from the Environment Explorer, right-click the Project's Environment and select **Version Control > Check Out** from the shortcut menu.

D Select **Recurse Environment** to specify all components, and click **OK**.

2 If your imported Project includes File eWays, these must be reconfigured in your Environment prior to deploying the Project.

To reconfigure your File eWays, do the following:

A From the Environment Explorer tree, right-click the File External System, and select **Properties** from the shortcut menu. The Properties Editor appears.

B Set the inbound and outbound directory values, and click **OK**. The File External System can now accommodate both inbound and outbound eWays.

3 Deploy your Projects.

Note: *Only projects developed on ICAN 5.0.2 and later can be imported and migrated successfully into the Sun Java Composite Application Platform Suite.*

2.3 Installing Enterprise Manager eWay Plug-Ins

The **Sun SeeBeyond Enterprise Manager** is a Web-based interface you use to monitor and manage your Sun Java Composite Application Platform Suite applications. The Enterprise Manager requires an eWay specific "plug-in" for each eWay you install. These plug-ins enable the Enterprise Manager to target specific alert codes for each eWay type, as well as 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 Sun Java Composite Application Platform Suite Installer's **Downloads** tab.

There are two ways to add 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** sub-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 **Downloads** 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-ins is installed and deployed.

2.3.1 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** sub-tab. Your installed eWay alert codes display under the **Results** section. If your eWay alert codes are not displayed under **Results**, do the following:
   - A From the **Install New Alert Codes** section, browse to and select the eWay 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.
   - B Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of the eWay's available alert codes is displayed in Table 2.

Table 2 Alert Codes for the JDBC/ODBC 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 establishment.	▪ 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 the connection pool.	▪ Verify that the database has not terminated with unexpected errors.

Alert Code\Description	Description Details	User Actions
DBCOMMON-CONNECT-FAILED000005=Connection handle not usable. Reason:[{0}]	The connection in the pool is stale and is not usable.	▪ 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 Transaction Support.
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 establishment.	▪ 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.
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.

For information on Managing and Monitoring alert codes and logs, as well as how to view the alert generated by the project component during runtime, see the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

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.*

Chapter 3

Setting Properties of the JDBC/ODBC eWay

This chapter describes how to set the properties of the JDBC/ODBC eWay.

What's In This Chapter

- **Creating and Configuring a JDBC eWay** on page 24
- **Configuring the eWay Connectivity Map Properties** on page 24
- **Configuring the eWay Environment Properties** on page 28
- **eWay Connectivity Map Properties** on page 29
- **eWay Environment Properties** on page 30

3.1 Creating and Configuring a JDBC eWay

All eWays contain a unique set of default configuration parameters. After the eWays are established and a JDBC External System is created in the Project's Environment, the eWay parameters are modified for your specific system. The JDBC eWay configuration parameters are modified from two 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 JDBC External System window.
- **Collaboration or Business Process**: JDBC 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 Connectivity Map configuration. Any properties that are not overridden retain their configured default settings.

3.2 Configuring the eWay Connectivity Map Properties

When you connect an External Application to a Collaboration, Enterprise Designer automatically assigns the appropriate eWay to the link. Each eWay is supplied with a list of eWay connections (transaction support levels) from which to choose.

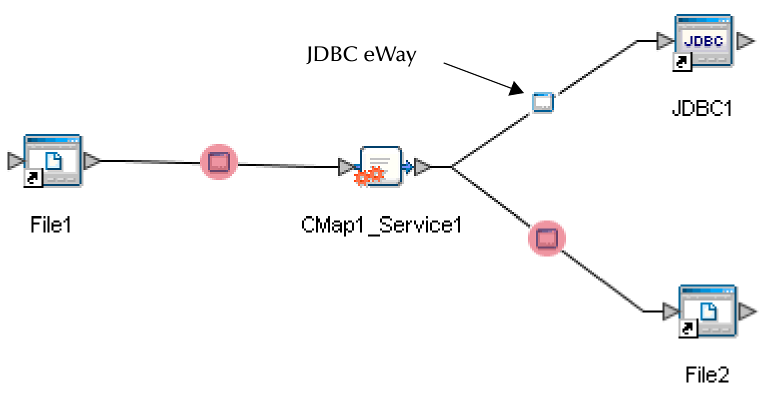
Transaction support levels provided by the JDBC eWay include:

- Outbound JDBC eWay
- Outbound JDBC XA eWay
- Outbound JDBC non-Transactional eWay

To configure the eWay properties:

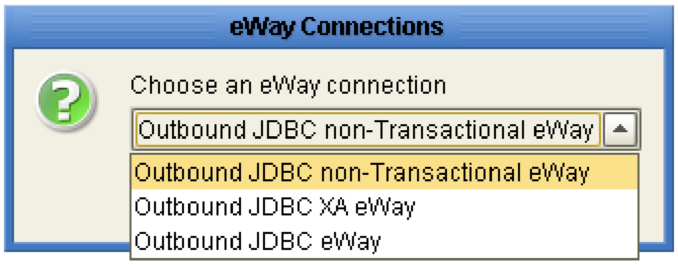
1 On the Enterprise Designer's Connectivity Map, double-click the JDBC eWay icon. The eWay Connections window appears.

Figure 5 Connectivity Map with Components



2 Select a parameter from the list and click **OK**.

Figure 6 Template window


eWay Connections

Choose an eWay connection

Outbound JDBC non-Transactional eWay

Outbound JDBC non-Transactional eWay
Outbound JDBC XA eWay
Outbound JDBC eWay

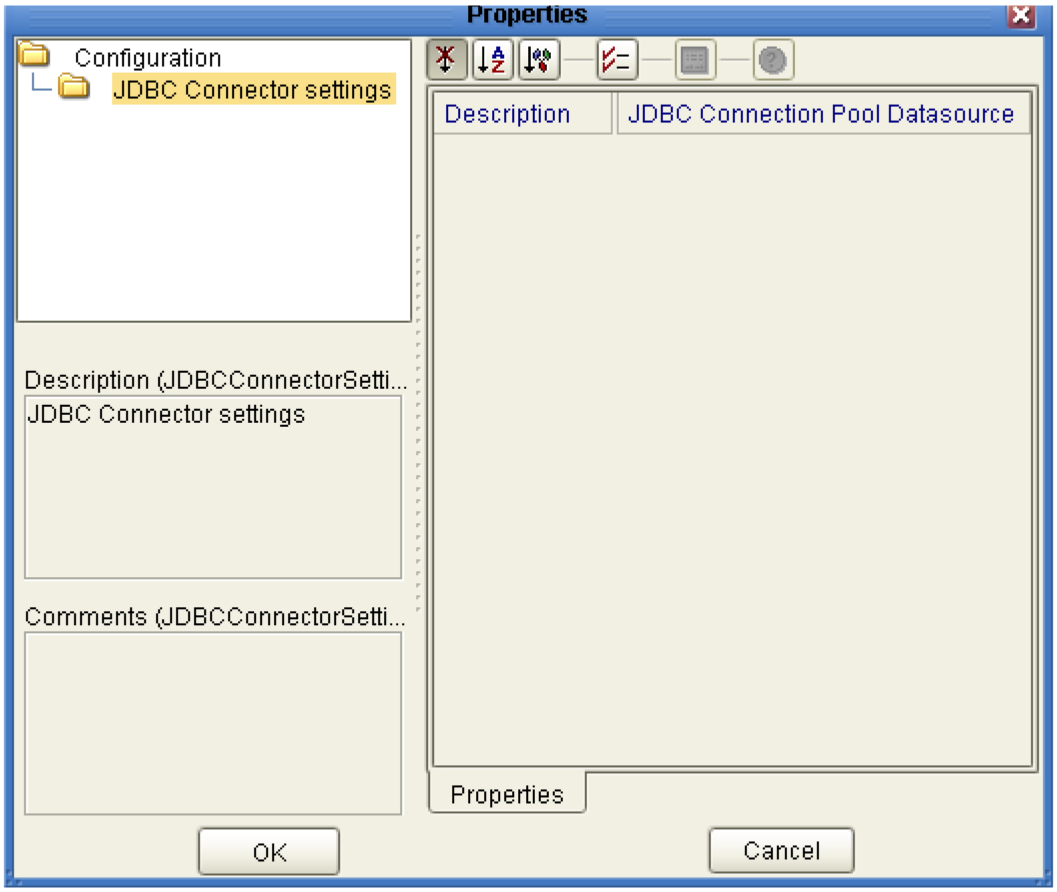
The choices to make are as follows:

- **Outbound JDBC 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 JDBC 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.

- **Outbound JDBC eWay:** Also referred to as LocalTransaction, this support level is opposite to NoTransaction, and this means that the transaction, when The Properties window opens, displaying the default properties for the eWay.

3 The Properties window opens, displaying the default properties for the eWay.

Figure 7 Outbound eWay Properties



3.2.1 Transaction Support Levels Between Different Versions

The types of transaction support levels used in Java CAPS 5.1.0 may be different from the support levels used in Java CAPS 5.1.1. Projects that are imported from a Java CAPS 5.1.0 version can potentially display different results, depending on whether the 5.1.0 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 5.1.1, 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 5.1.0 versions, you can choose the method below:

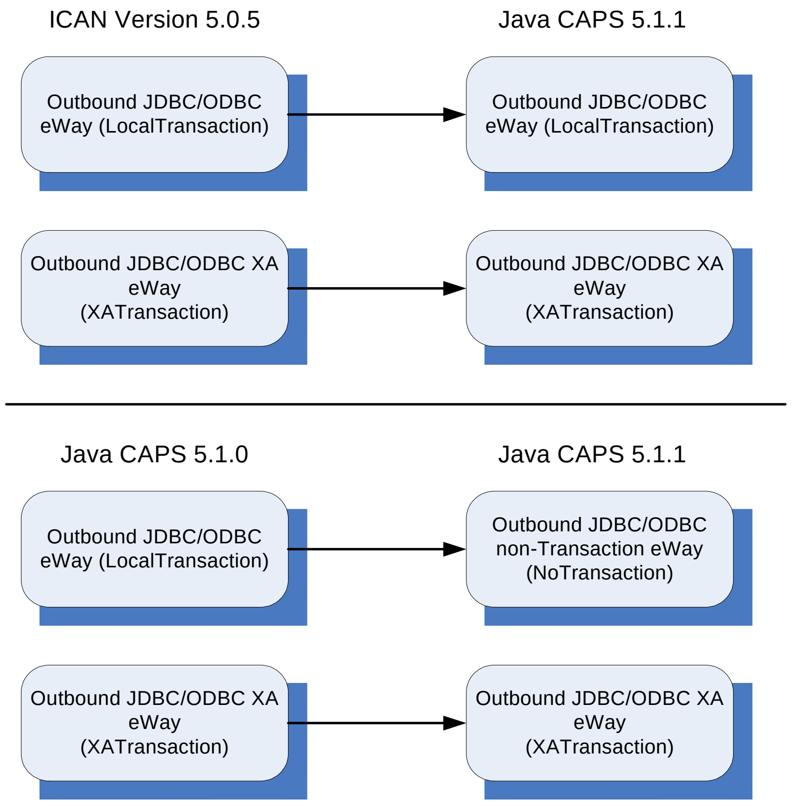
A In the Connectivity Map, delete the link to the database external application, then reconnect the link and select NoTransaction.

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 5.1.1, and 5.1.0 and 5.1.1, respectively. **Note that there are no changes when migrating from ICAN version 5.0.5 and Java CAPS 5.1.1.**

Figure 8 Transaction Support Levels


ICAN Version 5.0.5
Java CAPS 5.1.1
Outbound JDBC/ODBC eWay (LocalTransaction)
Outbound JDBC/ODBC eWay (LocalTransaction)
Outbound JDBC/ODBC XA eWay (XATransaction)
Outbound JDBC/ODBC XA eWay (XATransaction)
Java CAPS 5.1.0
Java CAPS 5.1.1
Outbound JDBC/ODBC eWay (LocalTransaction)
Outbound JDBC/ODBC non-Transaction eWay (NoTransaction)
Outbound JDBC/ODBC XA eWay (XATransaction)
Outbound JDBC/ODBC XA eWay (XATransaction)

Under the scenario noted above, if you want 5.1.1 behavior for a LocalTransaction, then set your eWay connection to be Outbound JDBC/ODBC non-Transactional eWay (NoTransaction).

3.3 Configuring the eWay Environment Properties

The eWay Environment Configuration properties contain parameters that define how the eWay connects to and interacts with other eGate components within the Environment. When you create a new JDBC External System, you may configure the type of External System required.

Available External System properties include:

- Inbound JDBC eWay
- Outbound JDBC eWay
- Outbound JDBC non-Transactional eWay
- Outbound JDBC XA eWay

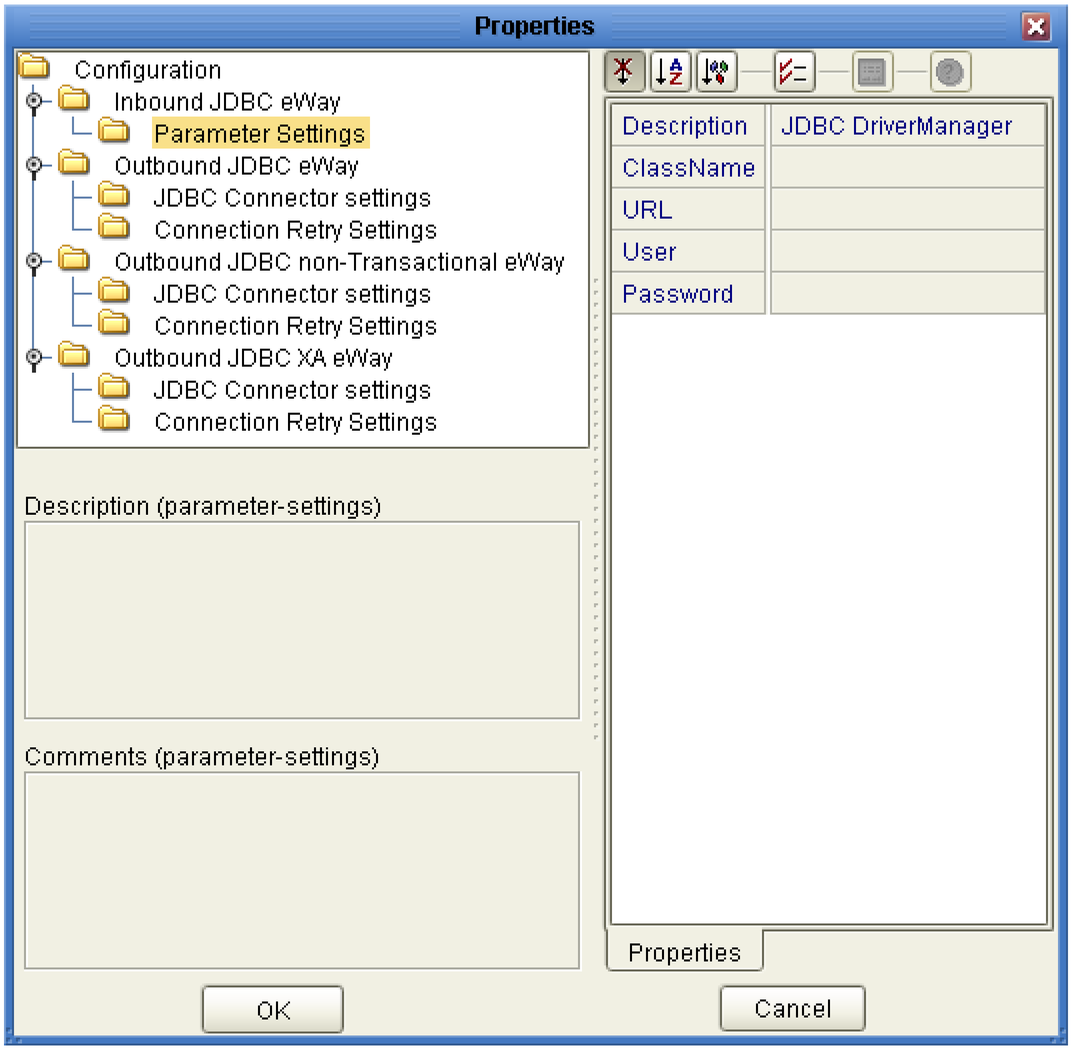
To Configure the Environment Properties:

1. In Enterprise Explorer, click the Environment Explorer tab.
2. Expand the Environment created for the JDBC Project and locate the JDBC External System.

Note: *For more information on creating an Environment, see the "Sun SeeBeyond eGate Integrator Tutorial".*

3. Right-click the External System created for the JDBC Project and select Properties from the list box. The Environment Configuration Properties window appears.

Figure 9 JDBC eWay Environment Configuration



Description	JDBC DriverManager
ClassName	
URL	
User	
Password	

4 Click on any folder to display the default configuration properties for that section.

5 Click on any property field to make it editable.

After modifying the configuration properties, click **OK** to save the changes.

3.4 eWay Connectivity Map Properties

The eWay Connectivity Map consists of the following properties categories.

Outbound eWay Configuration Sections Include:

- **Connectivity Map Outbound eWay Properties** on page 30

Outbound non-Transactional eWay Configuration Settings Include:

- **Connectivity Map Outbound non-Transactional eWay Properties** on page 30

Outbound XA eWay Configuration Settings Include:

- **Connectivity Map Outbound XA eWay Properties** on page 30

3.4.1 Connectivity Map Outbound eWay Properties

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

Table 3 Outbound eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	A valid string.

3.4.2 Connectivity Map Outbound non-Transactional eWay Properties

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	The description of the database.	Any valid string.

3.4.3 Connectivity Map Outbound XA eWay Properties

The Outbound XA eWay Properties include outbound parameters used by the external database.

Table 5 Outbound XA eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	Any valid string.

3.5 eWay Environment Properties

eWay External System properties must be configured from within the Environment. Until you have successfully configured all eWays for your Java CAPS project, your project cannot be properly executed. The following list identifies the JDBC eWay properties. There are four eWay connection types that the JDBC/ODBC eWay implements.

Property Categories Configured in the Logical Host Environment

- **Inbound JDBC eWay Properties** on page 31

- **Outbound JDBC eWay Properties** on page 31
- **Outbound non-Transactional JDBC eWay Properties** on page 34
- **Outbound XA JDBC eWay Properties** on page 36

3.5.1 Inbound JDBC eWay Properties

Before deploying your eWay, you will need to set the Environment properties. The Inbound JDBC eWay includes the following configuration section:

- Parameter Settings

Details for the Inbound JDBC eWay Parameter Settings are listed in Table 6.

Table 6 Inbound JDBC eWay—Parameter Settings

Name	Description	Required Value
Description	The description of the database.	A valid string.
ClassName	Displays the Java class in the JDBC driver that is used to implement the Driver Manager interface. Change this as needed for your driver.	A valid class name. See **JDBC/ODBC Drivers** on page 116 for some of the popular drivers you can use with this eWay.
URL	This is the JDBC URL required to gain access to the database. The URL usually starts with **jdbc**; followed by <subprotocol> and ends with information that identifies the data source, as follows: **jdbc:<driver>:<data-source-name>[;<attribute-name>=<attribute-value>]** If you do not select URL in the **connection method** this parameter is ignored. For more information on the JDBC URL, please consult the documentation of your specific driver.	The applicable JDBC URL. See **JDBC/ODBC Drivers** on page 116 for some of the popular drivers you can use with this eWay.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

3.5.2 Outbound JDBC eWay Properties

The Outbound JDBC eWay includes the following configuration sections:

- JDBC Connector Settings
- Connection Retry Settings

JDBC Connector Settings

Details for the JDBC Connector Settings used by the external database are detailed in Table 7.

Table 7 Outbound JDBC eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	A valid string. The configured default is **JDBC Connection Pool Datasource**.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the **ConnectionPoolDataSource**, **XADataSource** interface. Change as needed for your driver.	A valid class name. For example: **com.ibm.as400.access.AS400JDBCConnectionPoolDataSource** See **JDBC/ODBC Drivers** on page 116 for some of the popular drivers you can use with this eWay.
ClassNamefor OtherInterfaces	For Sun Java Composite Application Platform Suite components such as eTL that do not use **ConnectionPoolDataSource**, use this entry to enter the name of the driver class. This entry is not used by the JDBC eWay.	A valid class name.
ServerName	This setting 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.
DatabaseName	Specifies the name of the database instance.	Any valid string.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties	The Connection Pool 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. You must ensure that the driver is installed on both the Logical Host machine and the Enterprise Designer machine.	Set the driver properties according to driver vendor's instruction. For example: **setDefTdpName#DBSQL##setWorkspace#Navigator##**
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**.

Table 7 Outbound JDBC eWay—JDBC Connector Settings (Continued)

Name	Description	Required Value
DataSource name	Specifies the name of the **XADataSource** or **ConnectionPoolDataSource** implementation, to which the DataSource object delegates behind the scenes when there is connection pooling or distributed transaction management being done.	The name of the **XADataSource** or **ConnectionPoolDataSource** implementation. This property is Optional. In most cases, leave this box empty.
MinPoolSize	Specifies the minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size you set depends on the transaction volume and response time of the application. 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 **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. If the pool size is too big, you may end up with too many connections with the database. The pool size depends on the transaction volume and response time.	A valid numeric value. The default is **10**.
MaxIdleTime	Specifies 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

Details for the Connection Retry Settings used by the external database are detailed in Table 8.

Table 8 Outbound JDBC eWay—Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection upon failure to acquire one.	A valid numeric value. The default is **0**.

Table 8 Outbound JDBC eWay—Connection Retry Settings (Continued)

Name	Description	Required Value
ConnectionRetryInterval	Specifies the milliseconds of pause before each attempt to reaccess 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 10 times in 5 seconds apart when the Connection Retries is 10 and the Connection Retry Interval is 5000	A valid numeric value. The default is **1000**.

3.5.3 Outbound non-Transactional JDBC eWay Properties

The Outbound non-Transactional JDBC eWay includes the following configuration sections:

- JDBC Connector Settings
- Connection Retry Settings

JDBC Connector Settings

Details for the JDBC Connector Settings used by the external database are detailed in Table 9.

Table 9 Outbound non-Transactional eWay —JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	A valid string. The configured default is **JDBC non-Transactional Connection Pool Datasource**.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the **ConnectionPoolDataSource/XADataSource** interface. Change as needed for your driver.	A valid class name. For example: **com.ddtek.jdbcx.sequelink.SequeLinkDataSource**.
ClassNamefor OtherInterfaces	For Sun Java Composite Application Platform Suite components that do not use **ConnectionPoolDataSource**, use this entry to enter the name of the driver class. This class will not be used by the eWay.	A valid class name.
ServerName	This setting specifies the host name of the external database server.	Any valid string.

Table 9 Outbound non-Transactional eWay —JDBC Connector Settings (Continued)

Name	Description	Required Value
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number.
DatabaseName	Specifies the name of the database instance.	Any valid string.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties	The Connection Pool 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. You must ensure that the driver is installed on both the Logical Host machine and the Enterprise Designer machine.	Set the driver properties according to driver vendor's instruction. For example: **setDefTdpName#DBSQL##setWorkspace#Navigator##**
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**.
DataSource name	Specifies the name of the **XADataSource** or **ConnectionPoolDataSource** implementation, to which the DataSource object delegates behind the scenes when there is connection pooling or distributed transaction management being done.	The name of the **XADataSource** or **ConnectionPoolDataSource** implementation. This property is Optional. In most cases, leave this box empty.
MinPoolSize	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 you set depends on the transaction volume and response time of the application. 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 **0**.

Table 9 Outbound non-Transactional eWay —JDBC Connector Settings (Continued)

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. If the pool size is too big, you may end up with too many connections with the database. The pool size depends on the transaction volume and response time.	A valid numeric value. The default is **10**.
MaxIdleTime	Specifies 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

Details for the Connection Retry Settings used by the external database are detailed in Table 10.

Table 10 Outbound non-Transactional eWay—Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection upon failure to acquire one.	A valid numeric value. The default is **0**.
ConnectionRetryInterval	Specifies the milliseconds of pause before each attempt to reaccess 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 10 times in 5 seconds apart when the Connection Retries is 10 and the Connection Retry Interval is 5000	A valid numeric value. The default is **1000**.

3.5.4 Outbound XA JDBC eWay Properties

The Outbound XA JDBC eWay includes the following configuration sections:

- JDBC Connector Settings

- Connection Retry Settings

JDBC Connector Settings

Details for the JDBC Connector Settings used by the external database are detailed in Table 11.

Table 11 Outbound XA JDBC eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	A valid string. The configured default is **JDBC XA Datasource**.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the **ConnectionPoolDataSource** or **XADataSource** interface. Change as needed for your driver.	A valid class name. For example: **com.attunity.jdbc.NvXADataSource**
ClassNamefor OtherInterfaces	For Sun Java Composite Application Platform Suite components that do not use **ConnectionPoolDataSource**, use this entry to enter the name of the driver class. This class will not be used by the eWay.	A valid class name.
ServerName	This setting 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.
DatabaseName	Specifies the name of the database instance.	Any valid string.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties	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. You must ensure that the driver is installed on both the Logical Host machine and the Enterprise Designer machine.	Set the driver properties according to driver vendor's instruction. For example: **setDefTdpName#DBSQL##setWorkspace#Navigator##**
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**.

Table 11 Outbound XA JDBC eWay—JDBC Connector Settings (Continued)

Name	Description	Required Value
DataSource name	Specifies the name of the **XADataSource** implementation, to which the DataSource object delegates behind the scenes when there is connection pooling or distributed transaction management being done.	The name of the **XADataSource** implementation. This property is Optional. In most cases, leave this box empty.
MinPoolSize	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 you set depends on the transaction volume and response time of the application. 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 **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. If the pool size is too big, you may end up with too many connections with the database. The pool size depends on the transaction volume and response time.	A valid numeric value. The default is **10**.
MaxIdleTime	Specifies 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

Details for the Connection Retry Settings used by the external database are detailed in Table 12.

Table 12 Outbound XA JDBC eWay—Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection upon failure to acquire one.	A valid numeric value. The default is **0**.

Table 12 Outbound XA JDBC eWay—Connection Retry Settings (Continued)

Name	Description	Required Value
ConnectionRetryInterval	Specifies the milliseconds of pause before each attempt to reaccess 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 10 times in 5 seconds apart when the Connection Retries is 10 and the Connection Retry Interval is 5000	A valid numeric value. The default is **1000**.

Chapter 4

Using the JDBC/ODBC eWay Database Wizard

This chapter describes how to use the JDBC eWay Database Wizard to build OTDs.

What's In This Chapter:

- **About the Database OTD Wizard** on page 40
- **Creating a New JDBC OTD** on page 40
- **Steps to Edit an Existing JDBC OTD** on page 55

4.1 About 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 OTD Wizard allows the addition and removal of columns/nodes in an OTD. Nodes with the same name and type as existing nodes are allowed by the wizard, but should not be created, and will result in generic code generation errors upon activation of the OTD.

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

4.2 Creating a New JDBC OTD

The following steps are required to create a new OTD for the JDBC Adapter.

- **Select Wizard Type** on page 41

- **Connect To Database** on page 41
- **Select Database Objects** on page 42
- **Select Tables/Views/Aliases** on page 43
- **Select Procedures** on page 47
- **Add Prepared Statement** on page 51
- **Specify the OTD Name** on page 54
- **Review Selections** on page 54

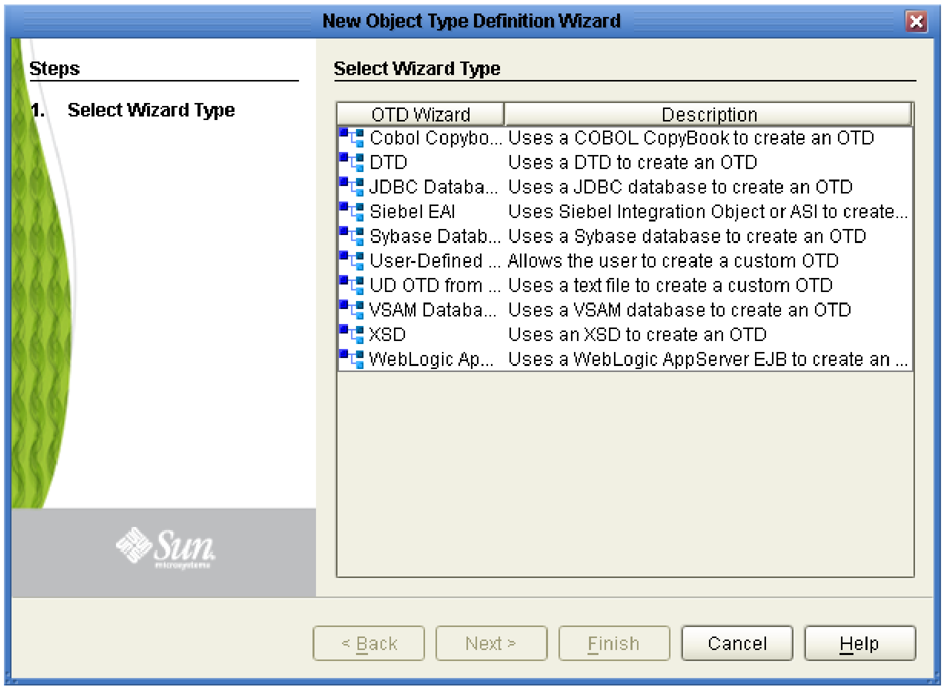
4.2.1 Select Wizard Type

Select the type of wizard required to build an OTD in the New Object Type Definition Wizard.

Steps Required to Select the JDBC Database OTD Wizard Include:

On the Project Explorer tree, right click the Project and select **New > Object Type Definition** from the shortcut menu. The **Select Wizard Type** page appears, displaying the available **OTD** wizards. See Figure 10.

Figure 10 OTD Wizard Selection



OTD Wizard	Description
Cobol Copybo...	Uses a COBOL CopyBook to create an OTD
DTD	Uses a DTD to create an OTD
JDBC Databa...	Uses a JDBC database to create an OTD
Siebel EAI	Uses Siebel Integration Object or ASI to create...
Sybase Datab...	Uses a Sybase database to create an OTD
User-Defined ...	Allows the user to create a custom OTD
UD OTD from ...	Uses a text file to create a custom OTD
VSAM Databa...	Uses a VSAM database to create an OTD
XSD	Uses an XSD to create an OTD
WebLogic Ap...	Uses a WebLogic AppServer EJB to create an ...

4.2.2 Connect To Database

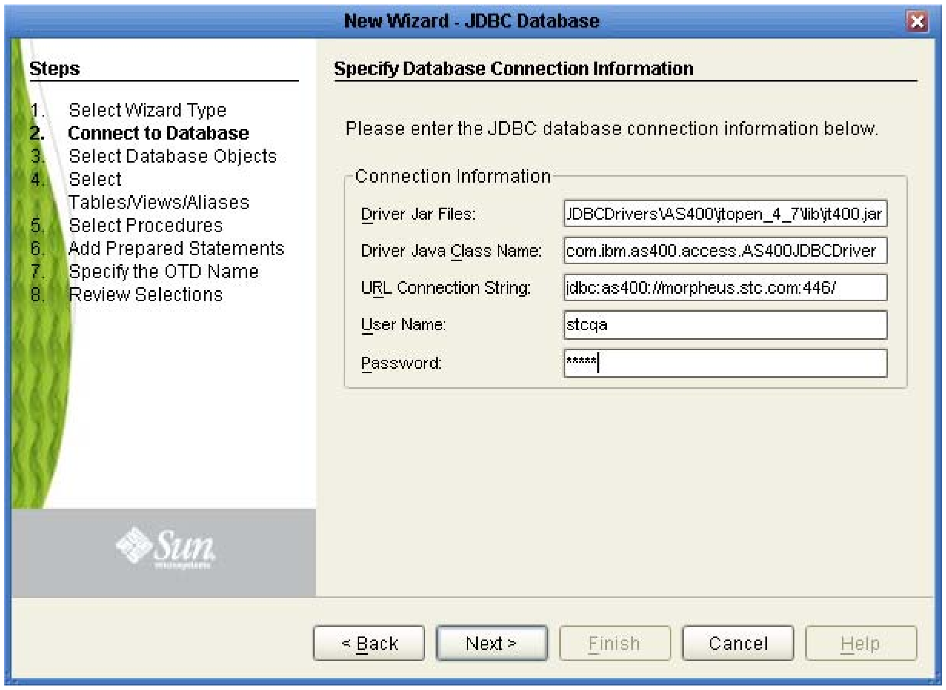
1 From the **New Object Type Definition Wizard** window, select **JDBC Database** and click the **Next** button. The **New Wizard - JDBC Database** window appears.

2 Enter the JDBC database connection information in the Connection Information frame.

Required Database Connection Fields include:

- Driver Jar Files – the location of the driver JAR file.
- Driver Java Class Name – the name of the Driver Manager Class.
- URL Connection String – the URL connection string for the driver.
- User name – a valid JDBC database username.
- Password – a password for the user name noted above.

Figure 11 Database Connection Information



Specify Database Connection Information

Please enter the JDBC database connection information below.

Connection Information

Driver Jar Files:	JDBCDrivers\AS400\jtopen_4_7\lib\jt400.jar
Driver Java Class Name:	com.ibm.as400.access.AS400JDBCDriver
URL Connection String:	jdbc:as400://morpheus.stc.com:446/
User Name:	stcqa
Password:	*****

4.2.3 Select Database Objects

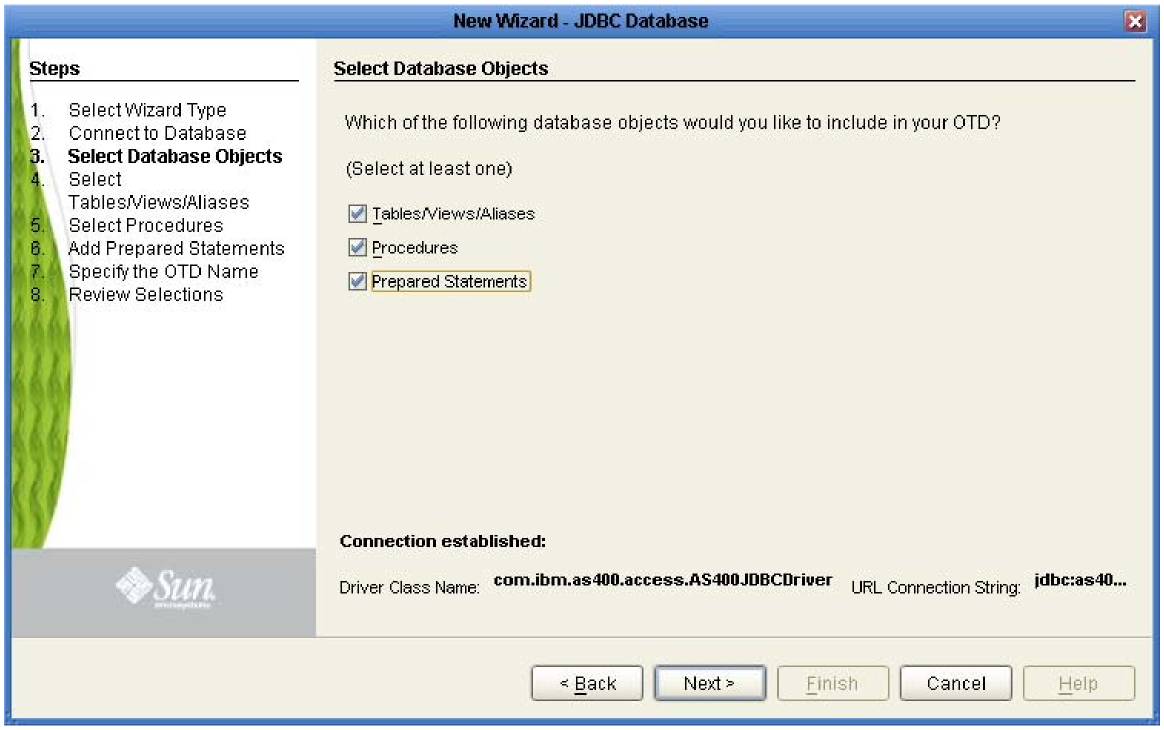
Select the type of JDBC database objects you want included in the OTD.

Steps Required to Select Database Objects Include:

1 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. See Figure 12.

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

Figure 12 Select Database Objects



Select Database Objects

Which of the following database objects would you like to include in your OTD?

(Select at least one)

- [x] Tables/Views/Aliases
- [x] Procedures
- [x] Prepared Statements

Connection established:

Driver Class Name: **com.ibm.as400.access.AS400JDBCDriver** URL Connection String: **jdbc:as40...**

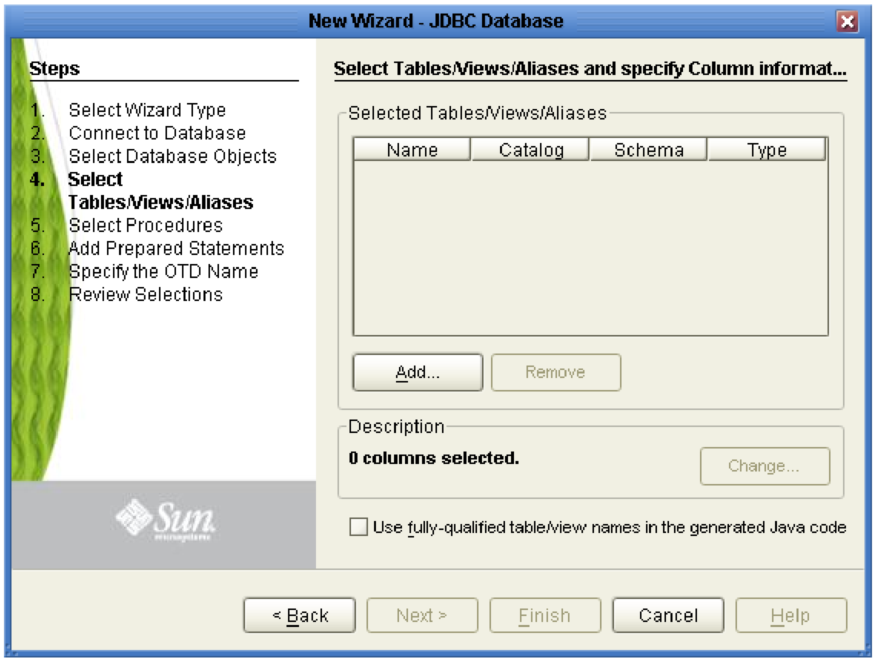
4.2.4 Select Tables/Views/Aliases

Select the types of tables, views, or aliases required in the OTD.

Steps Required to Select Table/Views/Aliases Include:

1 In the **Select Tables/Views/Aliases** window, click **Add**. See Figure 13.

Figure 13 Select Tables/Views/Aliases


New Wizard - JDBC Database
Steps
1. Select Wizard Type
2. Connect to Database
3. Select Database Objects
4. Select Tables/Views/Aliases
5. Select Procedures
6. Add Prepared Statements
7. Specify the OTD Name
8. Review Selections
Select Tables/Views/Aliases and specify Column informat...
Selected Tables/Views/Aliases
Name
Catalog
Schema
Type
Add...
Remove
Description
Change...
< Back
Next >
Finish
Cancel
Help

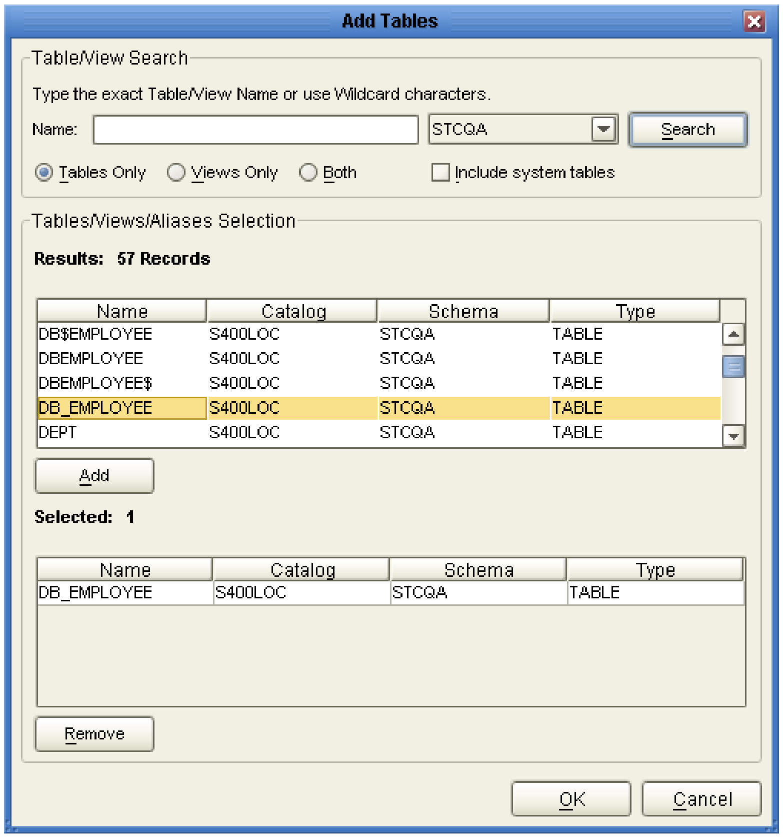
0 columns selected.

☐ Use fully-qualified table/view names in the generated Java code

2 In the **Add Tables** window, select if your selection criteria will include table data, view only data, both, and/or system tables.

3 From the **Table/View Name** drop down list, select the location of your database table and click **Search**. See Figure 14.

Figure 14 Database Wizard - All Schemes



Results: 57 Records

Name	Catalog	Schema	Type
DB$EMPLOYEE	S400LOC	STCQA	TABLE
DBEMPLOYEE	S400LOC	STCQA	TABLE
DBEMPLOYEE$	S400LOC	STCQA	TABLE
DB_EMPLOYEE	S400LOC	STCQA	TABLE
DEPT	S400LOC	STCQA	TABLE

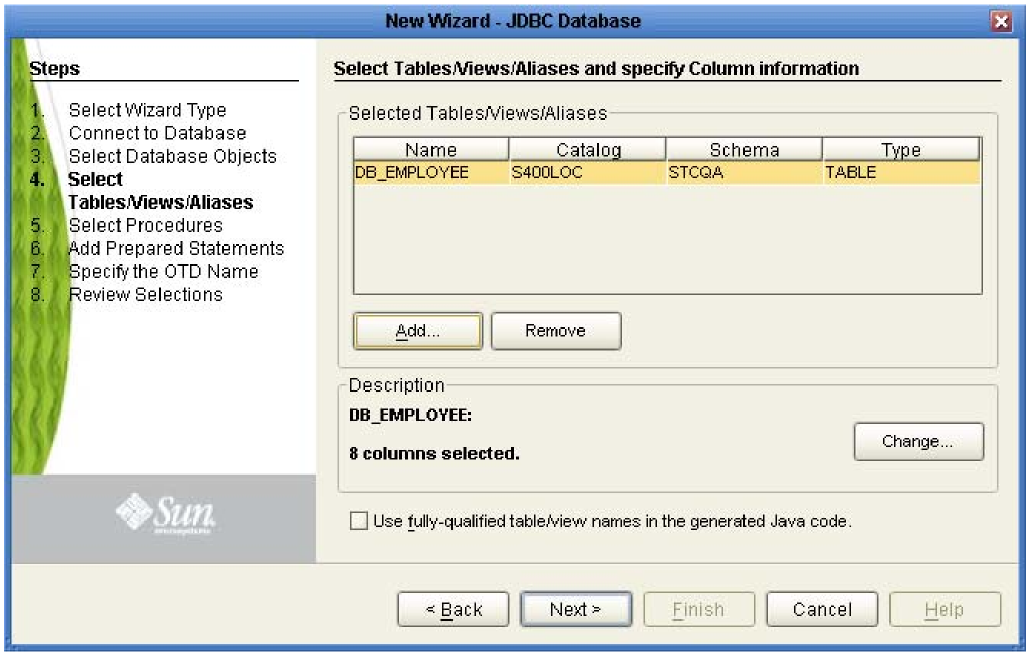
Selected: 1

Name	Catalog	Schema	Type
DB_EMPLOYEE	S400LOC	STCQA	TABLE

4 Select the table of choice and click **OK**.

The table selected is added to the **Selected Tables/Views/Aliases** section. See Figure 15.

Figure 15 Selected Tables/Views/Aliases window with a table selected


New Wizard - JDBC Database
Steps
1. Select Wizard Type
2. Connect to Database
3. Select Database Objects
4. Select Tables/Views/Aliases
5. Select Procedures
6. Add Prepared Statements
7. Specify the OTD Name
8. Review Selections
Select Tables/Views/Aliases and specify Column information
Selected Tables/Views/Aliases
Add...
Remove
Description
Change...
< Back
Next >
Finish
Cancel
Help

Name	Catalog	Schema	Type
DB_EMPLOYEE	S400LOC	STCQA	TABLE

DB_EMPLOYEE:

8 columns selected.

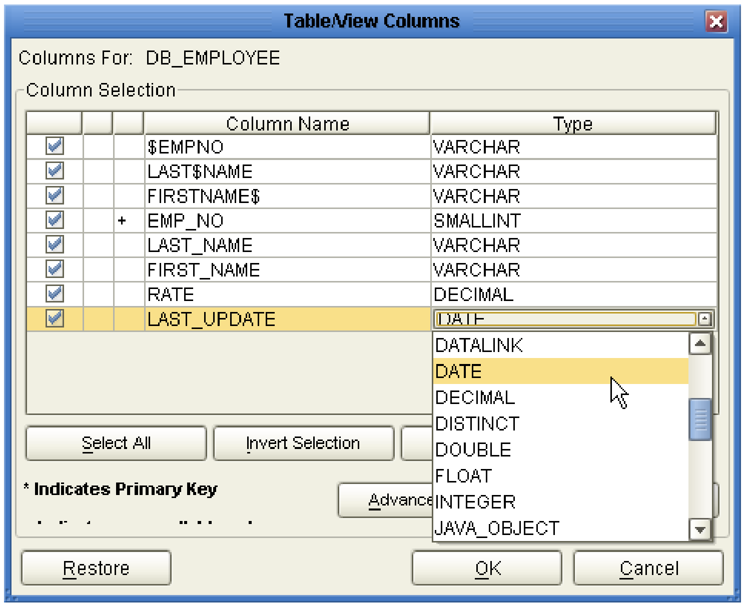
☐ Use fully-qualified table/view names in the generated Java code.

5 In the **Selected Tables/Views/Aliases** section, 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 In the **Table/View Columns** window, you can select or deselect your table columns. You can also change the data type for each table by highlighting the data type and selecting a different one from the drop down list. If you would like to change any of the tables columns, click **Change**. See Figure 16.

The data type is usually listed as **Other** when the driver cannot detect the data type. In these situations we recommend changing the data type to one that is more appropriate for the type of column data.

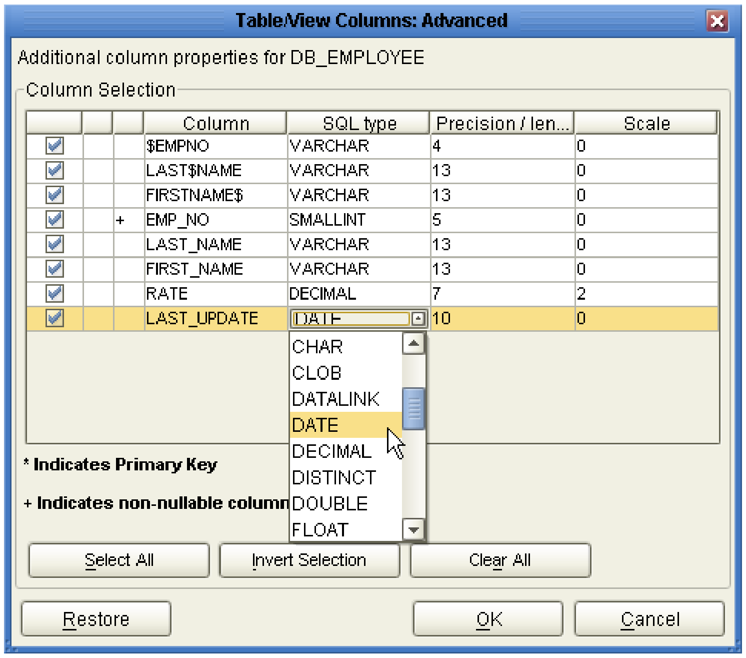
Figure 16 Table/View Columns



			Column Name	Type
✓			$EMPNO	VARCHAR
✓			LAST$NAME	VARCHAR
✓			FIRSTNAME$	VARCHAR
✓		+	EMP_NO	SMALLINT
✓			LAST_NAME	VARCHAR
✓			FIRST_NAME	VARCHAR
✓			RATE	DECIMAL
✓			LAST_UPDATE	DATE

7 Click **Advanced** to change the data type, percision/length, or scale. Once you have finished your table choices, click **OK**. In general, you will not need to make any changes. See Figure 17.

Figure 17 Table/View Columns — Advanced


Table/View Columns: Advanced
Additional column properties for DB_EMPLOYEE
Column Selection

			Column	SQL type	Precision / len...	Scale
✓			$EMPNO	VARCHAR	4	0
✓			LAST$NAME	VARCHAR	13	0
✓			FIRSTNAME$	VARCHAR	13	0
✓		+	EMP_NO	SMALLINT	5	0
✓			LAST_NAME	VARCHAR	13	0
✓			FIRST_NAME	VARCHAR	13	0
✓			RATE	DECIMAL	7	2
✓			LAST_UPDATE	DATE	10	0

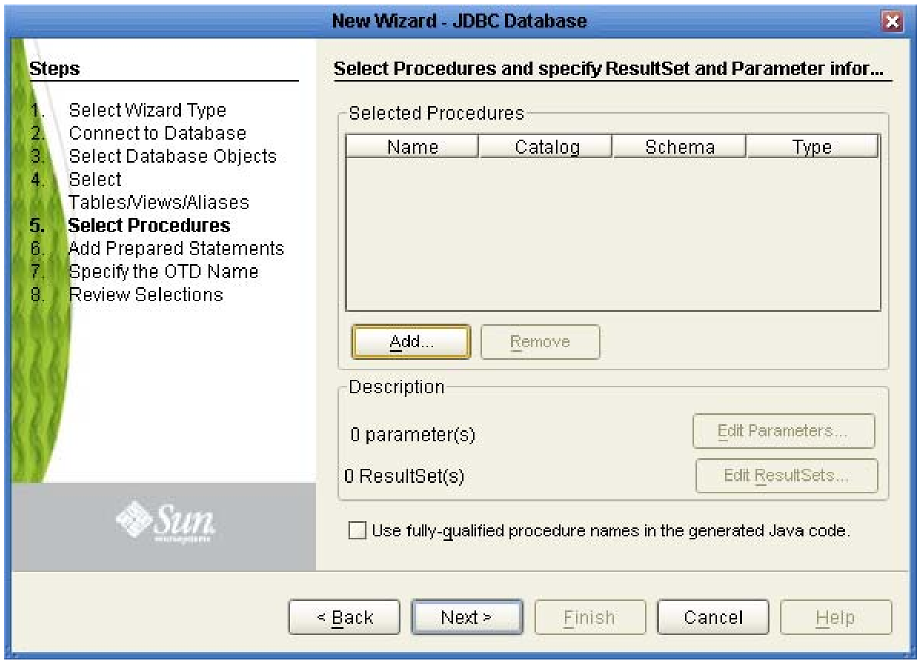
4.2.5 Select Procedures

Select the type of stored procedures required in your OTD.

Steps Required to Select Stored Procedures Include:

1 On the **Select Procedures and specify Resultset and Parameter Information** window, click **Add.**

Figure 18 Select Procedures and specify Resultset and Parameter Information



New Wizard - JDBC Database

Steps

1. Select Wizard Type
2. Connect to Database
3. Select Database Objects
4. Select Tables/Views/Aliases
5. **Select Procedures**
6. Add Prepared Statements
7. Specify the OTD Name
8. Review Selections

Select Procedures and specify ResultSet and Parameter infor...

Selected Procedures

Name	Catalog	Schema	Type

Description

0 parameter(s)

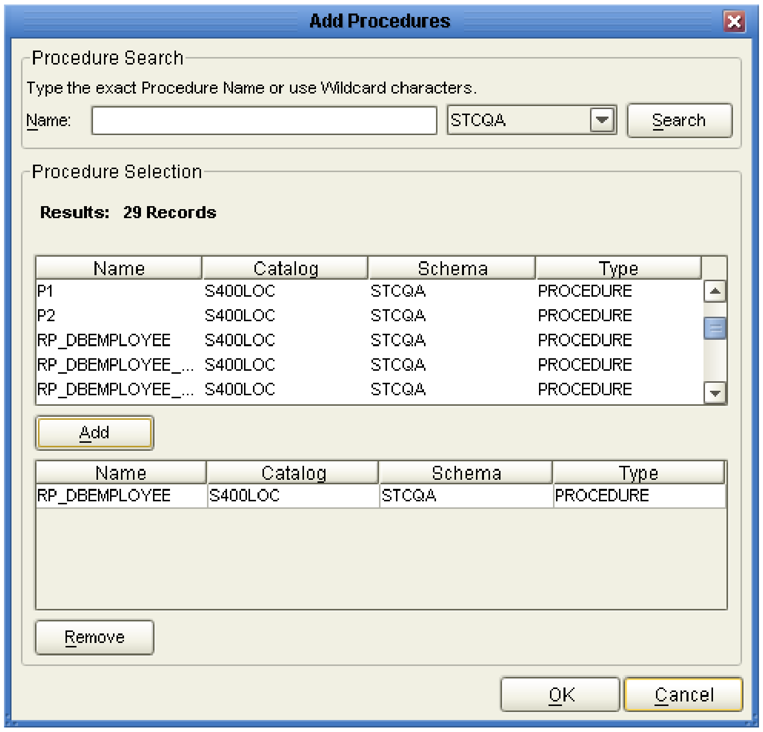
0 ResultSet(s)

☐ Use fully-qualified procedure names in the generated Java code.

2 On the **Select Procedures** window, enter the name of a Procedure or select a schema from the drop down list. Click **Search**. Wildcard characters can also be used.

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

Figure 19 Add Procedures



Type the exact Procedure Name or use Wildcard characters.

Name: STCQA

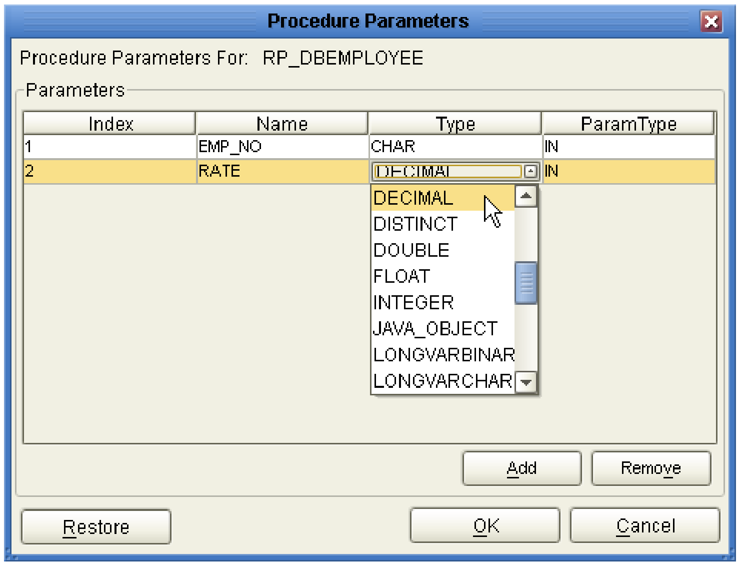
Results: 29 Records

Name	Catalog	Schema	Type
P1	S400LOC	STCQA	PROCEDURE
P2	S400LOC	STCQA	PROCEDURE
RP_DBEMPLOYEE	S400LOC	STCQA	PROCEDURE
RP_DBEMPLOYEE_...	S400LOC	STCQA	PROCEDURE
RP_DBEMPLOYEE_...	S400LOC	STCQA	PROCEDURE

Name	Catalog	Schema	Type
RP_DBEMPLOYEE	S400LOC	STCQA	PROCEDURE

4 On the **Select Procedures and specify Resultset and Parameter Information** window click **Edit Parameters** to make any changes to the selected Procedure. See Figure 20.

Figure 20 Procedure Parameters



Procedure Parameters For: RP_DBEMPLOYEE

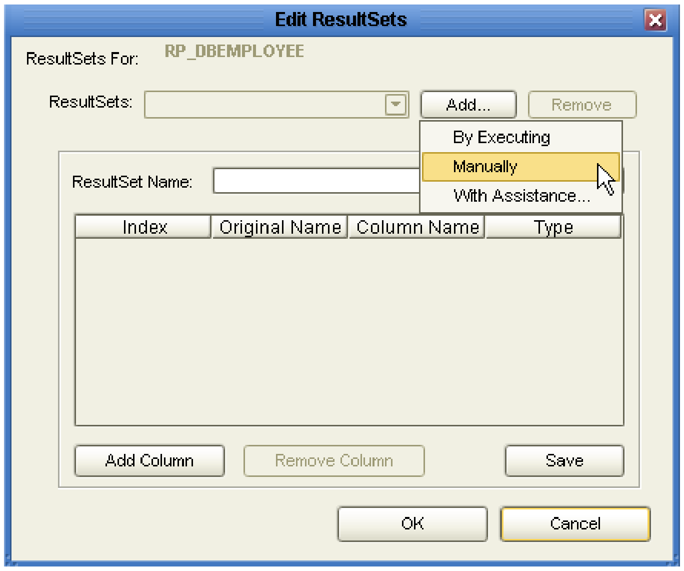
Index	Name	Type	ParamType
1	EMP_NO	CHAR	IN
2	RATE	DECIMAL	IN

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

6 To select how you would like the OTD to generate the nodes for the Resultset click **Edit Resultsets**.

7 Click Add to add the type of Resultset node you would like to generate.

Figure 21 Edit Resultset



The DBWizard provides three different ways to generate the ResultSet nodes of a Stored Procedure. They are "**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.

Note: *Not all drivers and databases support stored procedures or stored procedures with ResultSets. You may contact the driver's vendor for more information.*

4.2.6 Add Prepared Statement

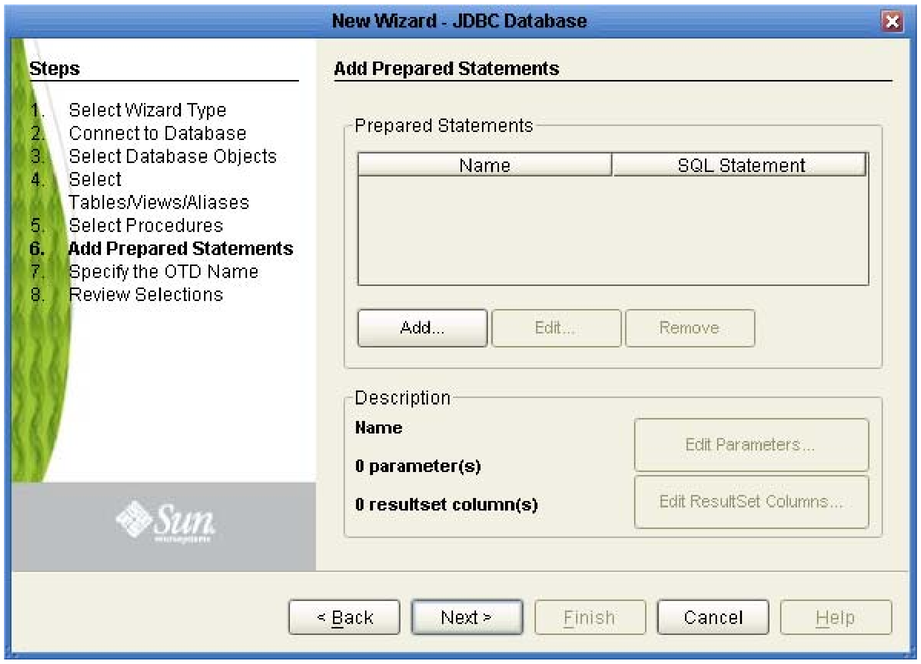
Add a Prepared Statement object to your OTD.

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.*

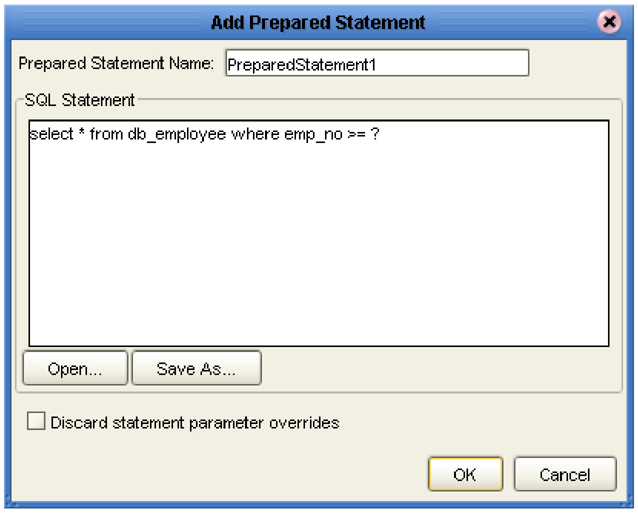
1 On the **Add Prepared Statements** window, click **Add**.

Figure 22 Prepared Statement



2 Enter the name of a Prepared Statement or create a SQL statement by clicking in the SQL Statement window. When finished creating the statement, click **Save As** giving the statement a name. This name will appear as a node in the OTD. Click **OK**. See Figure 23.

Figure 23 Prepared SQL Statement



Prepared Statement Name: PreparedStatement1

SQL Statement

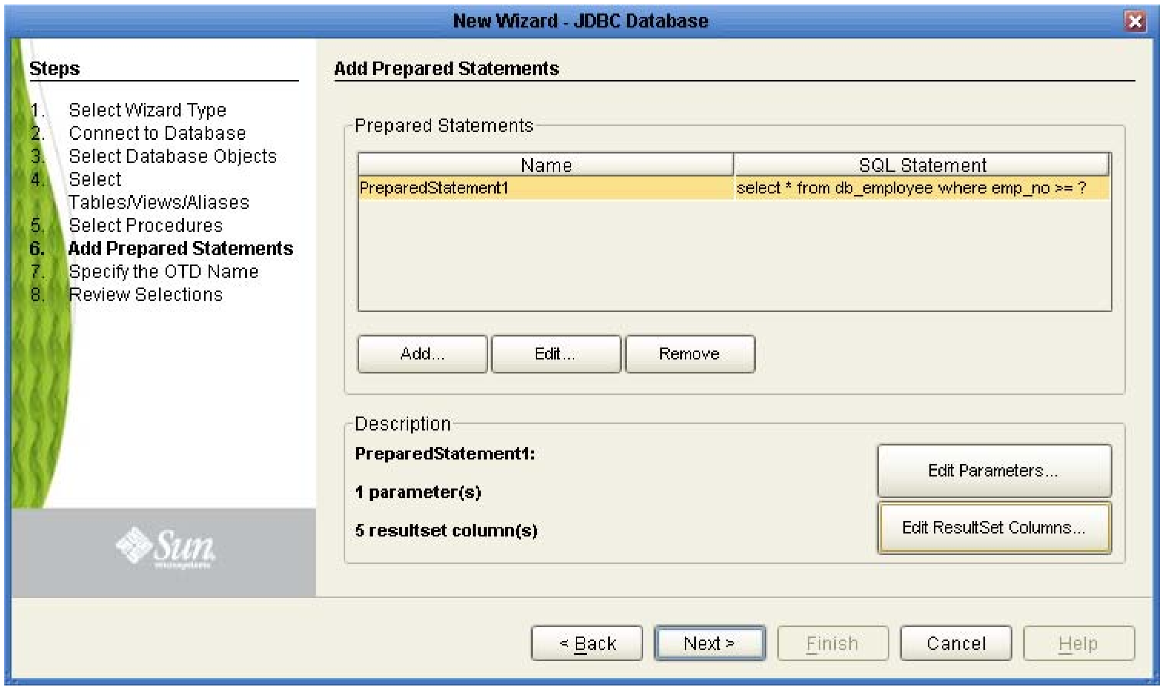
```
select * from db_employee where emp_no >= ?
```

☐ Discard statement parameter overrides

3 On the **Add Prepared Statement** window, 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 it. Click **OK**. Figure 24.

Figure 24 Edit the Prepared Statement Parameters



Name	SQL Statement
PreparedStatement1	select * from db_employee where emp_no >= ?

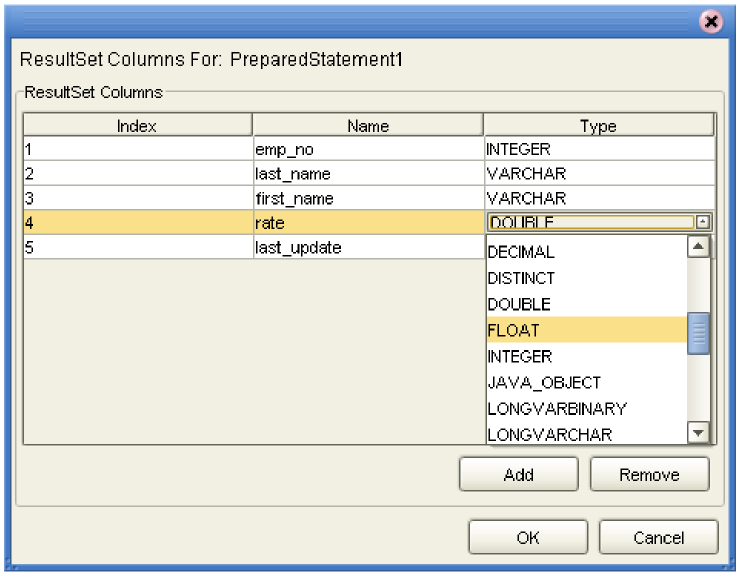
PreparedStatement1:

1 parameter(s)

5 resultset column(s)

5 To edit Resultset Columns, click **Edit Resultset Columns**. The ResultSet Columns window appears. See Figure 25.

Figure 25 ResultSet Columns



Index	Name	Type
1	emp_no	INTEGER
2	last_name	VARCHAR
3	first_name	VARCHAR
4	rate	DOUBLE
5	last_update	

6 Click Add to add a new ResultSet column. Both the Name and Type are editable.

7 Click OK to return to the Add Prepared Statements window.

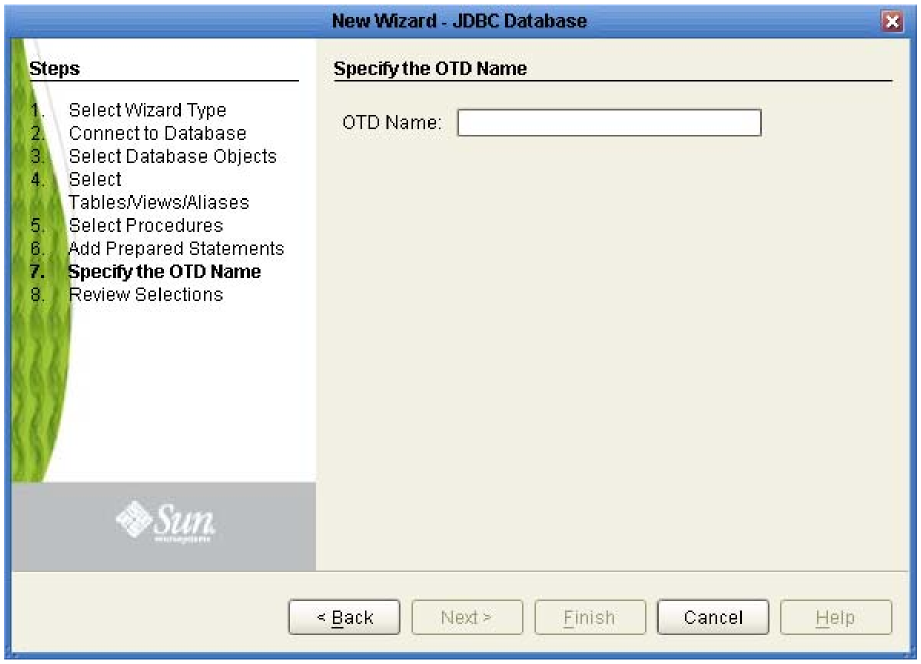
4.2.7 Specify the OTD Name

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 26.

Figure 26 Naming an OTD


New Wizard - JDBC Database

Steps

1. Select Wizard Type
2. Connect to Database
3. Select Database Objects
4. Select Tables/Views/Aliases
5. Select Procedures
6. Add Prepared Statements
7. **Specify the OTD Name**
8. Review Selections

Specify the OTD Name

OTD Name:

2 Click **Next**.

4.2.8 Review Selections

Review the selections made for the new OTD.

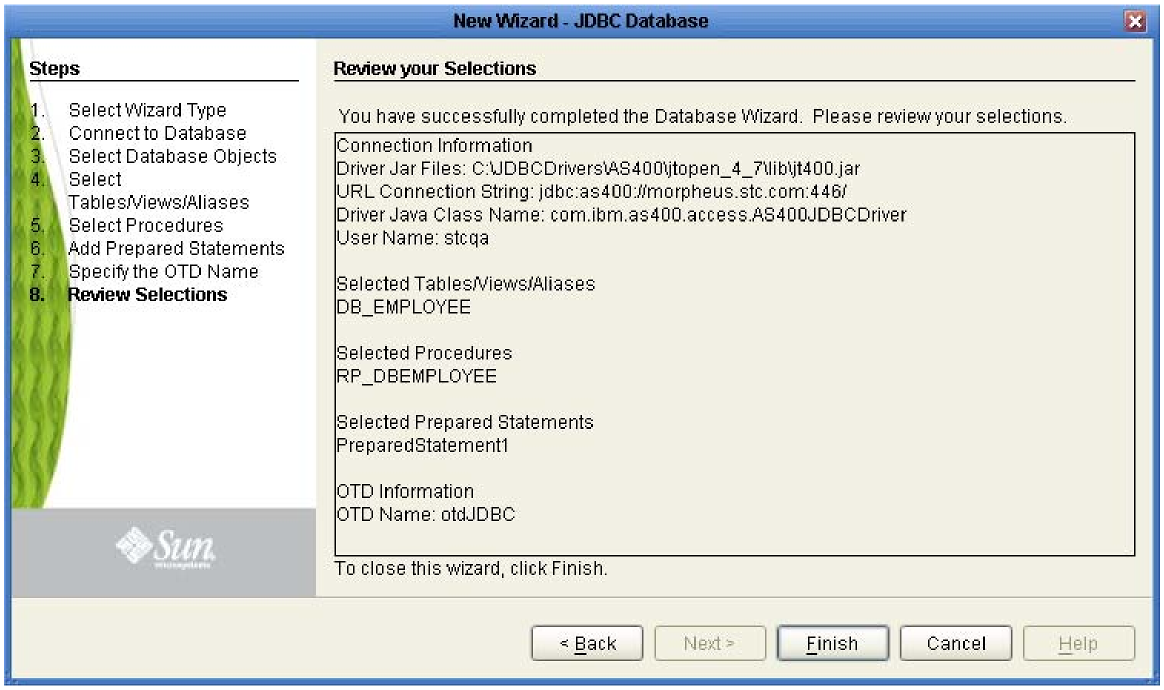
Steps Required to Review Your OTD Selections:

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 27.

The resulting **OTD** appears on the Enterprise Designer’s Project Explorer.

Figure 27 Database Wizard - Summary



You have successfully completed the Database Wizard. Please review your selections.

Connection Information
Driver Jar Files: C:\JDBCDrivers\AS400\jtopen_4_7\lib\jt400.jar
URL Connection String: jdbc:as400://morpheus.stc.com:446/
Driver Java Class Name: com.ibm.as400.access.AS400JDBCDriver
User Name: stcqa

Selected Tables/Views/Aliases
DB_EMPLOYEE

Selected Procedures
RP_DBEMPLOYEE

Selected Prepared Statements
PreparedStatement1

OTD Information
OTD Name: otdJDBC

To close this wizard, click Finish.

4.3 Steps to Edit an Existing JDBC OTD

You can edit any database OTD you create directly from the Enterprise Designer Project Explorer.

Steps to Edit the OTD from the Enterprise Designer Include:

1. Unlock the OTD. To do this, right-click the OTD in the Project Explorer and select **Version Control** > **Check Out** from the menu.

   The Version Control - Check Out window appears.

2. Select the OTD you want to check out, then click **Check Out**.
3. From the Project Explorer, right-click the OTD again and select **Edit** from the menu.

   The JDBC Database Connection Information wizard appears.

4. Enter the connection information as described in **"Connect To Database" on page 41**, and click **Next**.
5. Step through each of the wizard steps and click **Finish** to save your changes.

Note: *You must verify during project activation or at runtime that no errors are generated after editing an OTD. Errors could occur if you delete a database object that is included in a Collaboration.*

Chapter 5

Using JDBC/ODBC Operations

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

What's in This Chapter

- **JDBC eWay Database Operations (BPEL)** on page 56
- **JDBC eWay Database Operations (JCD)** on page 58

5.1 JDBC eWay Database Operations (BPEL)

Within a BPEL business process, the JDBC eWay uses BPEL Activities to perform basic outbound database operations, including:

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

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

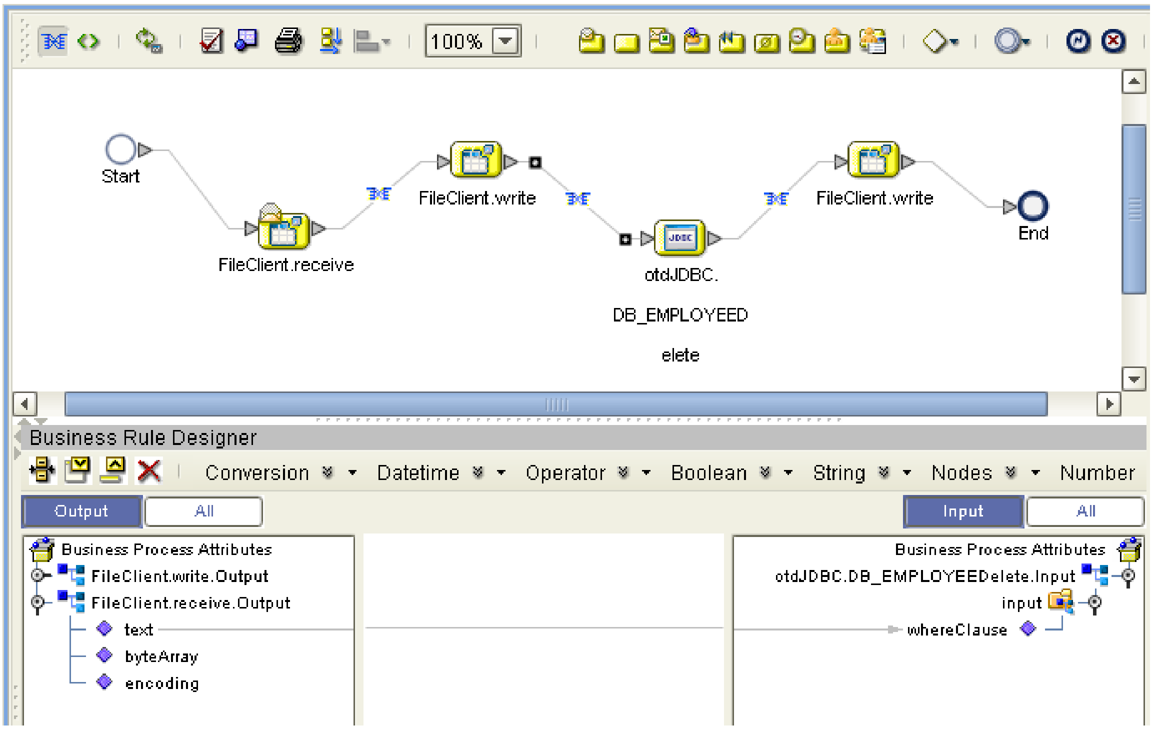
The ability to perform any of the above methods using a table OTD may not be possible with all third-party drivers. You have to use a Prepared Statement to perform such an operation. Check with the respective driver's vendor for further information. This feature is known as Updatable ResultSet.

5.1.1 Activity Input and Output

The Sun SeeBeyond Enterprise Designer – 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 **otdJDBC.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 13 JDBC.ODBC Operations

eInsight Operations	Activity Input	Activity Output
SelectAll	where() clause (optional)	Returns all rows that fit the condition of the where() clause.

Table 13 JDBC.ODBC Operations (Continued)

eInsight Operations	Activity Input	Activity Output
SelectMultiple	number of rows where() clause (optional)	Returns the number of rows specified that fit the condition of the where() clause, and the number of rows to be returned. For example: If the number of rows that meet the condition are 5 and the number of available rows are 10, then only 5 rows will be returned. Alternately, if the number of rows that meet the condition are 20, but if the number of available rows are 10, then only 10 rows are returned.
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.

5.2 JDBC 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*)

The ability to perform any of the above methods using a table OTD may not be possible with all third-party drivers. You have to use a Prepared Statement to perform such an operation. Check with the respective driver's vendor for further information. This feature is known as Updatable ResultSet.

Note: *Refer to the Javadoc for a full description of methods included in the JDBC 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. The ability to update via a ResultSet is called "Updatable ResultSet", which is a feature supported by this eWay.

By default, the Table OTD has UpdatableConcurrency and ScrollTypeForwardOnly. Normally you do not have to change the default setting.

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

For example:

```
package prjJDBC_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.otdInputDTD_1394195520.DBemployees otdInputDTD_DBemployees_1,
otdJDBC.OtdJDBCOTD otdJDBC_1, dtd.otdOutputDTD882991309.DBemployee
otdOutputDTD_DBemployee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Selecting record(s) from db_employee
table via table select .." );
        FileClient_1.write();
        otdJDBC_1.getDB_EMPLOYEE().select( input.getText() );
        while (otdJDBC_1.getDB_EMPLOYEE().next()) {
            otdOutputDTD_DBemployee_1.setEmpNo(
typeConverter.shortToString( otdJDBC_1.getDB_EMPLOYEE().getEMP_NO(),
"#", false, "" ) );
            otdOutputDTD_DBemployee_1.setLastname(
otdJDBC_1.getDB_EMPLOYEE().getLAST_NAME() );
            otdOutputDTD_DBemployee_1.setFirstname(
otdJDBC_1.getDB_EMPLOYEE().getFIRST_NAME() );
            otdOutputDTD_DBemployee_1.setRate(
otdJDBC_1.getDB_EMPLOYEE().getRATE().toString() );
            otdOutputDTD_DBemployee_1.setLastDate(
typeConverter.dateToString(
otdJDBC_1.getDB_EMPLOYEE().getLAST_UPDATE(), "yyyy-MM-dd hh:mm:ss",
false, "" ) );
            FileClient_1.setText(
otdOutputDTD_DBemployee_1.marshalToString() );
            FileClient_1.write();
        }
        FileClient_1.setText( "Done table select." );
        FileClient_1.write();
    }

}
```

The Insert Operation

To perform an insert operation on a table:

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

   This example inserts an employee record.

```
package prjJDBC_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,
com.stc.connector.appconn.file.FileApplication FileClient_1,
dtd.otdInputDTD_1394195520.DBemployees otdInputDTD_DBemployees_1,
dtd.otdOutputDTD882991309.DBemployee otdOutputDTD_DBemployee_1,
otdJDBC.OtdJDBCOTD otdJDBC_1 )
        throws Throwable
    {
        FileClient_1.setText( "Inserting records into db_employee
table .." );
        FileClient_1.write();
        otdInputDTD_DBemployees_1.unmarshalFromString(
input.getText() );
        for (int i1 = 0; i1 <
otdInputDTD_DBemployees_1.countX_sequence_A(); i1 += 1) {
            otdJDBC_1.getInsert_Ps().setEmp_no(
typeConverter.stringToShort(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdJDBC_1.getInsert_Ps().setLast_name(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getLastname() );
            otdJDBC_1.getInsert_Ps().setFirst_name(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getFirstname() );
            otdJDBC_1.getInsert_Ps().setRate( new
java.math.BigDecimal( otdInputDTD_DBemployees_1.getX_sequence_A( i1
).getRate() ) );
            otdJDBC_1.getInsert_Ps().setLast_update(
typeConverter.stringToSQLDate(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdJDBC_1.getInsert_Ps().executeUpdate();
        }
        FileClient_1.setText( "Done Insert." );
        FileClient_1.write();
    }

}
```

The Update Operation

To perform an update operation on a table:

1. Execute the **update()** method.
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 prjJDBC_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,
otdJDBC.OtdJDBCOTD otdJDBC_1, dtd.otdOutputDTD882991309.DBemployee
otdOutputDTD_DBemployee_1, dtd.otdInputDTD_1394195520.DBemployees
otdInputDTD_DBemployees_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Update the Rate and Last_update fields
using Prepared Statement.. " );
        FileClient_1.write();
        otdJDBC_1.getUpdate_Ps().setEmp_no(
typeConverter.stringToShort( input.getText(), "#", false, 0 ) );
        otdJDBC_1.getUpdate_Ps().executeUpdate();
        FileClient_1.setText( "Done Update." );
        FileClient_1.write();
    }

}
```

The Delete Operation

To perform a delete operation on a table:

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

In this example DELETE an employee.

```
package prjJDBC_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,
dtd.otdInputDTD_1394195520.DBemployees otdInputDTD_DBemployees_1,
otdJDBC.OtdJDBCOTD otdJDBC_1, dtd.otdOutputDTD882991309.DBemployee
otdOutputDTD_DBemployee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Delete record .." );
        FileClient_1.write();
        otdJDBC_1.getDB_EMPLOYEE().delete( input.getText() );
        FileClient_1.setText( "Done delete." );
        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. Remember that while in the Collaboration Editor you can drag and drop nodes from the OTD into the Collaboration Editor.

Executing Stored Procedures

The OTD represents the Stored Procedure “LookUpGlobal” with two parameters, an inbound parameter (INLOCALID) and an outbound parameter (OUTGLOBALPRODUCTID). These inbound and outbound parameters are generated by the Database Wizard and are represented in the resulting OTD as nodes. Within the Transformation Designer, you can drag values from the input parameters, execute the call, collect data, and drag the values to the output parameters.

Steps for executing the Stored Procedure include:

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

For Stored Procedures that return ResultSets and Update Count, the following methods are provided to manipulate the ResultSet:

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

Many drivers do not support manipulating ResultSets in a Stored Procedure. It is recommended that you use specific eWay Adapters for Oracle, SQL Server, Sybase, DB2, and so forth, to peform such operations.

JDBC 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 PreparedStatementAgent class, simplifies the whole process of determining whether any results, be it 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 **getUpdateCount()** 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.

Frequently, Update Counts information that is returned from a Stored Procedures is insignificant. You should process returned ResultSet information and avoid looping through all of the Update Counts. The following three methods control exactly what information should be returned from a stored procedure call. The **enableResultSetsOnly()** method, added to the PreparedStatement Agent class allows only ResultSets to be returned and thus every **resultsAvailable()** called only returns Boolean true if a ResultSet is available. Likewise, the **enableUpdateCountsOnly()** causes **resultsAvailable()** to return true only if an Update Count is available. The default case of **enableResultsetsAndUpdateCount()** method allows both ResultSets and Update Counts to be returned.

Collaboration usability for a stored procedure ResultSet

The Column data of the ResultSets can be dragged-and-dropped from their nodes to the Business Rules. Below is a code snippet that can be generated by the Collaboration Editor:

```
while (getSPIn().getSpS_multi().resultsAvailable())
{
if (getSPIn().getSpS_multi().getUpdateCount() > 0)
  {
    System.err.println("Updated
"+getSPIn().getSpS_multi().getUpdateCount()+" rows");
  }

  if (getSPIn().getSpS_multi().getNormRS().available())
  {
    while (getSPIn().getSpS_multi().getNormRS().next())
    {
      System.err.println("Customer Id   =
"+getSPIn().getSpS_multi().getNormRS().getCustomerId());
      System.err.println("Customer Name =
"+getSPIn().getSpS_multi().getNormRS().getCustomerName());
      System.err.println();
    }
    System.err.println("===");
  }
  else if (getSPIn().getSpS_multi().getDbEmployee().available())
  {
    while (getSPIn().getSpS_multi().getDbEmployee().next())
    {
      System.err.println("EMPNO    =
"+getSPIn().getSpS_multi().getDbEmployee().getEMPNO());
      System.err.println("ENAME    =
"+getSPIn().getSpS_multi().getDbEmployee().getENAME());
      System.err.println("JOB      =
"+getSPIn().getSpS_multi().getDbEmployee().getJOB());
      System.err.println("MGR      =
"+getSPIn().getSpS_multi().getDbEmployee().getMGR());
      System.err.println("HIREDATE =
"+getSPIn().getSpS_multi().getDbEmployee().getHIREDATE());
      System.err.println("SAL      =
"+getSPIn().getSpS_multi().getDbEmployee().getSAL());
      System.err.println("COMM     =
"+getSPIn().getSpS_multi().getDbEmployee().getCOMM());
      System.err.println("DEPTNO   =
"+getSPIn().getSpS_multi().getDbEmployee().getDEPTNO());
      System.err.println();
    }
    System.err.println("===");
  }
}
```

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

After calling "**resultsAvailable()**", 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, it is recommended that for maximum portability, all of the results in a ResultSet object should be retrieved before OUT parameters are retrieved. Therefore, you should 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. You should not call both methods in your java code. Doing so may result in skipped data from one of the ResultSets when more than one ResultSet is present.
- The methods **available()** and **getResultSet()** can not be used in conjunction with multiple ResultSets being 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 ResultSet (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 assure the indexes match the stored procedure. By doing this, your ResultSet indexes are preserved.
- Generally, getMoreResults does not need to be called. It is needed if you do not want to use our enhanced methods and you want to follow the traditional JDBC calls on your own.

The DBWizard Assistant expects the column names to be in English when creating a ResultSet.

Prepared Statement

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) value(?, ?, ?)
```

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

```
getPrepStatement().getPreparedStatementTest().setAge(23);
getPrepStatement().getPreparedStatementTest().setName('Peter Pan');
getPrepStatement().getPreparedStatementTest().setDeptNo(6);
getPrepStatement().getPreparedStatementTest().executeUpdate();
```

Batch Operations

To achieve better performance, consider using a bulk insert if you have to insert many records. This is the “Add Batch” capability. The only modification required is to include the **addBatch()** method for each SQL operation and then the **executeBatch()** call to submit the batch to the database server. Batch operations apply only to Prepared Statements.

Not all drivers support batch operations. Check with the respective driver’s vendor for further information.

```
getPrepStatement().getPreparedStatementTest().setAge(23);
getPrepStatement().getPreparedStatementTest().setName(‘Peter Pan’);
getPrepStatement().getPreparedStatementTest().setDeptNo(6);
getPrepStatement().getPreparedStatementTest().addBatch();

getPrepStatement().getPreparedStatementTest().setAge(45);
getPrepStatement().getPreparedStatementTest().setName(‘Harrison
Ford’);
getPrepStatement().getPreparedStatementTest().setDeptNo(7);
getPrepStatement().getPreparedStatementTest().addBatch();
getPrepStatement().getPreparedStatementTest().executeBatch();
```

Chapter 6

Implementing the JDBC/ODBC eWay Sample Projects

This chapter provides an introduction to the JDBC eWay components, and information on how these components are created and implemented in a Sun Java Composite Application Platform Suite Project. Sample Projects are designed to provide an overview of the basic functionality of the JDBC eWay by identifying how information is passed between eGate and supported external databases.

It is assumed that you understand the basics of creating a Project using the 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

- **About the JDBC eWay Sample Projects** on page 68
- **Running the Sample Projects** on page 71
- **Running the SQL Script** on page 72
- **Importing a Sample Project** on page 72
- **Building and Deploying the prjJDBC_JCD Sample Project** on page 73
- **Building and Deploying the prjJDBC_BPEL Sample Project** on page 94

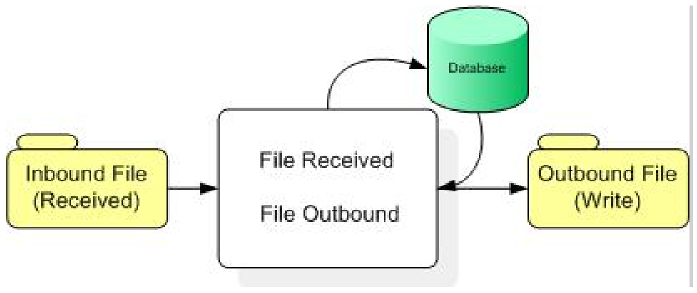
6.1 About the JDBC eWay Sample Projects

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

The **prjJDBC_JCD** sample Project uses input files to pass data into Collaborations. There are four Collaborations that demonstrate the Insert, Update, and Table Select operations, and two Collaboration to demonstrate Prepared Statements (Select and Update). Results are written out to an output file.

The **prjJDBC_BPEL** sample Project uses input files to pass data into business process. There are four business processes that demonstrate the Insert, Update, Delete, and Select operations. Results are written out to an output file.

Figure 29 Database project flow



Both the **prjJDBC_JCD** and **prjJDBC_BPEL** sample Projects demonstrate how to:

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

In addition to the sample Projects, the **JDBC510_SAMPLE_projects.zip** file also includes seven sample input trigger files and nine sample output files, as follows:

Sample input files

- TriggerInsert.in.~in (for JCE projects only)
- TriggerBpPsInsert.in.~in (for BPEL projects only)
- TriggerDelete.in.~in
- TriggerPsUpdate.in.~in (for JCE projects only)
- TriggerBpUpdate.in.~in (for BPEL projects only)
- TriggerPsSelect.in.~in (for JCE projects only)
- TriggerTableSelect.in.~in

Sample output JCD files

- JCE_Insert_output().dat
- JCE_Delete_output().dat
- JCE_PsUpdate_output().dat
- JCE_PsSelect_output().dat
- JCE_TableSelect_output().dat

Sample output BPEL files

- BPEL_PsInsert_output().dat
- BPEL_Delete_output().dat
- BPEL_Update_output().dat

- BPEL_TableSelect_output().dat

6.1.1 Sample Project Data

Data used for the sample Projects are contained within a table called **db_employee**. The table has the following columns:

Table 14 Sample Project Data - db_employee Table

Column Name	Data Type	Data Length
emp_no	INTEGER	10
last_name	VARCHAR	30
first_name	VARCHAR	30
rate	FLOAT	15
last_update	TIMESTAMP	19

6.1.2 Operations Used in the JDBC Sample Projects

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

- Insert
- Update
- Delete
- Select

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 an OTD using JDBC.
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 runtime. 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.*

6.1.3 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.

6.1.4 Sample Projects Drivers

Sample Projects included with this eWay were built using the AS/400 JDBC Toolbox Driver (jt400.jar). You must get this or a different driver from a third-party vendor and install and configure the eWay as per the vendors specification, using the vendor's specific driver jar file and appropriate driver specific parameters such as driver class name, URL connection string, and so forth.

The JAR file for the driver you select must be copied to the following folder:

```
<JavaCAPS51>\logicalhost\is\domains\domain1\lib
```

where <*JavaCAPS51*> is the directory where the Sun Java Composite Application Platform Suite is installed.

When using any driver to build OTDs, be sure to include the absolute path of the locally installed JAR file in the Database Connection Information window. See **Figure 11 on page 42**.

6.2 Running the Sample Projects

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

1 Run the SQL script.

The script 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 the SQL statement **JDBC_sample_script.sql**, that is included in the sample Project. Note that you must use a database tool to run the script.

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 JDBC 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 ResultSet.

6.4 Importing a Sample Project

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

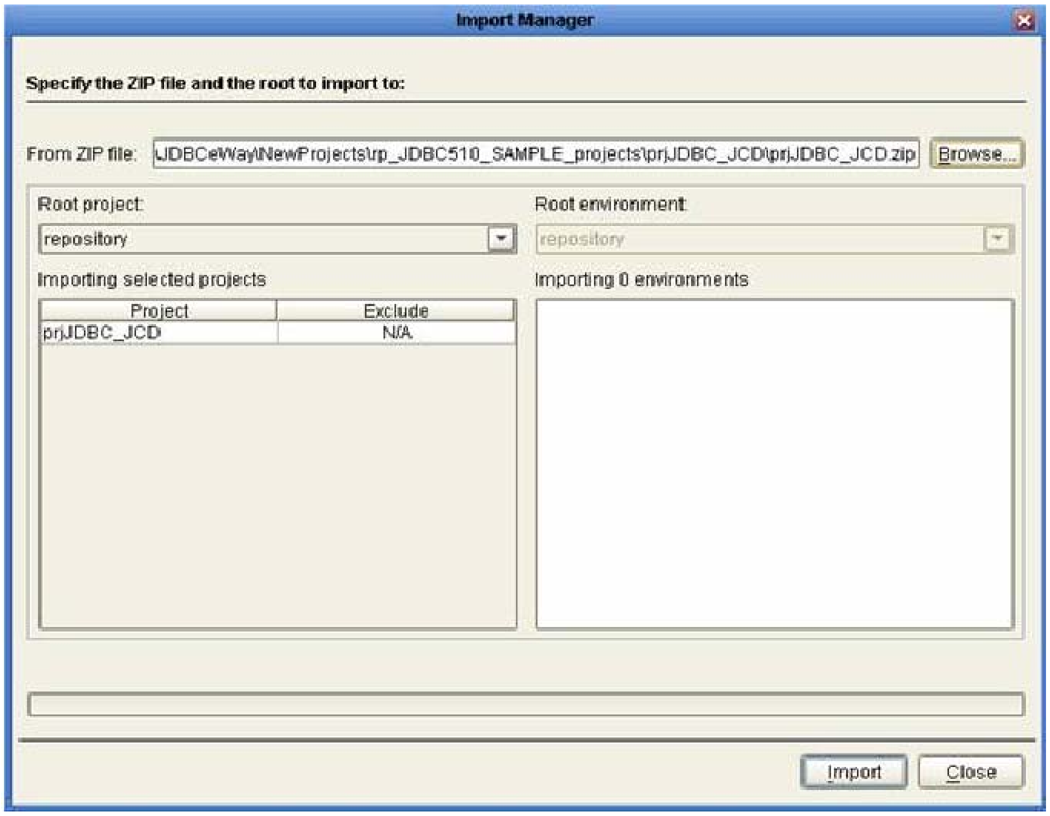
1. Extract the samples from the Sun Java Composite Application Platform Suite Installer to a local file.

   Sample files are uploaded with the eWay's documentation SAR file, and then downloaded from the Installer's **Documentation** tab. The **JDBC_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.

Figure 30 Import Manager Dialog Box



Specify the ZIP file and the root to import to:

From ZIP file: JDBCeWay\NewProjects\rp_JDBC510_SAMPLE_projects\prjJDBC_JCD\prjJDBC_JCD.zip

Root project: repository

Root environment: repository

Importing selected projects

Project	Exclude
prjJDBC_JCD	N/A

Importing 0 environments

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

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

6.5 Building and Deploying the prjJDBC_JCD Sample Project

This section provides step-by-step instructions for manually creating the prjJDBC_JCD sample Project.

Steps required to create the sample project

- **Creating a Project** on page 74
- **Creating the OTDs** on page 74
- **Creating a Connectivity Map** on page 75

- **Creating the Collaboration Definitions (Java)** on page 77
- **Create the Collaboration Business Rules** on page 80
- **Binding the eWay Components** on page 86
- **Creating an Environment** on page 87
- **Configuring the eWays** on page 88
- **Creating the Deployment Profile** on page 91
- **Creating and Starting the Domain** on page 92
- **Building and Deploying the Project** on page 93
- **Running the Sample** on page 93

6.5.1 Creating a Project

The first step is to create a new Project in the 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. Right-click **Project1** and select **Rename** from the shortcut menu. Rename the Project (for this sample, **prjJDBC_JCD**).

6.5.2 Creating the OTDs

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

- JDBC Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a JDBC Database OTD:

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 **JDBC Database OTD Wizard** from the list of OTD Wizards and click **Next**.
3. Enter the connection information for the JDBC database. Connection fields include:
   - Driver Jar Files
   - Driver Java Class Name
   - URL Connection String
   - 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 JDBC 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 value for 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 **otdJDBC**.

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**.

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

6.5.3 Creating a Connectivity Map

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

Steps required to create a new Connectivity Map:

1. From the Project Explorer tree, right-click the new **prjJDBC_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
   - cmPsUpdate

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 **prjJDBC_JCD** sample Project requires the following components:

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

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

Steps required to select a JDBC 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, **JDBC** 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
   - JDBC1 to esJDBCOUT
4. Rename each Connectivity Map Service to match the intended operation, as for example:
   - jcdDelete

- jcdInsert
- jcdPsSelect
- jcdTableSelect
- jcdPsUpdate

6.5.4 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 **prjJDBC_JCD** sample include:

- jcdDelete
- jcdInsert
- jcdPsSelect
- jcdTableSelect
- jcdPsUpdate

jcdDelete Collaboration

Steps required to create the jcdDelete 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 **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 **prjJDBC_JCD** > **otdALL** > **otdJDBC**. The **otdJDBC** OTD is added to the Selected OTDs field.
5. Click the **Up One Level** button twice 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 **prjJDBC_JCD** > **otdALL** > **otdJDBC**. The **otdJDBC** 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 twice 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 **prjJDBC_JCD** > **otdALL** > **otdJDBC**. The **otdJDBC** 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 twice 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 **prjJDBC_JCD** > **otdALL** > **otdJDBC**. The **otdJDBC** 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 twice 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.

jcdPsUpdate Collaboration

Steps required to create the jcdPsUpdate 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 **jcdPsUpdate**) 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 **prjJDBC_JCD > otdALL > otdJDBC**. The **otdJDBC** OTD is added to the Selected OTDs field.

5 Click the **Up One Level** button twice 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 **jcdPsUpdate** Collaboration appears in the right pane of the Enterprise Designer.

6.5.5 Create the Collaboration Business Rules

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

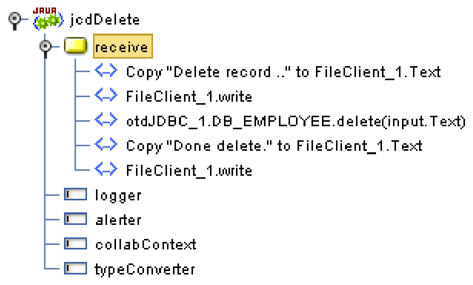
Creating the jcdDelete Business Rules

The **jcdDelete** Collaboration implements the Input Web Service Operation to read the **TriggerDelete.in** file and then delete a record. 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 deleted from the database when the TriggerDelete.in file is empty.*

The **jcdDelete** Collaboration contains the Business Rules displayed in Figure 31.

Figure 31 jcdDelete Business Rules



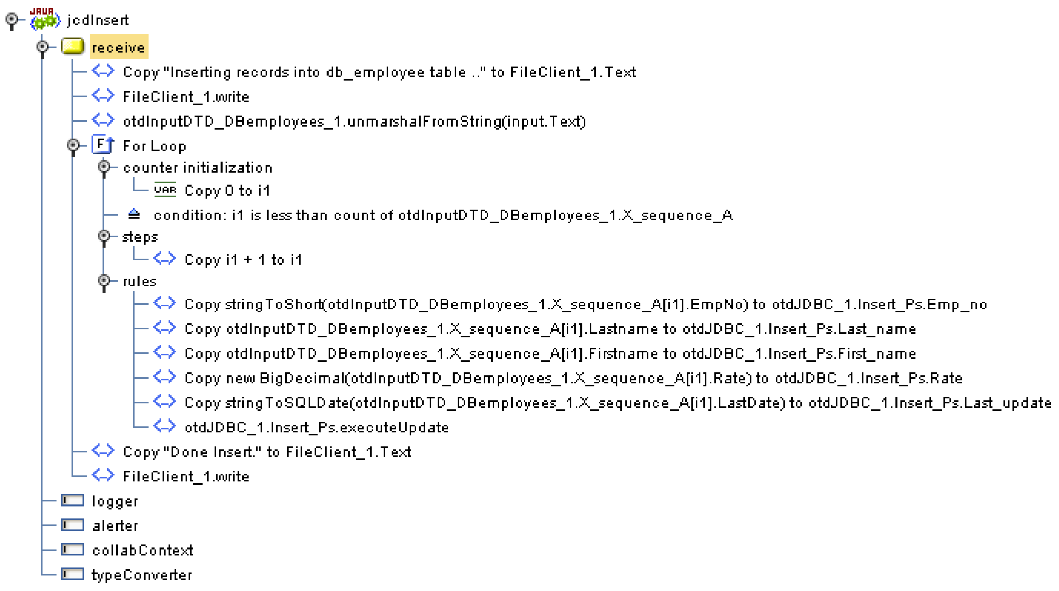
```
jcdDelete
  receive
    Copy "Delete record .." to FileClient_1.Text
    FileClient_1.write
    otdJDBC_1.DB_EMPLOYEE.delete(input.Text)
    Copy "Done delete." to FileClient_1.Text
    FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

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 **otdJDBC** 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 32.

Figure 32 jcdInsert Business Rules



```
jcdInsert
  receive
    Copy "Inserting records into db_employee table .." to FileClient_1.Text
    FileClient_1.write
    otdInputDTD_DBemployees_1.unmarshalFromString(input.Text)
    For Loop
      counter initialization
        Copy 0 to i1
      condition: i1 is less than count of otdInputDTD_DBemployees_1.X_sequence_A
      steps
        Copy i1 + 1 to i1
      rules
        Copy stringToShort(otdInputDTD_DBemployees_1.X_sequence_A[i1].EmpNo) to otdJDBC_1.Insert_Ps.Emp_no
        Copy otdInputDTD_DBemployees_1.X_sequence_A[i1].Lastname to otdJDBC_1.Insert_Ps.Last_name
        Copy otdInputDTD_DBemployees_1.X_sequence_A[i1].Firstname to otdJDBC_1.Insert_Ps.First_name
        Copy new BigDecimal(otdInputDTD_DBemployees_1.X_sequence_A[i1].Rate) to otdJDBC_1.Insert_Ps.Rate
        Copy stringToSQLDate(otdInputDTD_DBemployees_1.X_sequence_A[i1].LastDate) to otdJDBC_1.Insert_Ps.Last_update
        otdJDBC_1.Insert_Ps.executeUpdate
    Copy "Done Insert." to FileClient_1.Text
    FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

Sample code from the jcdInsert Includes:

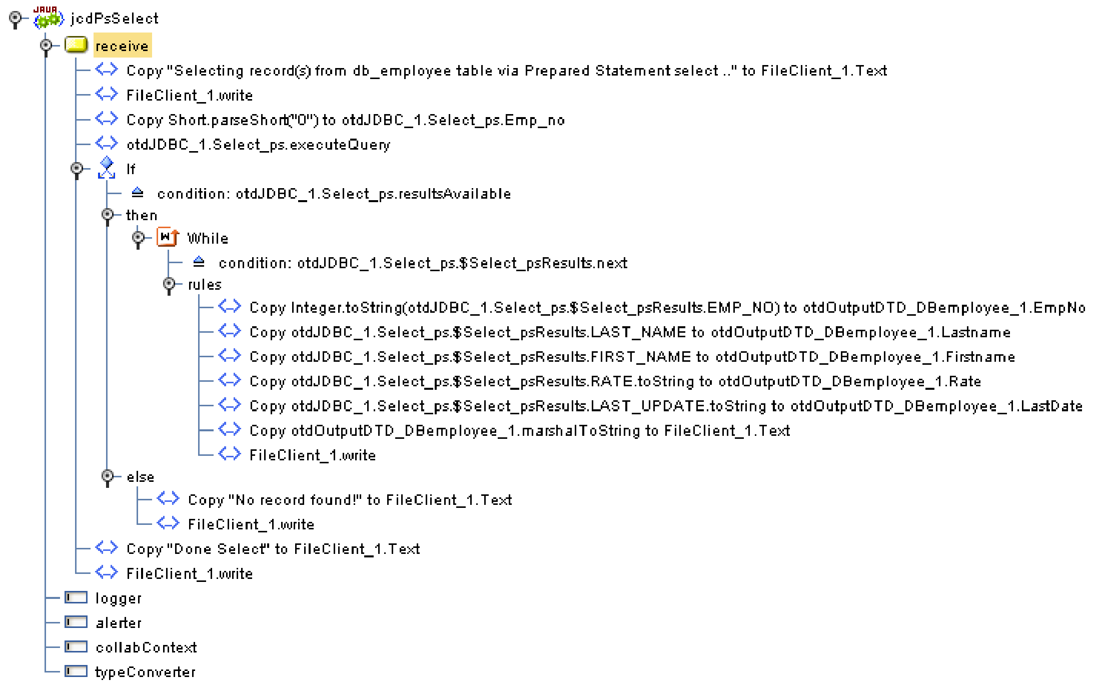
```
package prjJDBC_JCDjcdALL;
public class jcdInsert
{
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;
public void receive( com.stc.connector.appconn.file.FileTextMessage
input, com.stc.connector.appconn.file.FileApplication FileClient_1,
dtd.otdInputDTD_1394195520.DBemployees otdInputDTD_DBemployees_1,
dtd.otdOutputDTD882991309.DBemployee otdOutputDTD_DBemployee_1,
otdJDBC.OtdJDBCOTD otdJDBC_1 )
        throws Throwable
    {

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

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

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

        otdInputDTD_DBemployees_1.unmarshalFromString(
input.getText() );

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

        for (int i1 = 0; i1 <
otdInputDTD_DBemployees_1.countX_sequence_A(); i1 += 1) {
            otdJDBC_1.getInsert_Ps().setEmp_no(
typeConverter.stringToShort(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdJDBC_1.getInsert_Ps().setLast_name(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getLastname() );
            otdJDBC_1.getInsert_Ps().setFirst_name(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getFirstname() );
            otdJDBC_1.getInsert_Ps().setRate( new
java.math.BigDecimal( otdInputDTD_DBemployees_1.getX_sequence_A( i1
).getRate() ) );
            otdJDBC_1.getInsert_Ps().setLast_update(
typeConverter.stringToSQLDate(
otdInputDTD_DBemployees_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdJDBC_1.getInsert_Ps().executeUpdate();

\\ Writes a message to confirm the inserted records.

         }
        FileClient_1.setText( "Done Insert." );
        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 33.

Figure 33 jcdPsSelect Business Rules



```
jcdPsSelect
  receive
    Copy "Selecting record(s) from db_employee table via Prepared Statement select .." to FileClient_1.Text
    FileClient_1.write
    Copy Short.parseShort("0") to otdJDBC_1.Select_ps.Emp_no
    otdJDBC_1.Select_ps.executeQuery
    If
      condition: otdJDBC_1.Select_ps.resultsAvailable
      then
        While
          condition: otdJDBC_1.Select_ps.$Select_psResults.next
          rules
            Copy Integer.toString(otdJDBC_1.Select_ps.$Select_psResults.EMP_NO) to otdOutputDTD_DBemployee_1.EmpNo
            Copy otdJDBC_1.Select_ps.$Select_psResults.LAST_NAME to otdOutputDTD_DBemployee_1.Lastname
            Copy otdJDBC_1.Select_ps.$Select_psResults.FIRST_NAME to otdOutputDTD_DBemployee_1.Firstname
            Copy otdJDBC_1.Select_ps.$Select_psResults.RATE.toString to otdOutputDTD_DBemployee_1.Rate
            Copy otdJDBC_1.Select_ps.$Select_psResults.LAST_UPDATE.toString to otdOutputDTD_DBemployee_1.LastDate
            Copy otdOutputDTD_DBemployee_1.marshalToString to FileClient_1.Text
            FileClient_1.write
      else
        Copy "No record found!" to FileClient_1.Text
        FileClient_1.write
    Copy "Done Select" to FileClient_1.Text
    FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

Sample code from the jcdPsSelect Includes:

```
package prjJDBC_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, otdJDBC.OtdJDBCOTD otdJDBC_1,
dtd.otdOutputDTD882991309.DBemployee otdOutputDTD_DBemployee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

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

        FileClient_1.setText( "Selecting record(s) 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();
       otdJDBC_1.getSelect_ps().setEmp_no( Short.parseShort( "0" ) );
        otdJDBC_1.getSelect_ps().executeQuery();
        if (otdJDBC_1.getSelect_ps().resultsAvailable()) {
            while
(otdJDBC_1.getSelect_ps().get$Select_psResults().next()) {
```

```
                otdOutputDTD_DBemployee_1.setEmpNo( Integer.toString(
otdJDBC_1.getSelect_ps().get$Select_psResults().getEMP_NO() ) );
                otdOutputDTD_DBemployee_1.setLastname(
otdJDBC_1.getSelect_ps().get$Select_psResults().getLAST_NAME() );
                otdOutputDTD_DBemployee_1.setFirstname(
otdJDBC_1.getSelect_ps().get$Select_psResults().getFIRST_NAME() );
                otdOutputDTD_DBemployee_1.setRate(
otdJDBC_1.getSelect_ps().get$Select_psResults().getRATE().toString()
);
                otdOutputDTD_DBemployee_1.setLastDate(
otdJDBC_1.getSelect_ps().get$Select_psResults().getLAST_UPDATE().toSt
ring() );
                FileClient_1.setText(
otdOutputDTD_DBemployee_1.marshalToString() );
                FileClient_1.write();
            }
        } else {
            FileClient_1.setText( "No record found!" );
            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( "Done Select" );
        FileClient_1.write();
    }
}
```

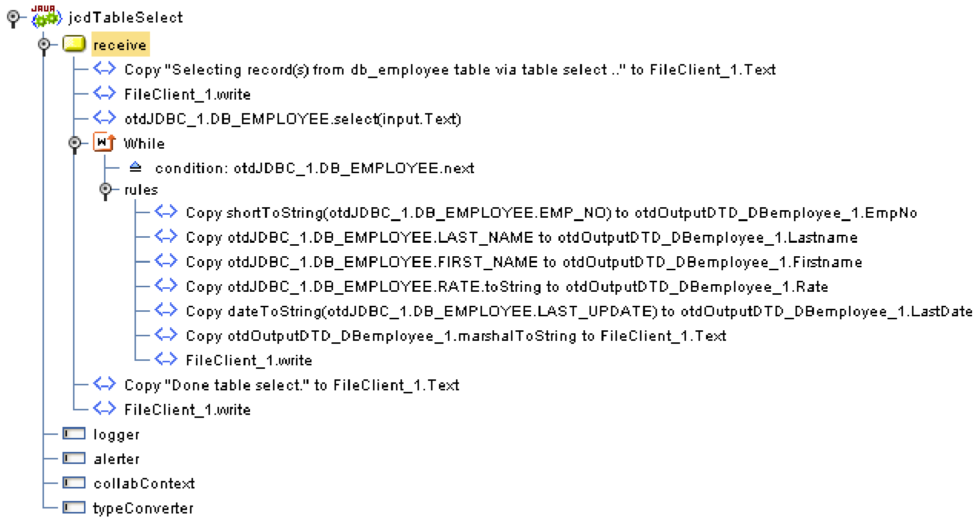
Creating the jcdTableSelect Business Rules

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 a certain criteria. 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 34.

Figure 34 jcdTableSelect Business Rules



```
jcdTableSelect
  receive
    Copy "Selecting record(s) from db_employee table via table select .." to FileClient_1.Text
    FileClient_1.write
    otdJDBC_1.DB_EMPLOYEE.select(input.Text)
    While
      condition: otdJDBC_1.DB_EMPLOYEE.next
      rules
        Copy shortToString(otdJDBC_1.DB_EMPLOYEE.EMP_NO) to otdOutputDTD_DBemployee_1.EmpNo
        Copy otdJDBC_1.DB_EMPLOYEE.LAST_NAME to otdOutputDTD_DBemployee_1.Lastname
        Copy otdJDBC_1.DB_EMPLOYEE.FIRST_NAME to otdOutputDTD_DBemployee_1.Firstname
        Copy otdJDBC_1.DB_EMPLOYEE.RATE.toString to otdOutputDTD_DBemployee_1.Rate
        Copy dateToString(otdJDBC_1.DB_EMPLOYEE.LAST_UPDATE) to otdOutputDTD_DBemployee_1.LastDate
        Copy otdOutputDTD_DBemployee_1.marshalToString to FileClient_1.Text
        FileClient_1.write
    Copy "Done table select." to FileClient_1.Text
    FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

Sample code from the jcdTableSelect Includes:

```
package prjJDBC_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.otdInputDTD_1394195520.DBemployees
otdInputDTD_DBemployees_1, otdJDBC.OtdJDBCOTD otdJDBC_1,
dtd.otdOutputDTD882991309.DBemployee otdOutputDTD_DBemployee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

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

        FileClient_1.setText( "Selecting record(s) 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 a
certain criteria. Checking the next() method ensures all rows are
retrieved in the while loop.

        otdJDBC_1.getDB_EMPLOYEE().select( input.getText() );
        while (otdJDBC_1.getDB_EMPLOYEE().next()) {
            otdOutputDTD_DBemployee_1.setEmpNo(
typeConverter.shortToString( otdJDBC_1.getDB_EMPLOYEE().getEMP_NO(),
"#", false, "" ) );
            otdOutputDTD_DBemployee_1.setLastname(
otdJDBC_1.getDB_EMPLOYEE().getLAST_NAME() );
            otdOutputDTD_DBemployee_1.setFirstname(
otdJDBC_1.getDB_EMPLOYEE().getFIRST_NAME() );
```

```
            otdOutputDTD_DBemployee_1.setRate(
otdJDBC_1.getDB_EMPLOYEE().getRATE().toString() );
            otdOutputDTD_DBemployee_1.setLastDate(
typeConverter.dateToString(
otdJDBC_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_DBemployee_1.marshalToString() );
            FileClient_1.write();
        }

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

        FileClient_1.setText( "Done table select." );
        FileClient_1.write();
    }
}
```

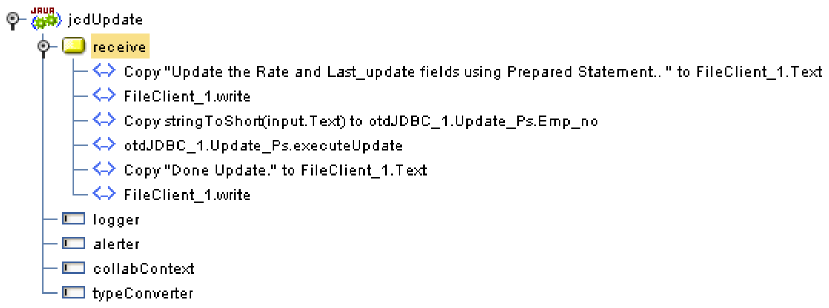
Creating the jcdUpdate Business Rules

The **jcdUpdate** Collaboration implements the Input Web Service Operation to read the **TriggerUpdate.in**. file and then update a particular record. 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 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.*

The **jcdUpdate** Collaboration contains the Business Rules displayed in Figure 35.

Figure 35 jcdTableUpdate



```
jcdUpdate
  receive
    Copy "Update the Rate and Last_update fields using Prepared Statement.." to FileClient_1.Text
    FileClient_1.write
    Copy stringToShort(input.Text) to otdJDBC_1.Update_Ps.Emp_no
    otdJDBC_1.Update_Ps.executeUpdate
    Copy "Done Update." to FileClient_1.Text
    FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

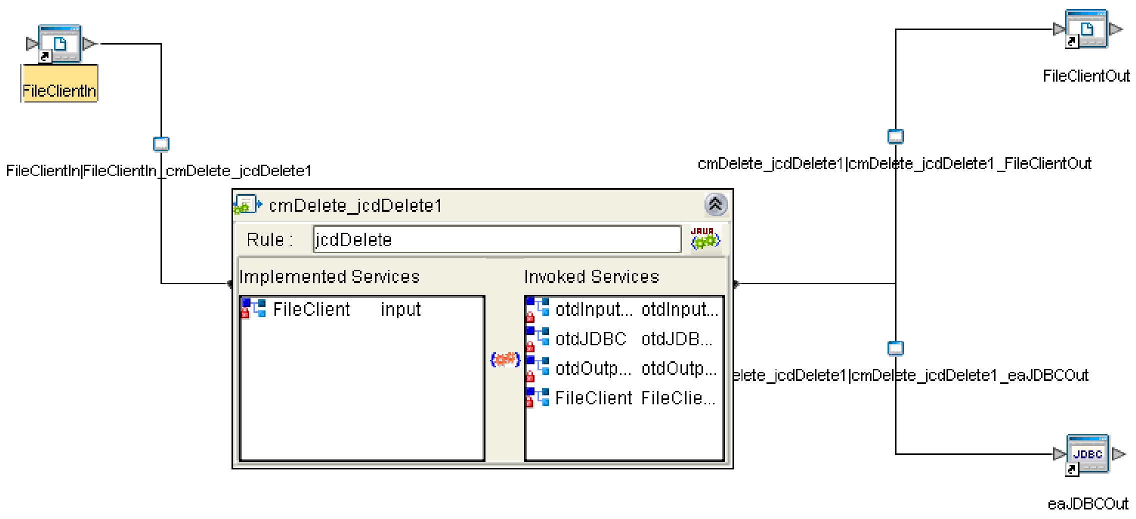
6.5.6 Binding the eWay Components

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

Steps required to bind eWay components together:

1. 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 **jcdDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **jcdDelete**.
5. From the **jcdDelete** Binding dialog box, map **otdJDBC_1** (under Invoked Services) to the **esJDBCOUT** External Application.
6. From the **jcdDelete** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 36.

Figure 36 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.

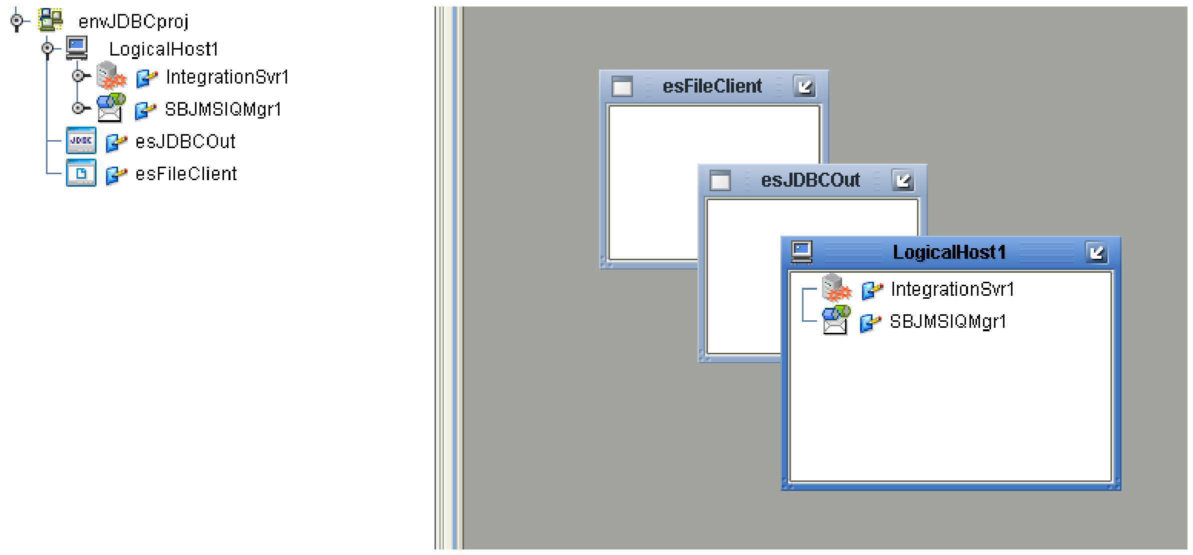
6.5.7 Creating an Environment

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 **envJDBCProj**.
4. Right-click **envJDBCProj** and select **New > JDBC External System**. Name the External System **esJDBC.** Click **OK**. **esJDBC** is added to the Environment Editor.
5. Right-click **envJDBCProj** and select **New > File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.
6. Right-click **envJDBCProj** 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 37.

Figure 37 Environment Editor - envJDBCProj



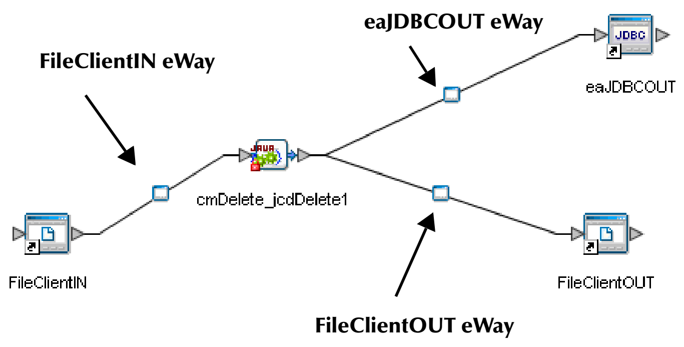
8. Save your current changes to the Repository.

6.5.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjJDBC_JCD** sample Project uses three eWays that are represented as nodes between the External Applications and the Business Process. See Figure 38.

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

Figure 38 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 15. Click **OK** to close the Properties Editor.

Table 15 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 **FileClientOUT** eWay on each of the Connectivity Maps and modify the properties for your system, as seen in Table 16. Click **OK** to close the Properties Editor.

Table 16 FileClientOUT eWay Property Settings

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

Table 16 FileClientOUT eWay Property Settings (Continued)

Connectivity Map	Property Name	Required Value
cmUpdate	Output 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 JDBC External System (**esJDBC** in this sample), and select **Properties**. The Properties Editor opens to the JDBC eWay Environment configuration.
2. Modify the JDBC eWay Environment configuration properties for your system, as seen in Table 17, and click **OK**.

Table 17 JDBC eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound JDBC eWay > JDBC Connector settings	ServerName	Enter the host name of the database server being used.
	DatabaseName	Enter the name of the particular database that is being used on the server.
	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 File eWay Environment configuration.
4. Modify the File eWay Environment configuration properties for your system, as seen in Table 18, and click **OK**.

Table 18 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 ▪ TriggerPsUpdate.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: ▪ JCD_Delete_output0.dat ▪ JCD_Insert_output0.dat ▪ JCD_PsSelect_output0.dat ▪ JCD_TableSelect_output0.dat ▪ JCD_PsUpdate_output0.dat

Configuring the Integration Server

You must set your 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.
4. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.
5. 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*.

6.5.9 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.

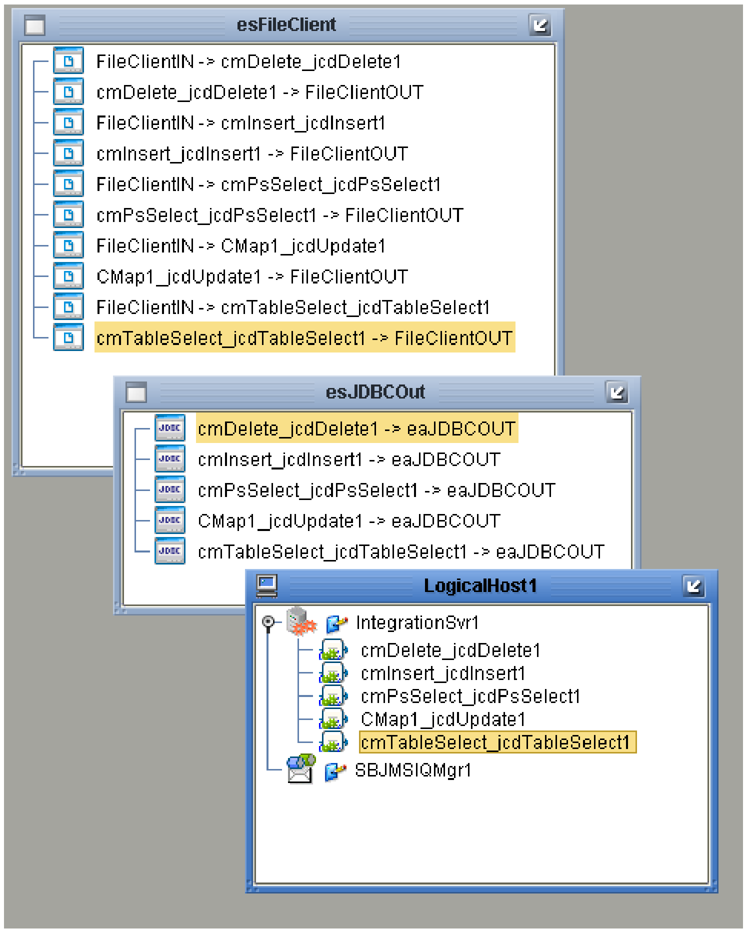
Steps required to create the Deployment Profile:

1 From the Enterprise Explorer's Project Explorer, right-click the **prjJDBC_JCD** Project and select **New** > **Deployment Profile**.

2 Enter a name for the Deployment Profile (for this sample **dpJDBC_JCD**). Select **envJDBCProj** 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. See Figure 39.

Figure 39 Deployment Profile


esFileClient
FileClientIN -> cmDelete_jcdDelete1
cmDelete_jcdDelete1 -> FileClientOUT
FileClientIN -> cmInsert_jcdInsert1
cmInsert_jcdInsert1 -> FileClientOUT
FileClientIN -> cmPsSelect_jcdPsSelect1
cmPsSelect_jcdPsSelect1 -> FileClientOUT
FileClientIN -> CMap1_jcdUpdate1
CMap1_jcdUpdate1 -> FileClientOUT
FileClientIN -> cmTableSelect_jcdTableSelect1
cmTableSelect_jcdTableSelect1 -> FileClientOUT

esJDBCOut
cmDelete_jcdDelete1 -> eaJDBCOUT
cmInsert_jcdInsert1 -> eaJDBCOUT
cmPsSelect_jcdPsSelect1 -> eaJDBCOUT
CMap1_jcdUpdate1 -> eaJDBCOUT
cmTableSelect_jcdTableSelect1 -> eaJDBCOUT

LogicalHost1
IntegrationSvr1
cmDelete_jcdDelete1
cmInsert_jcdInsert1
cmPsSelect_jcdPsSelect1
CMap1_jcdUpdate1
cmTableSelect_jcdTableSelect1
SBJMSIQMgr1

6.5.10 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.

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

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> 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. Your domain will continue to run unless you shut it down.
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*.

6.5.11 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.

6.5.12 Running the Sample

Additional steps are required to run the deployed sample Project.

Steps required to run the 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
- TriggerPsUpdate.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 JDBC eWay Sample Projects** on page 68 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 Building and Deploying the prjJDBC_BPEL Sample Project

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

Steps required to create the sample project include:

- **Creating a Project** on page 94
- **Creating the OTDs** on page 95
- **Creating the Business Process** on page 96
- **Creating the Connectivity Map** on page 106
- **Creating an Environment** on page 108
- **Configuring the eWays** on page 109
- **Creating the Deployment Profile** on page 112
- **Creating and Starting the Domain** on page 113
- **Building and Deploying the Project** on page 114
- **Running the Sample Project** on page 114

6.6.1 Creating a Project

The first step is to create a new Project in the 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 Right-click **Project1** and select **Rename** from the shortcut menu. Rename the Project (for this sample, **prjJDBC_BPEL**).

6.6.2 Creating the OTDs

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

- JDBC Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a JDBC 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 **JDBC Database OTD Wizard** from the list of OTD Wizards and click **Next**.

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

   - Driver Jar Files
   - Driver Java Class Name
   - URL Connection String
   - 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 JDBC 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 value for 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 **otdJDBC**.

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

Steps required to create inbound and outbound DTD OTDs:

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.

6.6.3 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 **bpelPsInsert**.

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

   - bpelUpdate
   - bpelDelete
   - bpTableSelect

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

Table 19 Business Process Activities

Business Process	Activity
bpelPsInsert	▪ FileClient.Receive ▪ FileClient.Write ▪ otdInputDTD_DBemployees.unmarshal ▪ otdJDBC.Insert_PsPSInsert ▪ FileClient.Write
bpelUpdate	▪ FileClient.receive ▪ FileClient.write ▪ otdJDBC.DB_EMPLOYEEUpdate ▪ FileClient.write
bpelDelete	▪ FileClient.receive ▪ FileClient.write ▪ otdJDBC.DB_EMPLOYEEDelete ▪ FileClient.write
bpelTableSelect	▪ FileClient.receive ▪ FileClient.write ▪ otdJDBC.DB_EMPLOYEESelectAll ▪ otdInputDTD_DBemployees.marshal ▪ FileClient.write ▪ FileClient.write

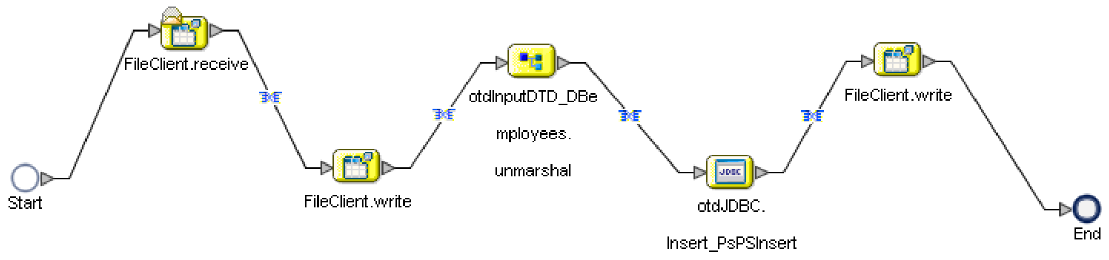
Configuring the bpelPsInsert Modeling Elements

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 40 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.*

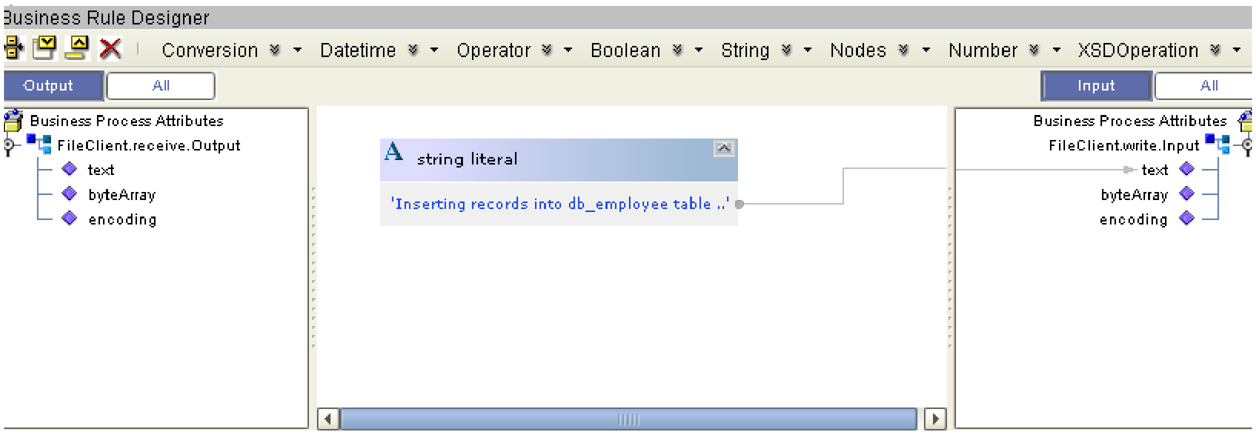
Figure 40 bpelPsInsert Business Process



Steps required to configure the bpelPsInsert business process:

1. Configure the business rule between the **FileClient.receive** and **FileClient.write** Activities, as seen in Figure 41.

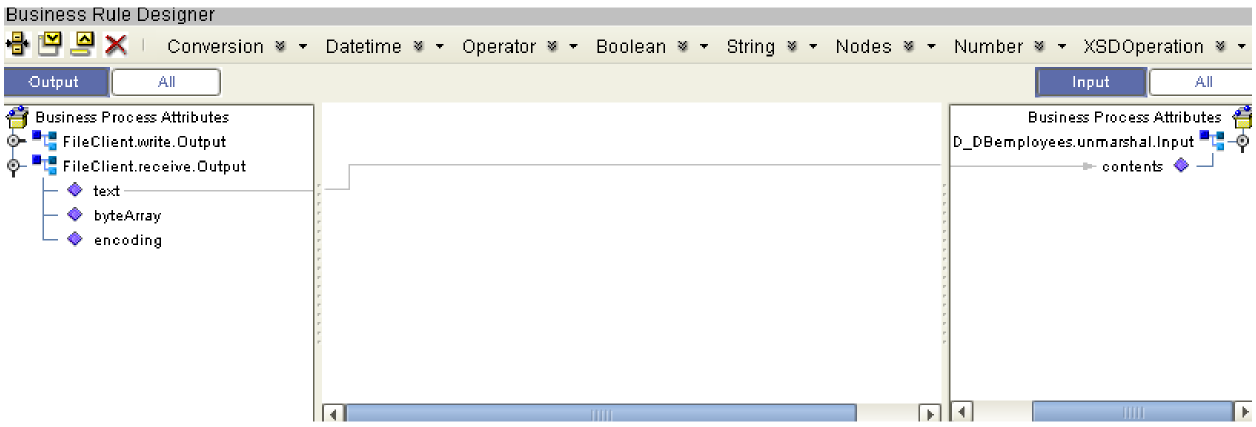
Figure 41 bpelPsInsert Business Rule # 1



'Inserting records into db_employee table ..'

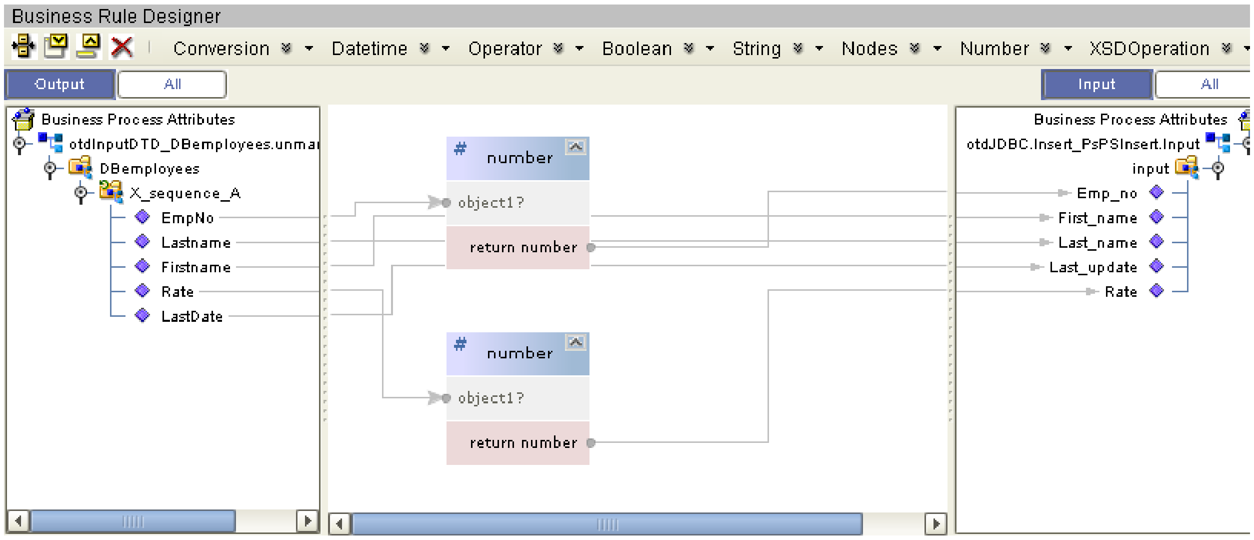
2. Configure the business rule between the **FileClient.write** Activity and **otdInputDTD_DBemployees.unmarshal** Activity, as seen in Figure 42.

Figure 42 bpelPsInsert Business Rule # 2



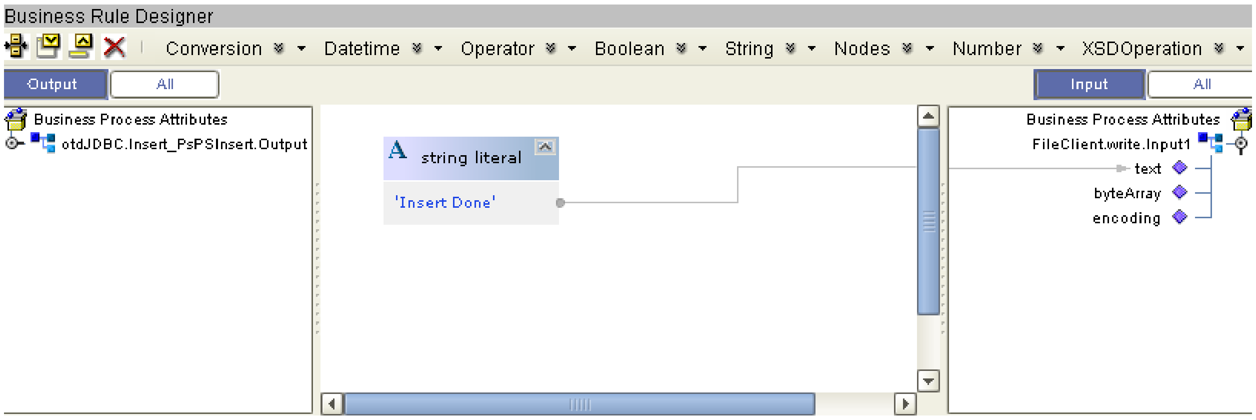
3. Configure the business rule between **otdInputDTD_DBemployees.unmarshal** and the **otdJDBC.Insert_psPSInsert** Activity, as seen in Figure 43.

Figure 43 bpelPsInsert Business Rule # 3



4 Configure the business rule between the **otdJDBC.Insert_psPSInsert** Activity and the **FileClient.write** Activity, as seen in Figure 44.

Figure 44 bpelPsInsert Business Rule # 4



Configuring the bpelUpdate Modeling Elements

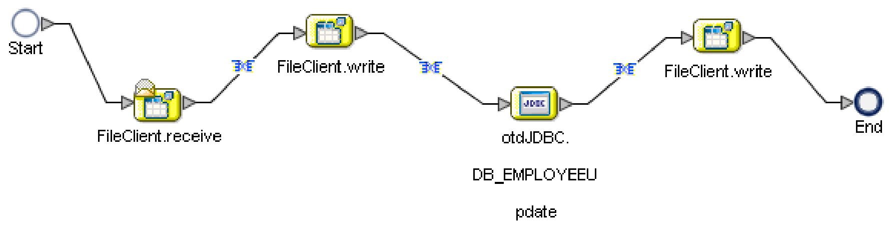
The bpelUpdate business process describes how to update a record in the JDBC database using the Business Process Designer.

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.*

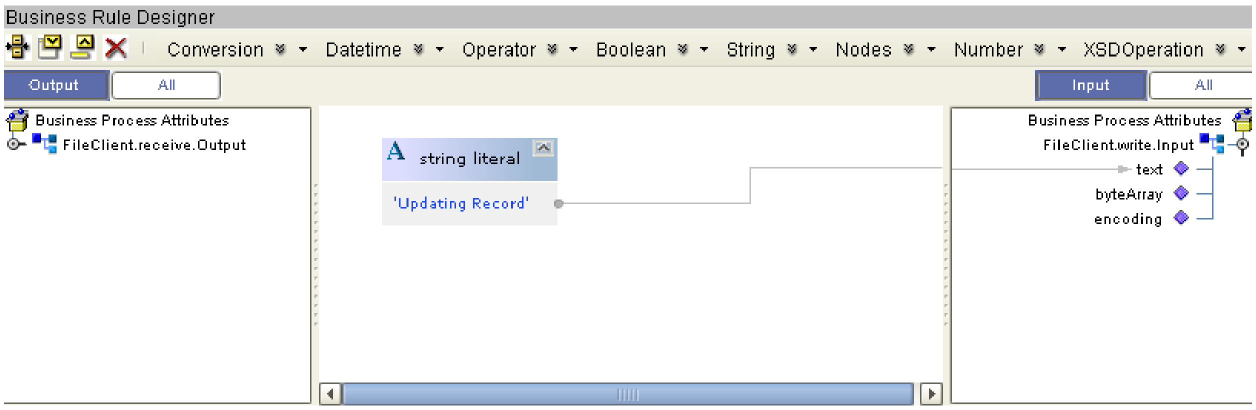
Figure 45 bpelUpdate Business Process



Steps required to configure the bpelUpdate business process:

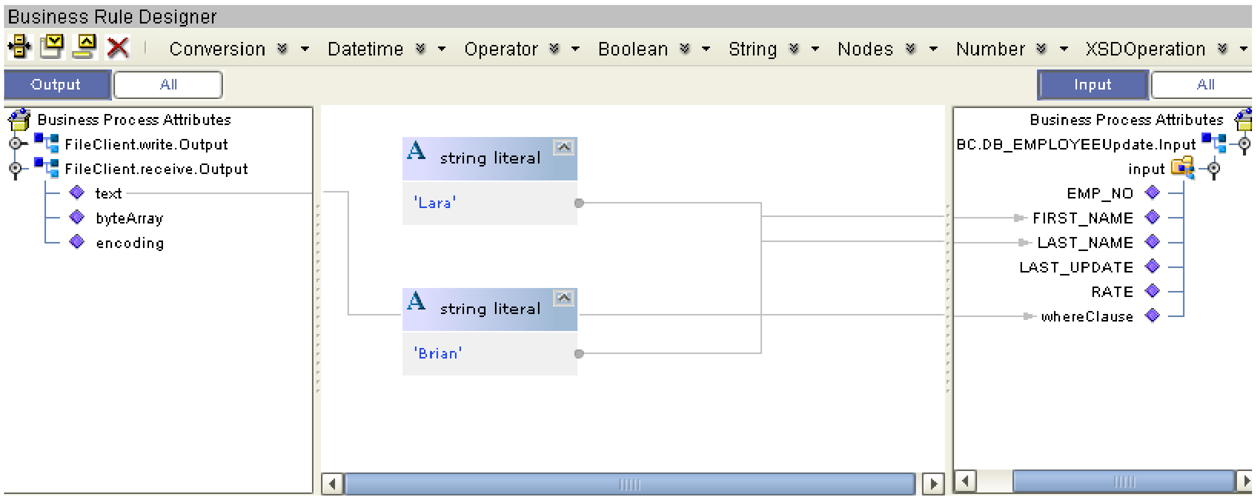
1. Configure the business rule between the **FileClient.receive** and **FileCleint.write** Activities, as seen in Figure 46.

Figure 46 bpelUpdate Business Rule # 1



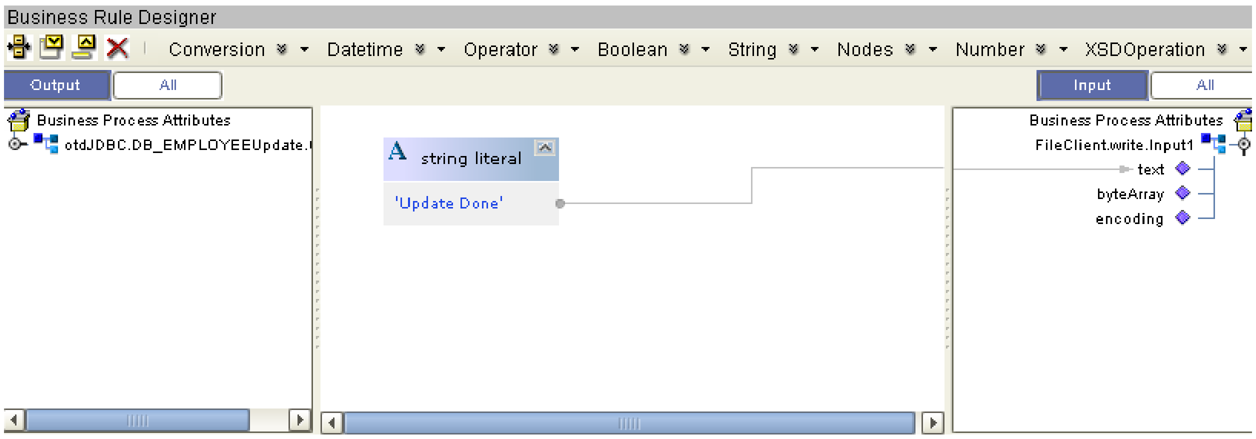
2. Configure the business rule between the **FileClient.write** Activity and **otdJDBC.DB_EMPLOYEEUpdate** Activity, as seen in Figure 47.

Figure 47 bpelUpdate Business Rule # 2



3 Configure the business rule between **otdJDBC.DB_EMPLOYEEUpdate** and the **FileClient.write** Activity, as seen in Figure 48.

Figure 48 bpelUpdate Business Rule # 3



Configuring the bpelDelete Modeling Elements

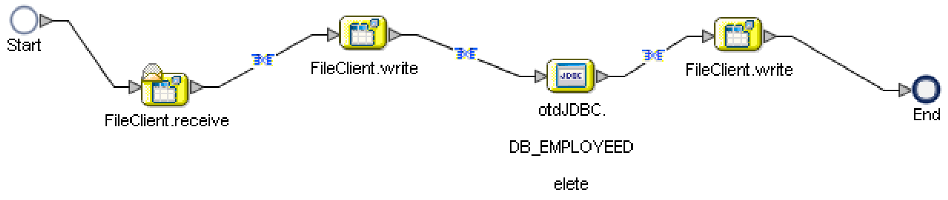
The bpelDelete business process describes how to delete a record in the JDBC database using the Business Process Designer.

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.*

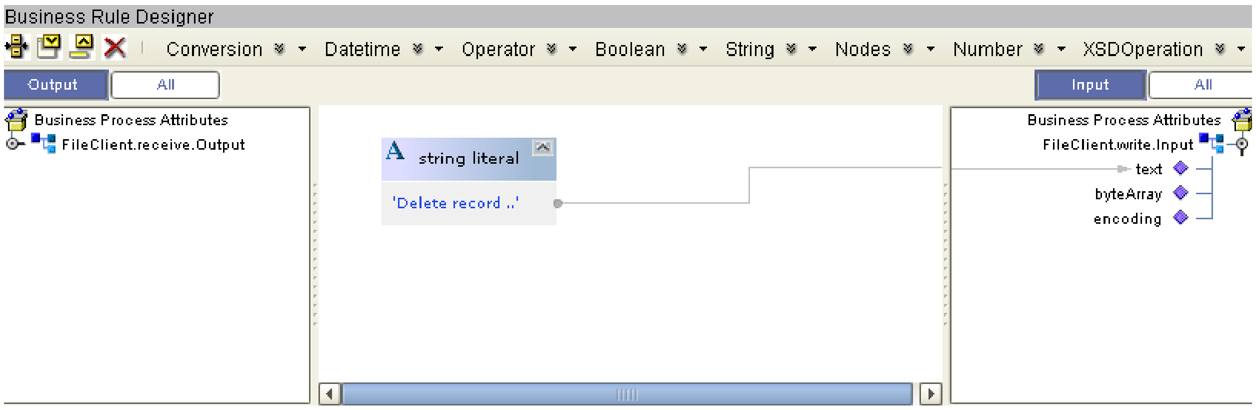
Figure 49 bpelDelete Business Process



Steps required to configure the bpelDelete business process:

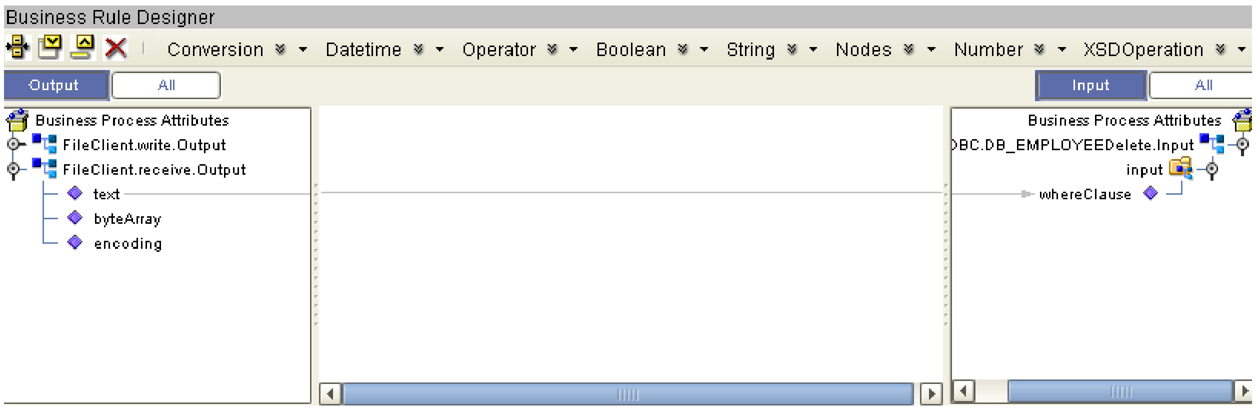
1. Configure the business rule between the **FileClient.receive** and **FileClient.write** Activities, as seen in Figure 50.

Figure 50 bpelDelete Business Rule # 1



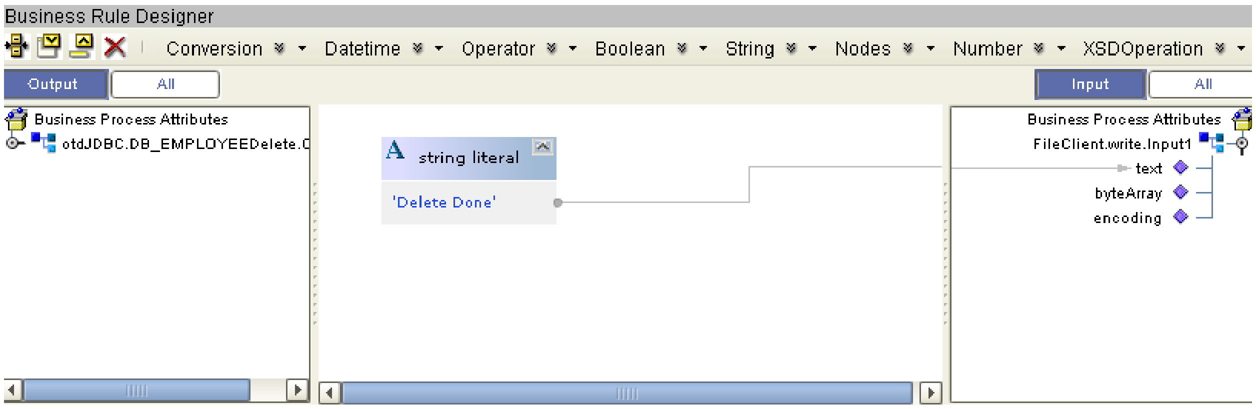
2. Configure the business rule between the **FileClient.write** Activity and **otdJDBC.DB_EMPLOYEEDelete** Activity, as seen in Figure 51.

Figure 51 bpDelete Business Rule # 2



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

Figure 52 bpelDelete Business Rule # 3



Configuring the bpelTableSelect Modeling Elements

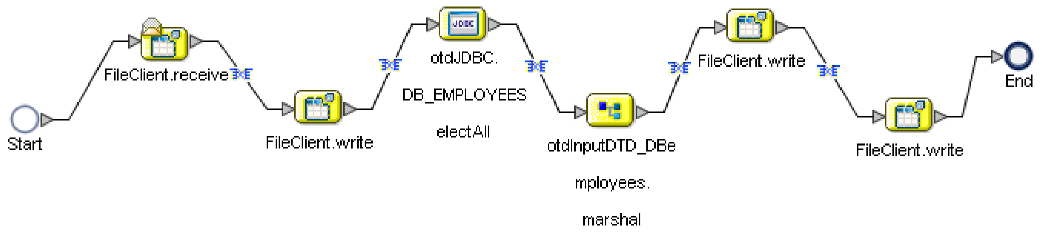
The bpelTableSelect business process is describes how to select all records the JDBC database using the Business Process Designer.

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.*

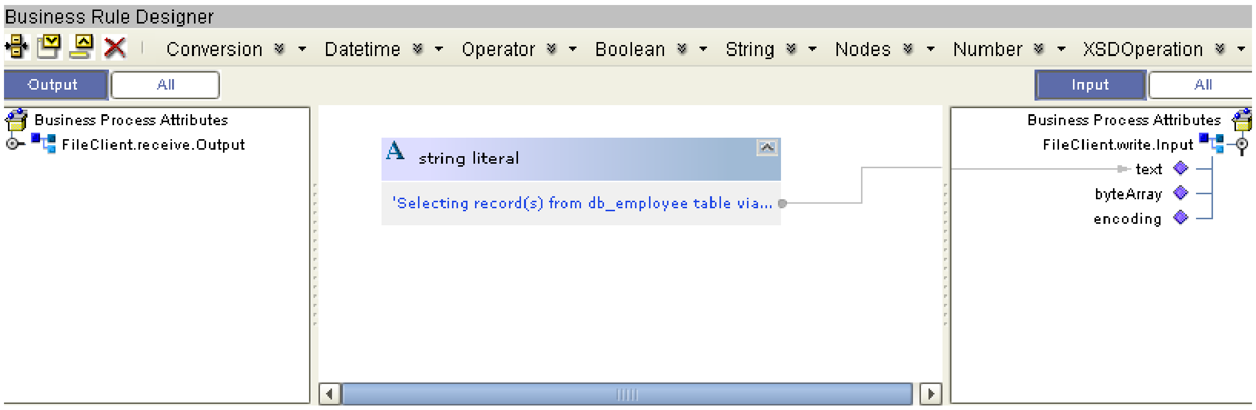
Figure 53 bpelTableSelect Business Process



Steps required to configure the bpelTableSelect business process:

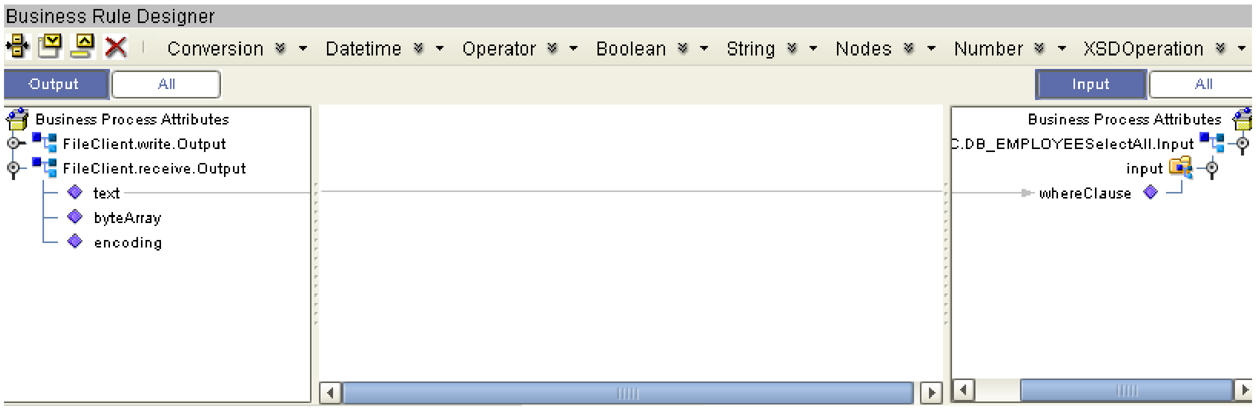
1 Configure the business rule between the **FileClient.receive** and **FileCleint.write** Activities, as seen in Figure 54.

Figure 54 bpelTableSelect Business Rule # 1



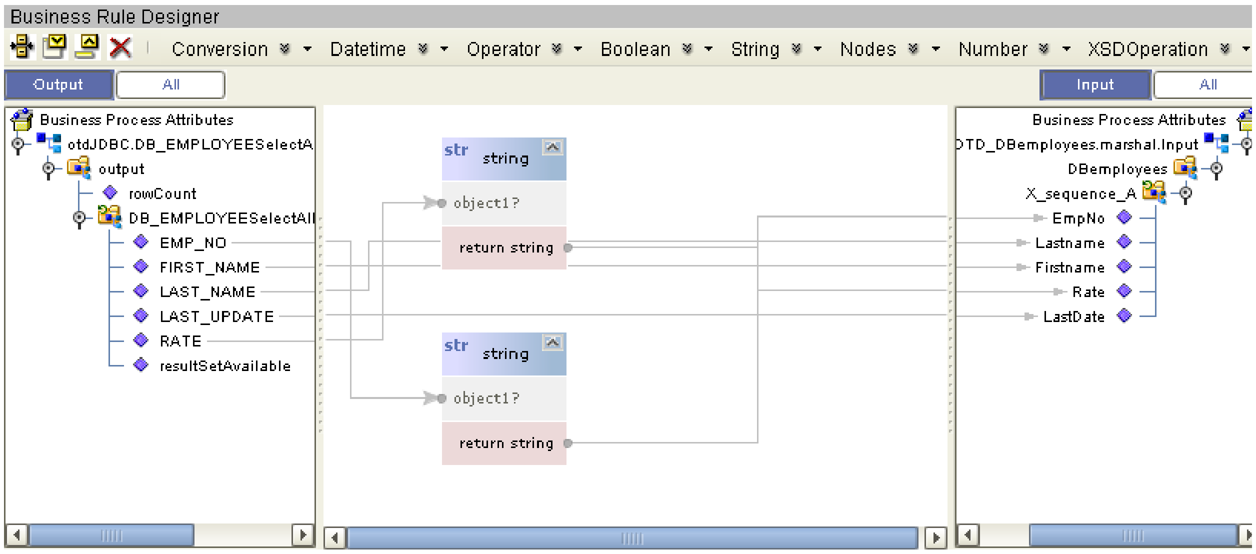
2 Configure the business rule between the **FileClient.write** Activity and **otdJDBC.DB_EMPLOYEESelectAll** Activity, as seen in Figure 55.

Figure 55 bpelTableSelect Business Rule # 2



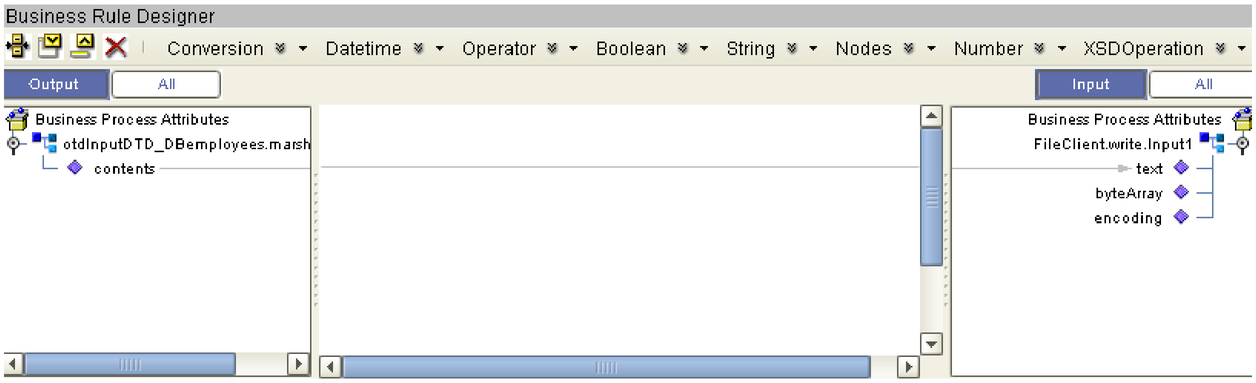
3 Configure the business rule between the **otdJDBC.DB_EMPLOYEESelectAll** Activity and the **otdInputDTD_DBemployees.marshal** Activity, as seen in Figure 56.

Figure 56 bpelSelectTable Business Rule # 3



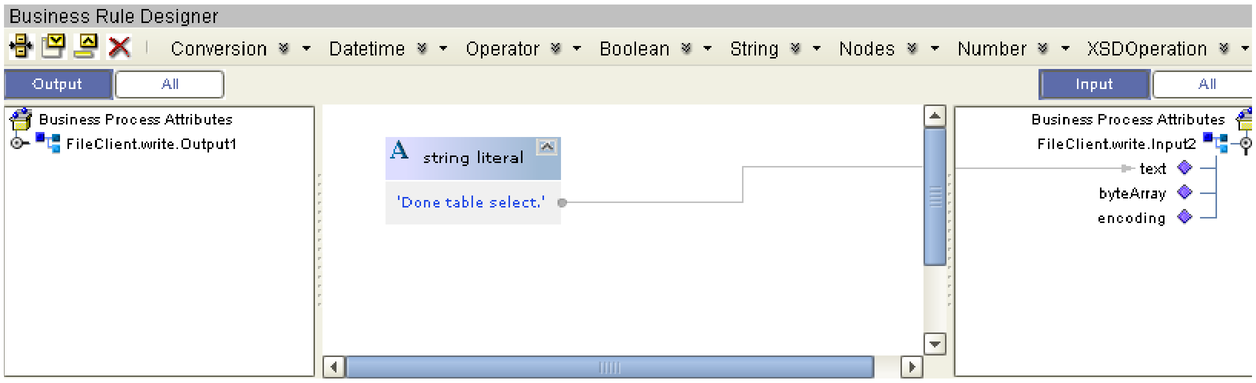
4 Configure the business rule between the **otdInputDTD_DBemployees.marshal** Activity and the **FileClient.write** Activity, as seen in Figure 57.

Figure 57 bpelTableSelect Business Rule # 4



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

Figure 58 bpelTableSelect Business Rule # 5



6.6.4 Creating the Connectivity Map

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

Steps required to create the Connectivity Map:

1. From the Project Explorer tree, right-click the new **prjJDBC_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 three additional Connectivity Maps—**CMap2**, **CMap3**, and **CMap4**—and rename them as follows:

   - cmDelete
   - cmPsInsert
   - 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 **prjJDBC_BPEL** sample Project requires the following components:

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

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

To Select a JDBC 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, **JDBC** 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
   - JDBC1 to eaJDBCOUT

To Select a JDBC 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

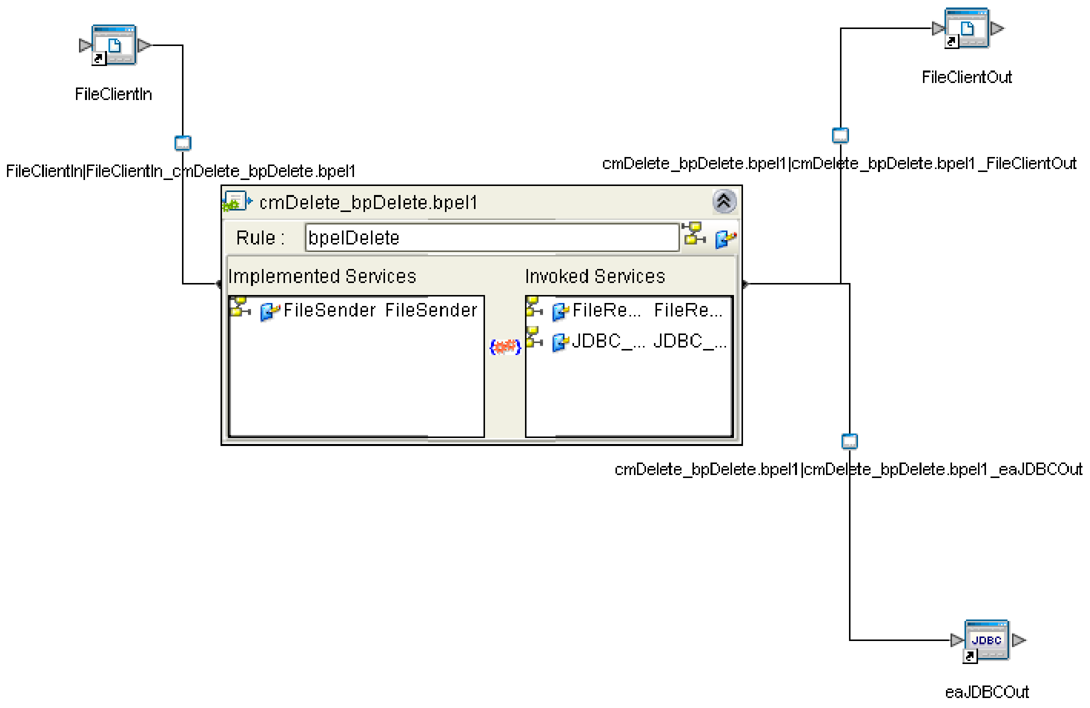
Binding the eWay Components

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 **bpelDelete** Business Process in the **cmDelete** Connectivity Map. The **bpelDelete** Binding dialog box appears.
2. From the **bpelDelete** 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 **bpelDelete**.
3. From the **bpelDelete** Binding dialog box, map **JDBC_otdJDBC** (under Invoked Services) to the **eaJDBCOUT** External Application.
4. From the **bpelDelete** Binding dialog box, map **FileReceiver** to the **FileClientOUT** External Application, as seen in Figure 59.

Figure 59 Connectivity Map - Associating (Binding) the Project's Components



5 Minimize the **bpelDelete** 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.

6.6.5 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.

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 **envJDBCProj**.

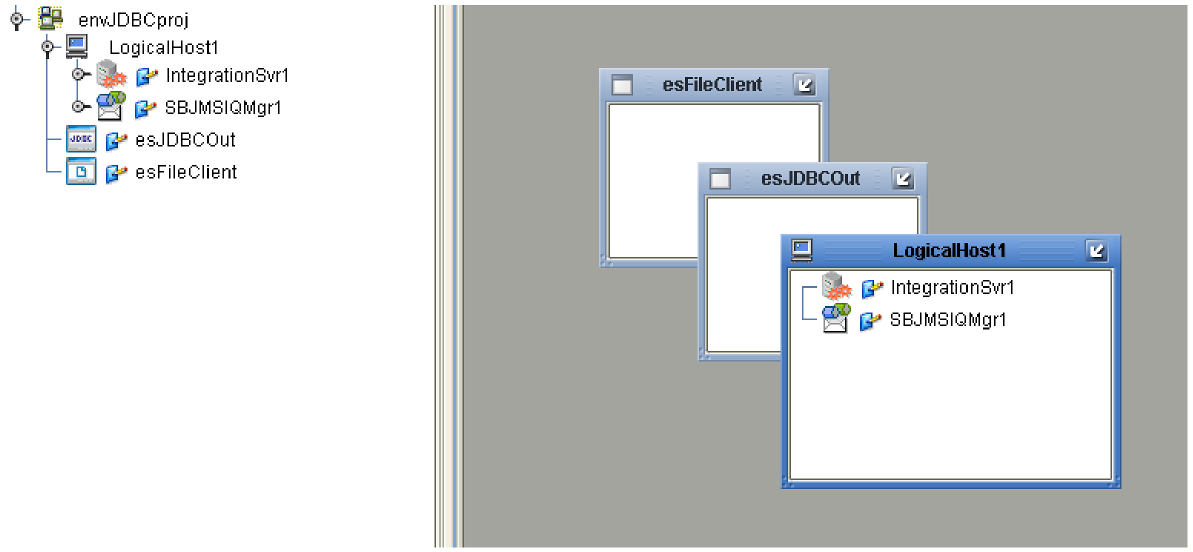
4 Right-click **envJDBCProj** and select **New > JDBC External System**. Name the External System **esJDBC**. Click **OK**. **esJDBC** is added to the Environment Editor.

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

6 Right-click **envJDBCProj** 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 60).

Figure 60 Environment Editor - envJDBCProj



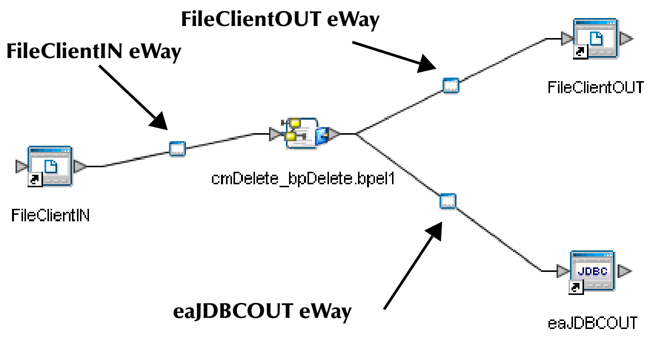
8 Save your current changes to the Repository.

6.6.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the The **prjJDBC_BPEL** sample Project use three eWays that are represented as a nodes between the External Applications and the Business Process, as seen in Figure 61.

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

Figure 61 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 20. Click **OK** to close the Properties Editor.

Table 20 FileClientIN eWay Property Settings

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

2. Double-click the **FileClientOUT** eWay on each of the Connectivity Maps and modify the properties for your system, as seen in Table 21. Click **OK** to close the Properties Editor.

Table 21 FileClientOUT eWay Property Settings

Connectivity Map	Property Name	Required Value
cmDelete	Output file name	BPEL_Delete_output%d.dat
cmInsert	Output file name	BPEL_Insert_output%d.dat
cmTableSelect	Output file name	BPEL_TableSelect_output%d.dat
cmUpdate	Output file name	BPEL_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 JDBC External System (**esJDBC** in this sample), and select **Properties**. The Properties Editor opens to the JDBC eWay Environment configuration.
2. Modify the JDBC eWay Environment configuration properties for your system, as seen in Table 22, and click **OK**.

Table 22 JDBC eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound JDBC eWay > JDBC Connector settings	ServerName	Enter the host name of the database server being used.
	DatabaseName	Enter the name of the particular database that is being used on the server.
	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 JDBC eWay Environment configuration.

4 Modify the File eWay Environment configuration properties for your system, as seen in Table 23, and click **OK**.

Table 23 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: ▪ TriggerBpPsInsert.in.~in ▪ TriggerDelete.in.~in ▪ TriggerTableSelect.in.~in ▪ TriggerBpUpdate.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_PsInsert_output0.dat ▪ BPEL_TableSelect_output0.dat ▪ BPEL_Update_output0.dat

Configuring the Integration Server

You must set your 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.
4. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.
5. 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*.

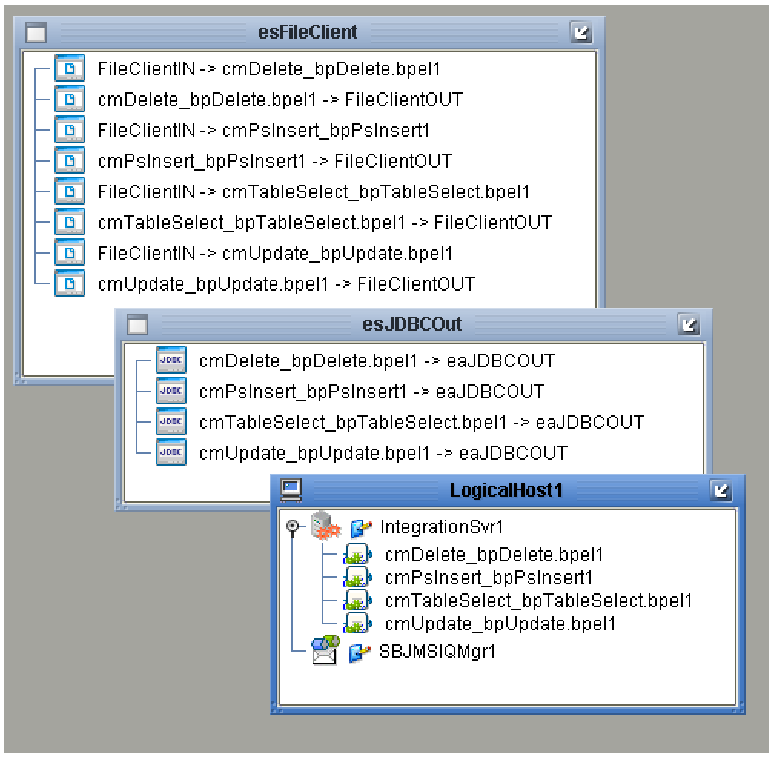
6.6.7 Creating the Deployment Profile

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.

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer's Project Explorer, right-click the **prjJDBC_BPEL** Project and select **New > Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpJDBC_BPEL**). Select **envJDBCProj** 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 62.

Figure 62 Deployment Profile


esFileClient
FileClientIN -> cmDelete_bpDelete.bpel1
cmDelete_bpDelete.bpel1 -> FileClientOUT
FileClientIN -> cmPsInsert_bpPsInsert1
cmPsInsert_bpPsInsert1 -> FileClientOUT
FileClientIN -> cmTableSelect_bpTableSelect.bpel1
cmTableSelect_bpTableSelect.bpel1 -> FileClientOUT
FileClientIN -> cmUpdate_bpUpdate.bpel1
cmUpdate_bpUpdate.bpel1 -> FileClientOUT
esJDBCOut
cmDelete_bpDelete.bpel1 -> eaJDBCOUT
cmPsInsert_bpPsInsert1 -> eaJDBCOUT
cmTableSelect_bpTableSelect.bpel1 -> eaJDBCOUT
cmUpdate_bpUpdate.bpel1 -> eaJDBCOUT
LogicalHost1
IntegrationSvr1
cmDelete_bpDelete.bpel1
cmPsInsert_bpPsInsert1
cmTableSelect_bpTableSelect.bpel1
cmUpdate_bpUpdate.bpel1
SBJMSIQMgr1

6.6.8 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 Sun Java Composite Application Platform Suite.*

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> 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. Your domain will continue to run unless you shut it down.
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.*

6.6.9 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.

6.6.10 Running the Sample Project

Additional steps are required to run the deployed sample Project.

Steps required to run the 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
   - TriggerBpUpdate.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 JDBC eWay Sample Projects** on page 68 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

JDBC/ODBC Drivers

Drivers are uniquely different in what they do and the type of functions they support. The JDBC/ODBC eWay allows you to pick and choose which driver is best suited for your application environment. There can be significant differences and limitations between drivers. The performance and functionality of the JDBC/ODBC eWay depends on the selected driver(s). Certain drivers may not support all JDBC features. Consult the documentation for your respective drivers for more information.

This appendix provides database configuration information and environment properties specifications for specific JDBC/ODBC drivers. You should use the information listed in the included tables to define values for required input parameters.

While any standards compliant JDBC/ODBC database driver may be used, the drivers covered in this chapter are used more frequently. For runtime, only drivers that support Connection Pool Data Source and XA Data Source are supported. Connection Pool Data Source takes advantage of the Integration Service's connection pooling in order to improve performance. For the OTD Wizard, the driver Manager Class will work. However, not all drivers support all metadata discovery methods, some of which are needed to build the OTD. Additionally, not all drivers support Updatable ResultSets, Stored Procedures, or Stored Procedures with ResultSets. Check with your driver vendor for what is supported. The ConnectionPoolDataSource should only be used for Outbound eWays. The Inbound eWay uses native JDBC and must use Driver Manager (see **Table 6 on page 31**).

It is recommended that you use the Oracle eWay when using the native Oracle driver. The JDBC eWay does not support some of the functions available in the Oracle eWay such as creating an OTD from a Prepared Statement, using a Stored Procedure with ResultSets, and CLOB support.

It is also recommended that you use the SQL Server eWay. The JDBC driver available for download from the Microsoft web site may not contain the latest version from the vendor.

Not all drivers support Updatable ResultSets. However, it does allow standard Insert and Update operations when used with the Prepared Statement feature:

```
Insert into employee (empno) values(?);
```

Remember to ensure that the input parameter data types match the data types specified in the database table targeted by the Prepared Statements as some drivers always return the data type as a string. Optionally, you may perform the data conversion in the Collaboration.

What's In This Chapter:

- **AS/400 Toolbox Driver** on page 117
- **Attunity Driver** on page 118
- **MySQL Connector/J Driver** on page 120
- **PostgreSQL Driver** on page 121
- **SyBase JConnect Driver** on page 123
- **Sequelink DataDirect Informix ODBC Driver** on page 124
- **Sequelink DataDirect MS Access ODBC Driver** on page 125
- **Teradata Driver** on page 127
- **Installing JDBC/ODBC Drivers** on page 128
- **Troubleshooting** on page 129

Refer to the supplied *Readme.txt* file for information regarding the location of the required driver files necessary to connect using the JDBC/ODBC eWay.

A.1 AS/400 Toolbox Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 117
- **Environment Properties** on page 118

A.1.1 OTD Wizard: Database Connection Information

To connect to AS/400, use the information provided in Table 1 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard. To access DB2, it is recommended to use the DB2 eWay Adapter or the DB2 Connect eWay Adapter.

Table 1 AS/400 Database Connection Information

Parameter	Value
Driver Jar Files	jt400.jar
Driver Java Class Name	com.ibm.as400.access.AS400JDBCDriver
URL Connection String	jdbc:as400://<server-name>:<server-port>/ **Note:** *NOTE: Default server port is **446**.*
User Name	Login name of the account used to access the AS/400 database.
Password	Password associated with the login account name used to connect to the AS/400 database.

A.1.2 Environment Properties

Use Table 2 to configure the environment properties for the specified JDBC/ODBC driver.

Table 2 AS/400 Database Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	com.ibm.as400.access.AS400JDBCConnectionPoolDataSource
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting the database.
PortNumber	<server-port> **Note:** *NOTE: Default server port is **446**.*
DatabaseName	
User	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.
DriverProperties	
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.
MaxIdleTime	The default is **0**.

A.2 Attunity Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 118
- **Environment Properties** on page 119

A.2.1 OTD Wizard: Database Connection Information

To connect to Attunity, use the information provided in Table 3 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard.

Table 3 Attunity Driver Database Connection Information

Parameter	Value
Driver Jar Files	nvjdbc2.jar
Driver Java Class Name	com.attunity.jdbc.NvDriver
URL Connection String	jdbc:attconnect://<server-name>;DefTdpName=<database-logical-name>;OneTdpMode=1 **Note:** *The* <database-logical-name> is created in the Attunity server.
User Name	Leave password field blank. Value configured when the database entry is created in the Attunity Server.
Password	Leave password field blank. Value configured when the database entry is created in the Attunity Server.

A.2.2 Environment Properties

Use Table 4 to configure the environment properties for the specified JDBC/ODBC driver.

Table 4 Attunity Driver Database Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	com.attunity.jdbc.NvXADataSource
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting the database.
PortNumber	<server-port> **Note:** *NOTE: Default server port is* ***2551****.*
DatabaseName	<database-name>
User	Leave user field blank. Value configured when the database entry is created in the Attunity Server.
Password	Leave password field blank. Value configured when the database entry is created in the Attunity Server.
DriverProperties	setDefTdpName#<database-logical-name>##setWorkspace#Navigator##
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.

Table 4 Attunity Driver Database Environment Properties (Continued)

Parameter	Value
MaxIdleTime	The default is **0**.

A.3 MySQL Connector/J Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 120
- **Environment Properties** on page 120

A.3.1 OTD Wizard: Database Connection Information

To connect to MYSQL, use the information provided in Table 5 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard.

Table 5 MySQL Connector/J Driver Database Connection Information

Parameter	Value
Driver Jar Files	mysql-connector-java-3.0.11-stable-bin.jar
Driver Java Class Name	com.mysql.jdbc.Driver
URL Connection String	jdbc:mysql://<server-name>:<server-port>/<database-name> **Note:** *NOTE: Default server port is **3306***
User Name	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.

A.3.2 Environment Properties

Use Table 6 to configure the environment properties for the specified JDBC/ODBC driver.

Table 6 MySQL Connector/J Driver Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	ccom.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSou rce
ClassNamefor OtherInterfaces	

Table 6 MySQL Connector/J Driver Environment Properties (Continued)

Parameter	Value
ServerName	Server name of the machine hosting the database.
PortNumber	<server-port> **Note:** *NOTE: Default server port is **3306**.*
DatabaseName	<database-name>
User	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.
DriverProperties	
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.
MaxIdleTime	The default is **0**.

Note: *It is not mandatory to enter driver properties in the Outbound JDBC eWay Environment properties for MySQL.*

A.4 PostgreSQL Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 121
- **Environment Properties** on page 122

A.4.1 OTD Wizard: Database Connection Information

To connect to SQL, use the information provided in Table 7 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard. To access SQL, it is recommended to use the SQL Server eWay Adapter.

Table 7 PostgreSQL Driver Connection Information

Parameter	Value
Driver Jar Files	postgresql-8.0-310.jdbc3.jar
Driver Java Class Name	org.postgresql.Driver

Table 7 PostgreSQL Driver Connection Information (Continued)

Parameter	Value
URL Connection String	jdbc:postgresql://<server-name>:<server-port>/<database-name> **Note:** *NOTE: Default server port is* ***5432****.*
User Name	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.

A.4.2 Environment Properties

Use Table 8 to configure the environment properties for the specified JDBC/ODBC driver.

Table 8 PostgreSQL Driver Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	org.postgresql.jdbc3.Jdbc3ConnectionPool
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting the database.
PortNumber	<server-port> **Note:** *NOTE: Default server port is* ***5432****.*
DatabaseName	<database-name>
User	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.
DriverProperties	
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.
MaxIdleTime	The default is **0**.

Note: *It is not mandatory to enter driver properties in the Outbound JDBC eWay Environment properties for PostgreSQL.*

A.5 SyBase JConnect Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 123
- **Environment Properties** on page 123

A.5.1 OTD Wizard: Database Connection Information

To connect to Sybase, use the information provided in Table 9 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard. To access Sybase, it is recommended to use the Sybase eWay Adapter.

Table 9 Sybase JConnect Driver Database Connection Information

Parameter	Value
Driver Jar Files	jconn2.jar
Driver Java Class Name	com.sybase.jdbc2.jdbc.SybDriver
URL Connection String	jdbc:sybase:Tds:<server-name>:<server-port> **Note:** *NOTE: Default server port is **4100**.*
User Name	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.

A.5.2 Environment Properties

Use Table 10 to configure the environment properties for the specified JDBC/ODBC driver.

Table 10 Sybase JConnect Driver Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting the database.
PortNumber	<server-port> **Note:** *NOTE: Default server port is **4100**.*
DatabaseName	<database-name>
User	Login name of the account used to access the database.

Table 10 Sybase JConnect Driver Environment Properties (Continued)

Parameter	Value
Password	Password associated with the login account name used to connect to the database.
DriverProperties	
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.
MaxIdleTime	The default is **0**.

Note: *It is not mandatory to enter driver properties in the Outbound JDBC eWay Environment properties for Sybase.*

A.6 Sequelink DataDirect Informix ODBC Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 124
- **Environment Properties** on page 125

A.6.1 OTD Wizard: Database Connection Information

The settings in Table 11 describe how to use the DataDirect Sequelink JDBC/ODBC bridge with the JDBC/ODBC eWay. This information demonstrates how Sequelink can be used to interface with the ODBC driver. To connect to an Informix database, it is recommended to use the Informix eWay Adapter.

Table 11 Sequelink DataDirect Informix ODBC Driver Database Connection Information

Parameter	Value
Driver Jar Files	sljc.jar
Driver Java Class Name	com.ddtek.jdbc.sequelkink.SequeLinkDriver
URL Connection String	jdbc:sequelink://<server-name>:<server-port> **Note:** *NOTE: Default server port is **19996**.*
User Name	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.

A.6.2 Environment Properties

Use Table 12 to configure the environment properties for the specified JDBC/ODBC driver.

Table 12 Sequelink DataDirect Informix ODBC Driver Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	com.ddtek.jdbcx.sequelkink.SequeLinkDataSource
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting Sequelink.
PortNumber	<server-port> **Note:** *NOTE: Default server port is **19996**.*
DatabaseName	
User	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.
DriverProperties	
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.
MaxIdleTime	The default is **0**.

Note: *It is not mandatory to enter driver properties in the Outbound JDBC eWay Environment properties for Sequelink DataDirect Informix ODBC.*

A.7 Sequelink DataDirect MS Access ODBC Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 126
- **Environment Properties** on page 126

A.7.1 OTD Wizard: Database Connection Information

To connect to Microsoft Access, via the Microsoft Access ODBC driver, use the information provided in Table 13 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard.

Table 13 MS Access ODBC Driver Database Connection Information

Parameter	Value
Driver Jar Files	sljc.jar
Driver Java Class Name	com.ddtek.jdbc.sequelink.SequeLinkDriver
URL Connection String	jdbc:sequelink://<server-name>:<server-port> **Note:** *NOTE: Default server port is **19996**.*
User Name	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.

A.7.2 Environment Properties

Use Table 14 to configure the environment properties for the specified JDBC/ODBC driver.

Table 14 MS Access ODBC Driver Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	com.ddtek.jdbcx.sequelkink.SequeLinkDataSource
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting Sequelink.
PortNumber	<server-port> **Note:** *NOTE: Default server port is **19996**.*
DatabaseName	
User	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.
DriverProperties	
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **0**.
MaxPoolSize	The default is **10**.

Table 14 MS Access ODBC Driver Environment Properties (Continued)

Parameter	Value
MaxIdleTime	The default is **0**.

Note: *It is not mandatory to enter driver properties in the Outbound JDBC eWay Environment properties for Sequelink DataDirect MS Access ODBC.*

A.8 Teradata Driver

Configuration Properties:

- **OTD Wizard: Database Connection Information** on page 127
- **Environment Properties** on page 127

A.8.1 OTD Wizard: Database Connection Information

To connect to Teradata, via the Teradata driver, use the information provided in Table 15 to complete the Connect to Database step of the JDBC/ODBC OTD Wizard.

Table 15 Teradata Driver Database Connection Information

Parameter	Value
Driver Jar Files	teradata.jar
Driver Java Class Name	com.ncr.teradata.TeraDriver
URL Connection String	jdbc:teradata://<server-name>:<server-port>/<database-server-name> **Note:** *NOTE: Default server port is **6666** for the Type-3 driver Gateway.*
User Name	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.

A.8.2 Environment Properties

Use Table 16 to configure the environment properties for the specified JDBC/ODBC driver.

Table 16 Teradata Driver Environment Properties

Parameter	Value
Description	JDBC Connection Pool Datasource
ClassName	com.ncr.teradata.TeraConnectionPoolDataSource

Table 16 Teradata Driver Environment Properties (Continued)

Parameter	Value
ClassNamefor OtherInterfaces	
ServerName	Server name of the machine hosting the database.
PortNumber	<server-port> **Note:** *NOTE: Default server port is* ***6666*** *for the Type-3 driver Gateway.*
DatabaseName	<database-name>
User	Login name of the account used to access the database.
Password	Password associated with the login account name used to connect to the database.
DriverProperties	setURL#jdbc:teradata://<server-name>:<server-port>/<database-server-name>##setDSName#<database-server_name>##
Delimiter	The default is **#.**
DataSourceName	
MinPoolSize	The default is **2**.
MaxPoolSize	The default is **10**.
MaxIdleTime	The default is **0**.

A.9 Installing JDBC/ODBC Drivers

The database drivers specified in your projects need to be installed on both the Enterprise Designer machine and the Logical host machine. When installing the drivers on the Enterprise Designer machine, you must specify the absolute path to the driver. When installing the drivers on the Logical Host, place the driver into the Logical Host `stcis` directory:

```
<JavaCAPS51>\logicalhost\is\lib; or
```

```
<JavaCAPS51>\logicalhost\is\domains\domain1\lib
```

where *<JavaCAPS51>* is the location of your Sun Java Composite Application Platform Suite installation.

The driver file must be copied to the latter folder if you are running multiple domains and wish to specify a driver for each domain. Otherwise, you only need to copy the driver file to the former folder address.

For procedures on how to install database drivers, see **"Sample Projects Drivers" on page 71**.

A.10 Troubleshooting

Refer to the following when troubleshooting Driver issues.

- The ReceiveOne operation in BPEL is not supported when using inbound functions with some drivers.
- Some drivers do not support Updatable ResultSets. If you find this to be the case, use a Prepared Statement to Update, Insert, and Delete data.
- Not all drivers provide metadata information such as column names and data types. If your table does not have column names and data types, add them before saving the OTD.

Index

A

activity input and output **56**
Add Prepared Statements **51**
alert codes, viewing **21**
AS/400 Driver **117**
Attunity Driver **118**
Automap **92**, **112**

B

binding
  dialog box **87**, **108**
BPEL operations **56**
  Delete **56**
  Insert **56**
  ReceiveOne **56**
  SelectAll **56**
  SelectMultiple **56**
  SelectOne **56**
  Update **56**

C

Collaboration
  editor **77**
configuring JDBC eWay **24**
connection retry settings, outbound **33**, **36**, **38**
Connectivity Map
  Inbound JDBC eWay Properties **31**
  Outbound JDBC eWay Properties **31**, **34**, **36**
conventions, text **15**

D

database operations (BPEL) **56**
  activity input and output **56**
  Delete **56**
  Insert **56**
  ReceiveOne **56**
  SelectAll **56**
  SelectMultiple **56**
  SelectOne **56**
  Update **56**
database operations (JCD) **58**
  delete(String sWhere) **58**
  deleteRow() **58**
  insert() **58**
  insertRow() **58**
  select(String where) **58**
  update(String sWhere) **58**
  updateRow() **58**
database OTD wizard
  add prepared statement **51**
  editing existing OTDs **55**
  review selections **54**
  select procedures **47**
  select tables/views **43**
  specify the OTD name **54**
DataDirect **124**, **125**
Deployment Profile
  Automap **92**, **112**
driver class, JDBC **32**, **34**, **37**
Drivers **116**
  AS/400 **117**
  Attunity **118**
  installing **128**
  MySQL Connector/J **120**
  PostgreSQL Driver **121**
  Sequelink Informix ODBC **124**
  Sequelink MS Access ODBC **125**
  Sybase JConnect **123**
  Teradata **127**

E

eWay Connectivity Map **24**, **29**
eWay environment properties **28**
eWay plug-ins, installing **20**
eWay Properties
  Inbound JDBC eWay Properties **31**
  Outbound JDBC eWay Properties **31**, **34**, **36**

I

Importing sample Projects **72**
Inbound JDBC eWay Properties **31**
installation **16**
Installing
  alert codes **21**
  eWay plug-ins **20**
  migration procedures **18**
  Repository on UNIX **16**
  sample Projects and Javadocs **18**

J

Javadocs, installing **18**

JCD operations**58**
    delete(String sWhere)**58**
    deleteRow()**58**
    insert()**58**
    insertRow()**58**
    select(String where)**58**
    update(String sWhere)**58**
    updateRow()**58**
JDBC
    driver class**32**, **34**, **37**
JDBC eWay Project
    building and deploying (BPEL)**94**
    eInsight and eGate components**71**
    Importing**72**
    running sample projects**93**, **114**
    running the SQL script**72**
JDBC_BPEL_Sample**68**

M

migration procedures**18**
MySQL Connector/J Driver**120**

O

operations**56**
OTD, editing existing**55**
outbound eWay properties**30**
Outbound JDBC eWay Properties**31**, **34**, **36**
outbound XA properties**30**

P

PostgreSQL Driver**121**
prepared statement
    batch operations**66**
    executing**66**
Project
    importing**72**
properties
    ConnectionRetries**33**, **36**, **38**
    ConnectionRetryInterval**34**, **36**, **39**
    DatabaseName**32**, **35**, **37**
    Delimiter**32**, **35**, **37**
    Description**30**, **31**, **32**, **34**, **37**
    DriverProperties**32**, **35**, **37**
    MaxIdleTime**33**, **36**, **38**
    MaxPoolSize**33**, **36**, **38**
    MinPoolSize**33**, **35**, **38**
    Password**31**, **32**, **35**, **37**
    PortNumber**32**, **35**, **37**
    ServerName**32**, **34**, **37**
    User**31**, **32**, **35**, **37**

R

ResultSet
    collaboration usability for a stored procedure**65**
ResultSet methods
    available**64**
    enableResultSetandUpdateCounts**64**
    enableResultSetOnly**64**
    enableUpdateCountsOnly**64**
    getUpdateCount**64**
    next**64**
    resultsAvailable**64**

S

sample Projects**72**
sample projects, installing**18**
Select Database Objects**42**
Select Procedures**48**
Select Table/Views**43**
Sequelink Informix ODBC Driver**124**
Sequelink MS Access ODBC Driver**125**
Setting Properties
    configuring JDBC eWay**24**
    connection retry settings**33**, **36**, **38**
    eWay Connectivity Map**24**, **29**
    eWay environment properties**28**
    outbound eWay**30**
    outbound XA properties**30**
SQL**59**
SQL operations, table**59**
SQL operations, table Delete**62**
SQL operations, table Insert**60**
SQL operations, table Select**59**
SQL operations, table Update**61**
Stored Procedure
    executing**63**
    manipulating ResultSet and update count**64**
supporting documents**15**
Sybase JConnect Driver**123**

T

Table
    SQL operations**59**
    SQL operations, Delete**62**
    SQL operations, Insert**60**
    SQL operations, Select**59**
    SQL operations, Update**61**
Teradata Driver**127**
text conventions**15**
Transaction Support Levels**26**
Troubleshooting**129**

U

update count**64**