SUN SEEBEYOND

eWAY™ ADAPTER FOR SQL SERVER USER'S GUIDE

Release 5.1.1

Version 20060626091408

Contents

Chapter 1

Introducing the SQL Server eWay 7

About SQL Server 7

About the SQL Server eWay 7

What's New in This Release 8

About this Document 8

What's in this Document 9

Scope 9

Intended Audience 9

Text Conventions 9

Screenshots 10

Related Documents 10

Sun Microsystems, Inc. Web Site 10

Documentation Feedback 10

Chapter 2

Installing the SQL Server eWay 11

Before You Install 11

Installing the SQL Server eWay 11

Installing the SQL Server eWay on an eGate supported system 12

Adding the eWay to an Existing Sun Java Composite Application Platform Suite Installation 12

Extracting the Sample Projects and Javadocs 13

Configuring the DataDirect 3.5 Drivers for XA Mode 13

Installing Stored Procedures for JTA 14

After You Install 15

Installing eWay Enterprise Manager Plug-Ins 15

Viewing Alert Codes 16

Chapter 3

Configuring the SQL Server eWay 18

Creating and Configuring the SQL Server eWay 18
Selecting SQL Server as the External Application 18
Configuring the SQL Server eWay Properties 19
Transaction Support Levels Between Different Versions 20
Using the Properties Editor 22

SQL Server eWay Connectivity Map Properties 23
Outbound Connectivity Map Properties 23
Properties in the Outbound eWay 23
Properties in the Outbound non-Transactional eWay 24
Properties in the Outbound eWay with XA Support 24
Inbound Connectivity Map Properties 24

SQL Server eWay Environment Configuration Properties 25
Inbound SQL Server eWay Properties 26
Outbound SQL Server eWay Properties 27
JDBC Connector Settings 27
Connection Retry Settings 29
Outbound SQL Server XA eWay Properties 30
JDBC Connector Settings (with XA support) 30
Connection Retry Settings (with XA support) 32

Adding JAR Files to the LogicalHost 33

Chapter 4

Using the SQL Server eWay Database Wizard 34

Using the Database OTD Wizard 34

Steps to Create a New SQL Server OTD 35
Select Wizard Type 35
Connect to Database 36
Select Database Objects 36
Select Table/Views/Aliases 37
Select Procedures 40
Add Prepared Statements 43
Specify the OTD Name 46
Review Selections 46

Editing Existing OTDs 47

Chapter 5

Using SQL Server Operations 49

SQL Server eWay Database Operations (BPEL) 49
Activity Input and Output 49

SQL Server eWay Database Operations (JCD) 51
The Table 51
The Query (Select) Operation 51
The Insert Operation 52
The Update Operation 53
The Delete Operation 54
The Stored Procedure 55
Executing Stored Procedures 55
Manipulating the ResultSet and Update Count Returned by Stored Procedure 56
Prepared Statement 58
Batch Operations 59

Chapter 6

Implementing SQL Server Sample Projects 60

About the SQL Server eWay Sample Projects 60
Operations Used in the SQL Server Sample Projects 61
Assigning Operations in JCD 61
Assigning Operations in BPEL 62
About the eInsight Engine and eGate Components 62
Steps Required to Run the Sample Projects 62
Running the SQL Script 63
Importing a Sample Project 63
Building and Deploying the prjSQL Server_BPEL Sample Project 64
Creating a Project 64
Creating the OTDs 64
Creating the Business Process 66
Creating the Business Process Flow 66
Configuring the bpInsert Modeling Elements 67
Configuring the bpUpdate Modeling Elements 70
Configuring the bpDelete Modeling Elements 72
Configuring the bpTableSelect Modeling Elements 73
Configuring the bpPsSelect Modeling Elements 76
Creating the Connectivity Map 80
Populating the Connectivity Map 80
Binding the eWay Components 81
Creating an Environment 82
Configuring the eWays 83
Configuring the eWay Properties 83
Steps required to configure the eWay properties: 83
Steps to Configure the Environment Explorer Properties 84
Configuring the Integration Server 85
Creating the Deployment Profile 86
Creating and Starting the Domain 86
Building and Deploying the Project 87
Deploying a Project to an HP NonStop Server 87
Running the Sample Project 87
Creating the prjSQLServer_JCD Sample Project 88
Creating a Project 88

Creating the OTDs **89**
Creating a Connectivity Map **90**
Populating the Connectivity Map 91
Creating the Collaboration Definitions (Java) **91**
jcdDelete Collaboration 92
jcdInsert Collaboration 92
jcdPsSelect Collaboration 93
jcdTableSelect Collaboration 93
jcdUpdate Collaboration 94
Create the Collaboration Business Rules **94**
Creating the jcdDelete Business Rules 94
Creating the jcdInsert Business Rules 95
Creating the jcdPsSelect Business Rules 97
Creating the jcdTableSelect Business Rules 98
Binding the eWay Components **100**
Creating an Environment **101**
Configuring the eWays **102**
Configuring the eWay Properties 103
Configuring the Environment Explorer Properties 104
Creating the Deployment Profile **105**
Creating and Starting the Domain **106**
Building and Deploying the Project **107**
Deploying a Project to an HP NonStop Server 107
Running the Sample **107**

Appendix A

Common DataType Conversions **109**

Common DataType Conversions **109**

Index **111**

Chapter 1

Introducing the SQL Server eWay

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

What's in This Chapter

- **"About SQL Server" on page 7**
- **"About the SQL Server eWay" on page 7**
- **"What's New in This Release" on page 8**
- **"About this Document" on page 8**
- **"Related Documents" on page 10**
- **"Sun Microsystems, Inc. Web Site" on page 10**
- **"Documentation Feedback" on page 10**

1.1 About SQL Server

SQL Server is Microsoft's® client-server Relational Data Base Management System (RDBMS), used for increased scalability, availability, and security of enterprise data and analytical applications while making them easier to create, deploy, and manage.

1.2 About the SQL Server eWay

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

The SQL Server eWay uses JCDs (Java Collaboration Definitions) and BPEL (Business Process Execution Language to perform the following:

- Query a database
- Automatically generate a graphical user interface (GUI) tree representation of database access objects

- Populate the structure with the actual data values during run time.

The SQL Server eWay also uses the same GUI structure as the rest of the eGate system to describe data flow through the entire enterprise. This feature enables business analysts to define the relationships between a database and relevant applications by dragging and dropping elements between graphical tree structures.

1.3 What's New in This Release

The SQL Server eWay version 5.1.1 includes the following changes and new features:

- **Version Control**: An enhanced version control system allows you to effectively manage changes to the eWay components.
- **Manual Connection** Management: Establishing a connection can now be performed automatically (configured as a property) or manually (using OTD methods from the Java Collaboration).
- **Multiple Drag-and-Drop Component Mapping from the Deployment Editor**: The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- **Support for Runtime LDAP Configuration**: eWay configuration properties now support LDAP key values.
- **MDB Pool Size Support**: Provides greater flow control (throttling) by specifying the maximum and minimum MDB pool size.
- **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.
- **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 Administrator Guide*.

1.4 About this Document

This guide explains how to install, configure, and operate the Composite Application Platform Suite SQL Server Intelligent Adapter, referred to as the SQL Server throughout this guide.

1.4.1 What's in this Document

This document includes the following chapters:

- **Chapter 1 "Introducing the SQL Server eWay"**: Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 "Installing the SQL Server eWay"**: Describes the system requirements and provides instructions for installing the SQL Server.
- **Chapter 3 "Configuring the SQL Server eWay"**: Provides instructions for configuring the eWay to communicate with external SQL Server databases.
- **Chapter 4 "Using the SQL Server eWay Database Wizard"**: Provides instructions for creating Object Type Definitions to be used with the SQL Server.
- **Chapter 5 "Using SQL Server Operations"**: Provides database operations used to access the SQL Server database.
- **Chapter 6 "Implementing SQL Server Sample Projects"**: Provides instructions for installing and running the sample Projects.

1.4.2 Scope

This user's guide provides a description of the SQL Server 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 SQL Server eWay Java methods, see the associated Javadoc.

1.4.3 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning 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.

1.4.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

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

Table 1 Text Conventions (Continued)

Text Convention	Used For	Examples
Blue bold	Hypertext links within document	See **Text Conventions** on page 9
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.4.5 Screenshots

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

1.4.6 Related Documents

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

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

1.5 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

1.6 Documentation Feedback

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

CAPS_docsfeedback@sun.com

Chapter 2

Installing the SQL Server eWay

What's in This Chapter

- "Before You Install" on page 11
- "Installing the SQL Server eWay" on page 11
- "After You Install" on page 15
- "Installing eWay Enterprise Manager Plug-Ins" on page 15

2.1 Before You Install

Open and review the **Readme.txt** file for the SQL Server eWay for any additional information or requirements, prior to installation. The **Readme.txt** file is located on the installation CD-ROM. Refer to the readme for the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

2.2 Installing the SQL Server eWay

The Sun Java Composite Application Platform Installer, a web-based application, is used to select and upload eWays and add-on files during the installation process. The following sections describe how to install the components required for this eWay.

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

2.2.1 Installing the SQL Server eWay on an eGate supported system

Follow the directions for installing the Sun Java Composite Application Platform Suite (CAPS). After you have installed eGate or eInsight, 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 eWays:
   - File eWay (the File eWay is used by most sample Projects)
   - SQL Server eWay

   To upload the SQL Server eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, expand the **Documentation** option and select SQLServereWayDocs.
4. Once you have selected the products to be installed, click **Next**. The Installing Files window appears after the last SAR file has been selected.
5. From the **Installing Files** window, review the product list. If it is correct, Click **Install Products**. The Sun Java Composite Application Platform Installer starts the installation.
6. When your product's installation is completed, click on the prompt, "**When installation completes, click here to continue**."
7. Continue installing the eGate Integrator as instructed in the *Sun Composite Application Platform Suite Installation Guide*.

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

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

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

1. Complete steps 1 through 6 on **Installing the SQL Server eWay on an eGate supported system** on page 12.
2. Open the Enterprise Designer and select **Update Center** from the Tools menu. The Update Center Wizard appears.
3. For Step 1 of the wizard, simply click **Next**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**

7 When prompted, restart the IDE (Integrated Development Environment) to complete the installation. To install the SQL Server eWay Samples and Javadocs.

2.2.2 Extracting the Sample Projects and Javadocs

The SQL Server 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

1 Click the Documentation tab of the Sun Java Composite Application Platform Suite Installer, then click the Add-ons tab.

2 Click the SQL Server eWay Adapter link. Documentation for the SQL Server 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 Sample Projects

1 Click the Documentation tab of the Sun Java Composite Application Platform Suite Installer, then click the Add-ons tab.

2 Click the SQL Server eWay Adapter link. Documentation for the SQL Server eWay appears in the right pane.

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

2.2.3 Configuring the DataDirect 3.5 Drivers for XA Mode

1 Download the **SQLServer eWay JTA (XA) Plug-In (SQLServer_jta.zip)** from Sun Java Composite Application Platform Installer.

2 Extract the **SQLServer_jta.zip** file into a temporary directory.

3 Copy the **sqljdbc.dll** to the Host where you have SQL Server installed.

For the 32-bit version of SQL Server

A Copy the **sqljdbc.dll** from the 32-bit sub-directory.

B To your Microsoft SQL Server\MSSQL\Binn directory located on the SQL Server install.

For the 64-bit version of SQL Server

A Copy the **sqljdbc.dll** from the 64-bit sub-directory.

B To your Microsoft SQL Server\MSSQL\Binn directory located on the SQL Server install.

4 From a command prompt, run the sql script **instjdbc.sql** using isql.

Note: *For more information see the DataDirect Connect JDBC User's Guide and Reference.* http://media.datadirect.com/download/docs/jdbc/jdbcref/jdbcsqlsrv.html#wp967739

Installing Stored Procedures for JTA

To use JDBC distributed transactions through JTA, the system administrator should use the following procedure to install Microsoft SQL Server JDBC XA procedures. Repeat this procedure for any Microsoft SQL Server installation that uses distributed transactions.

To install stored procedures for JTA:

1. Copy the **sqljdbc.dll** file (either the 32-bit or 64-bit version) to the Microsoft SQL Server database server:
   - On a 32-bit platform, copy the **sqljdbc.dll** file from the install_dir/SQLServer JTA/32Bit directory to the SQL_Server_Root/binn directory of the Microsoft SQL Server database server.
   - On a 64-bit platform, copy the **sqljdbc.dll** from the install_dir/SQLServer JTA/64Bit directory to the SQL_Server_Root/binn directory of the Microsoft SQL Server database server.

   where:
     - install_dir is your DataDirect Connect for JDBC installation directory.
     - SQL_Server_Root is your Microsoft SQL Server installation directory.
2. From the database server, use the ISQL utility to run the **instjdbc.sql** script. As a precaution, the system administrator should back up the master database before running **instjdbc.sql**.

   At a command prompt, run **instjdbc.sql** using the following syntax:

   ISQL -Usa -Psa_password -Sserver_name -ilocation**instjdbc.sql**

   where:
   - sa_password is the password of the system administrator.
   - server_name is the name of the server on which the Microsoft SQL Server database resides.
   - location is the full path to instjdbc.sql. This script is located in the install_dir/SQLServer JTA directory where install_dir is your DataDirect Connect for JDBC installation directory.
3. The **instjdbc.sql** script generates many messages. In general, these messages can be ignored; however, the system administrator should scan the output for any messages that may indicate an execution error. The last message should indicate that instjdbc.sql ran successfully. The script fails when there is insufficient space available in the master database to store the JDBC XA procedures or to log changes to existing procedures.

2.3 After You Install

Once the eWay is installed and configured it must then be incorporated into a Project before it can perform its intended functions. See the *Sun SeeBeyond eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

2.4 Installing eWay Enterprise Manager Plug-Ins

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

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

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

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

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

To add plug-ins from the Enterprise Manager

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

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

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

Viewing Alert Codes

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

To View the eWay Alert Codes

1. Add the eWay Enterprise Manager plug-in for this eWay.
2. From the **Enterprise Manager**'s Explorer toolbar, click **configuration**.
3. Click the **Web Applications Manager** tab and go to the **Manage Alert Codes** tab.
4. Browse for and select the Alert Properties File for the application plug-in that you added. The Alert Properties Files are located in the **alertcodes** folder of your Composite Application Platform Suite installation directory.
5. Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of available alert codes is displayed in Table 2.

Table 2 Alert Codes for the SQL Server eWay

Alert Code\Description	Description Details	User Actions
DBCOMMON-CONNECT-FAILED000001=Failed to connect to database {0} on host {1}. Reason: The Pooled connection could not be allocated: [{2}]	Occurs during the initial database connection.	▪ Database is down; start your database. ▪ External configuration information is invalid. You may need to verify the following: ◆ Server name ◆ Database name ◆ User ◆ Password ◆ Port
DBCOMMON-CONNECT-FAILED000002=Operation failed because of a database connection error. Reason: [{0}]	Occurs while retrieving a connection from the database or the pool.	▪ Verify that the database has not terminated with unexpected errors.
DBCOMMON-CONNECT-FAILED000005=Connection handle not usable. Reason:[{0}]	The connection in the pool is stale and 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 Transactions.

Alert Code\Description	Description Details	User Actions
DBCOMMON-XACONNECT-FAILED000001=Failed to connect to database {0} on host {1}. The XA connection could not be allocated: Reason [{2}]	Occurs during the initial database connection.	▪ Check if the database is configured for XA and if the database is running. ▪ External configuration information is invalid. You may need to verify the following: ♦ Server name ♦ Database name ♦ User ♦ Password ♦ Port
DBCOMMON-XASTART-FAILED000001=Unable to perform XAStart for the connection. Reason: [{0}]	A connection error has occurred which caused XASTART to fail.	▪ Check if the database is running, and that 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: *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

Configuring the SQL Server eWay

This chapter describes how to set the properties of the SQL Server eWay.

What's in This Chapter

- **"Creating and Configuring the SQL Server eWay" on page 18**
- **"SQL Server eWay Connectivity Map Properties" on page 23**
- **"SQL Server eWay Environment Configuration Properties" on page 25**
- **"Adding JAR Files to the LogicalHost" on page 33**

3.1 Creating and Configuring the SQL Server eWay

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

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

3.1.1 Selecting SQL Server as the External Application

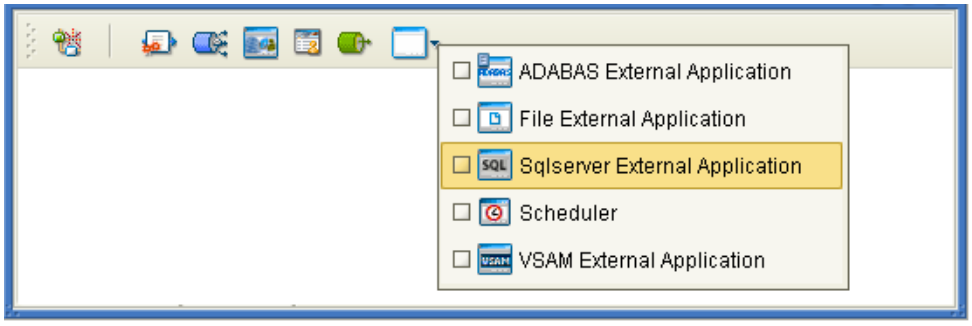
To create a SQL Server eWay you must first create a SQL Server External Application in your Connectivity Map. SQL Server eWays are located between a SQL Server External Application and a Service. Services are containers for Collaborations, Business Processes, eTL processes, and so forth.

To create the SQL Server External Application

1 From the Connectivity Map toolbar, click the **External Applications** icon.

2 Select the **SQL Server External Application** from the menu (see Figure 1). The selected SQL Server External Application icon now appears on the Connectivity Map toolbar.

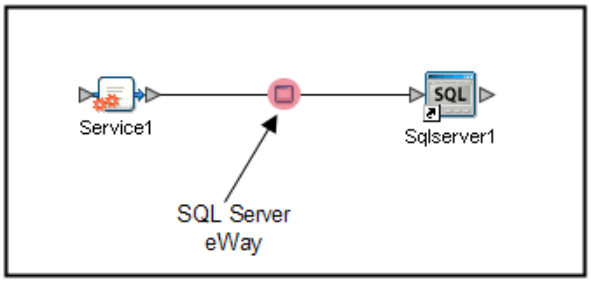
Figure 1 External Application menu



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

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

Figure 2 eWay Location.



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

3.1.2 Configuring the SQL Server eWay Properties

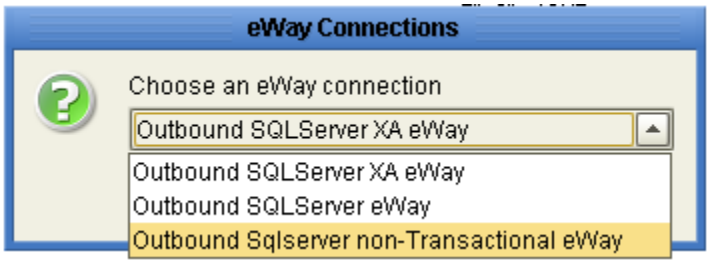
A Project's eWay properties can be modified after the eWay has been established in the Connectivity Map and the Environment has been created.

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

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

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

Figure 3 Template window



The choices to make are as follows:

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

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

Configuring the SQL Server eWay (Environment Explorer) Properties

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

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

3.1.3 Transaction Support Levels Between Different Versions

The types of transaction support levels used in Java CAPS 5.1 may be different from the support levels used in Java CAPS 5.1.1. 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:

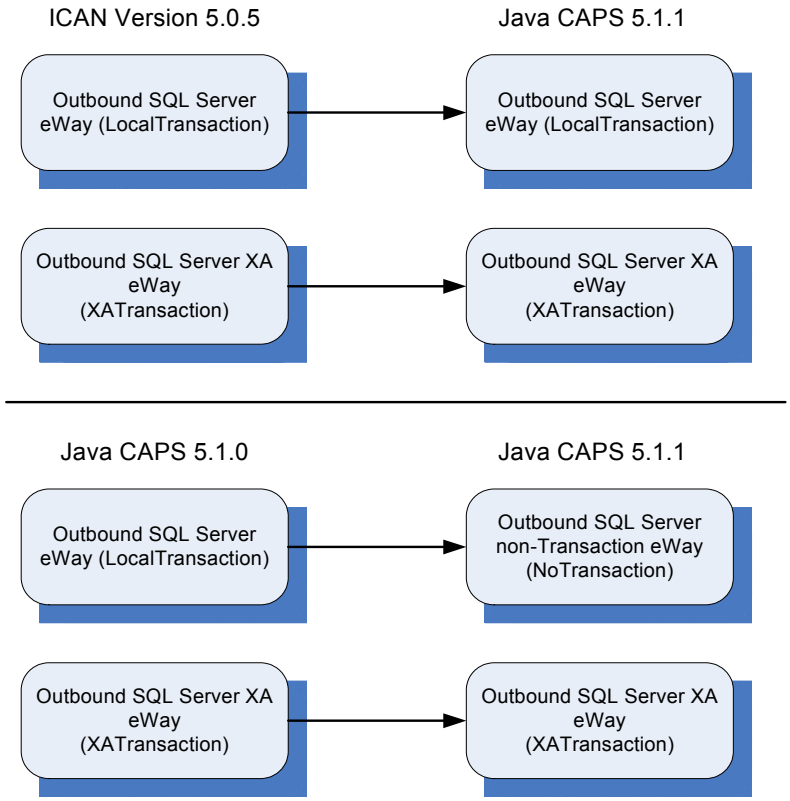
A In the Connectivity Map, delete the link to the database external application, then reconnect the link and select NoTransaction.

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 4 Transaction Support Levels



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

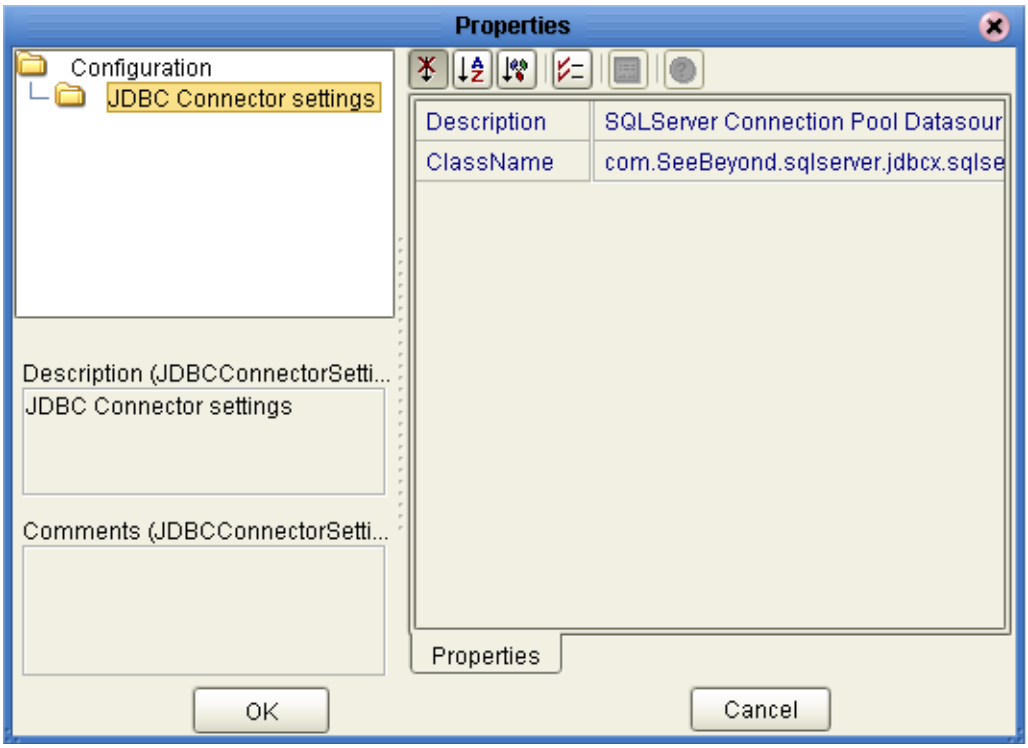
3.1.4 Using the Properties Editor

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

Modifying the Default eWay Configuration Properties

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

Figure 5 Properties Editor -- SQL Server Properties



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

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

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

3.2 SQL Server eWay Connectivity Map Properties

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

- **Outbound Connectivity Map Properties** on page 23
- **Inbound Connectivity Map Properties** on page 24

3.2.1 Outbound Connectivity Map Properties

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

- **Properties in the Outbound eWay** on page 23
- **Properties in the Outbound non-Transactional eWay** on page 24
- **Properties in the Outbound eWay with XA Support** on page 24

Properties in the Outbound eWay

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

Table 3 Outbound Connectivity Map JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **SQLServer Connection Pool Datasource.**
ClassName	Displays the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.	A valid class name. The default is **com.SeeBeyond.sqlserver.jdbcx.sqlserver.SQLServerDataSource**. ***Note:*** Do not change this value.

Properties in the Outbound non-Transactional eWay

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

Table 4 Outbound non-Transactional eWay—JDBC Connector Settings

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

Properties in the Outbound eWay with XA Support

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

Table 5 Outbound with XA Support JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **SQLServer XA Datasource**.
ClassName	Displays the Java class in the JDBC driver that is used to implement the SQLServer XA Datasource interface.	A valid class name. The default is **com.SeeBeyond.sqlserver.jdbcx.sqlserver.SQLServerDataSource**. ***Note:*** Do not change this value.

3.2.2 Inbound Connectivity Map Properties

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

Table 6 Inbound eWay Connectivity Map Parameter Settings

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

Table 6 Inbound eWay Connectivity Map Parameter Settings

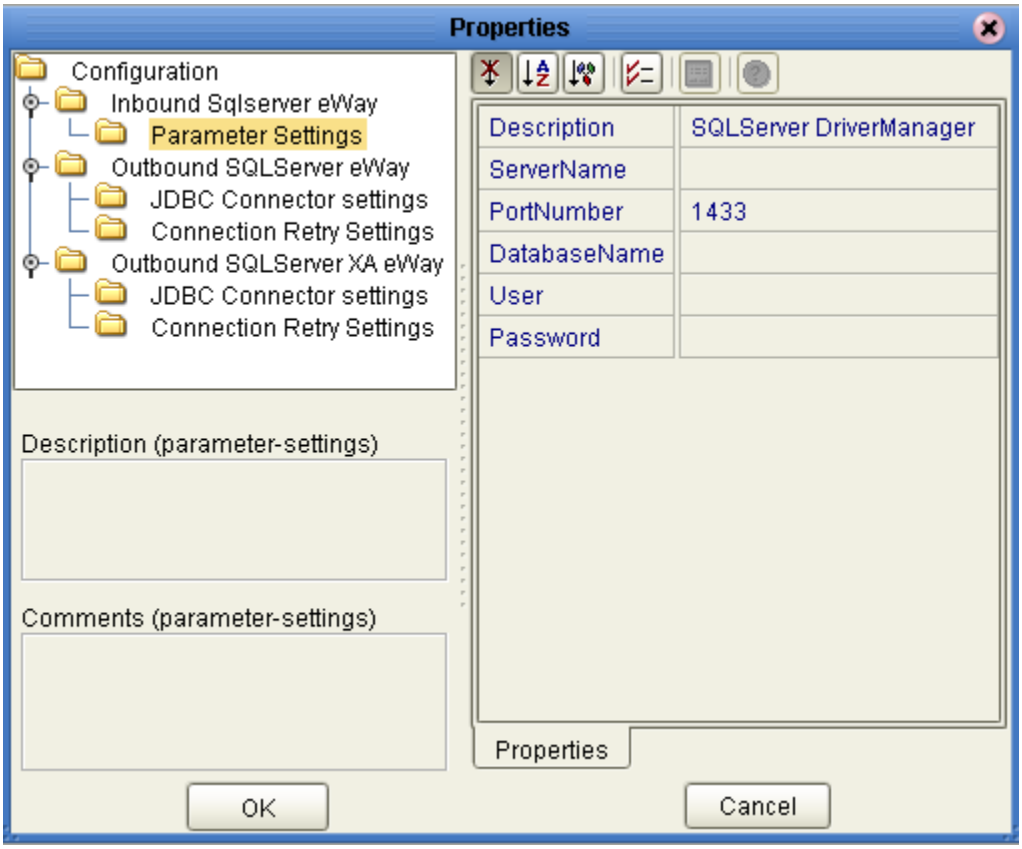
Name	Description	Required Value
PreparedStatement	The Prepared Statement used for polling against the database.	The Prepared Statement must be the same Statement you created using the Database OTD Wizard. Only a SELECT Statement is allowed. Additionally, no place holders should be used. This is a SQL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

3.3 SQL Server eWay Environment Configuration Properties

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

- **Inbound SQL Server eWay Properties** on page 26
- **Outbound SQL Server eWay Properties** on page 27
- **Outbound SQL Server XA eWay Properties** on page 30

Figure 6 SQL Server eWay Environment Configuration



3.3.1 Inbound SQL Server eWay Properties

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

Table 7 Inbound SQL Server eWay Environment Properties

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **SQLServer DriverManager**.
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is 1433.
DatabaseName	Specifies the name of the database instance used on the Server.	Any valid string.
User	Specifies the user name that the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

3.3.2 Outbound SQL Server eWay Properties

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

- **JDBC Connector Settings** on page 27
- **Connection Retry Settings** on page 29

JDBC Connector Settings

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

Table 8 Outbound eWay Environment JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is **SQLServer Connection Pool Datasource**.
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is 1433.
DatabaseName	Specifies the name of the database instance used on the Server.	Any valid string.
User	Specifies the user name that the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

Table 8 Outbound eWay Environment JDBC Connector Settings

Name	Description	Required Value
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.	The delimiter set by the user. For more information, see the Delimiter property below. Valid delimiters are: **"<method-name-1>#<param-1>#<param-2>#.....<param-n>##<method-name-2>#<param-1>#<param-2>#.....<param-n>##.....##".** For example: to execute the method setSpyAttributes, give the method a String for the URL "setSpyAttribute#<url>##". ***Note:*** The setSpyAttributes (for Data Direct drivers) that are contained in the following examples (between the last set of double octothorps [##] within each example), are used for debugging purposes and need not be used on every occasion. Optional—if you are using Spy Log: **"setURL#jdbc:Seebeyond:Sybase://<server>:4100;DatabaseName=<database>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##"**
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**. See the DriverProperties property above for more information on how the default value is used.

Table 8 Outbound eWay Environment JDBC Connector Settings

Name	Description	Required Value
MinPoolSize	Specifies the minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and the new connections should be created as needed. If the pool size is too small, you may experience longer connection times due to the existing number of physical connections. A connection that stays in the pool allows transactions to use it via a logical connection which is faster.	A valid numeric value. The default is **0**.
MaxPoolSize	Specifies the maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size depends on the transaction volume and response time. If the pool size is too big, you may end up with too many connections with the database.	A valid numeric value. The default is **10**.
MaxIdleTime	The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value.

Connection Retry Settings

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

Table 9 Outbound eWay Environment Connection Retry Settings

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

3.3.3 Outbound SQL Server XA eWay Properties

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

- **JDBC Connector Settings (with XA support)** on page 30
- **Connection Retry Settings (with XA support)** on page 32

JDBC Connector Settings (with XA support)

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

Table 10 Outbound SQL Server XA eWay Environment JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default value is **SQLServer XA Datasource**.
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is 1433.
DatabaseName	Specifies the name of the database instance used on the Server.	Any valid string.

Table 10 Outbound SQL Server XA eWay Environment JDBC Connector Settings

Name	Description	Required Value
User	Specifies the user name that the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties	The DataSource implementation may need to execute additional methods to assure a successful run. The additional methods will need to be identified in the Driver Properties.	The delimiter set by the user. For more information, see the Delimiter property below. Valid delimiters are: **"<method-name-1>#<param-1>#<param-2>#……<param-n>##<method-name-2>#<param-1>#<param-2>#……<param-n>##……##".** For example: to execute the method setSpyAttributes, give the method a String for the URL "setSpyAttribute#<url>##". ***Note:*** The setSpyAttributes (for Data Direct drivers) that are contained in the following examples (between the last set of double octothorps [##] within each example), are used for debugging purposes and need not be used on every occasion. Optional—if you are using Spy Log: **"setURL#jdbc:Seebeyond:Sybase://<server>:4100;DatabaseName=<database>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##"**

Table 10 Outbound SQL Server XA eWay Environment JDBC Connector Settings

Name	Description	Required Value
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**. See the DriverProperties property above for more information on how the default value is used.
MinPoolSize	Specifies the minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and the new connections should be created as needed. If the pool size is too small, you may experience longer connection times due to the existing number of physical connections. A connection that stays in the pool allows transactions to use it via a logical connection which is faster.	A valid numeric value. The default is **2**.
MaxPoolSize	Specifies the maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size depends on the transaction volume and response time. If the pool size is too big, you may end up with too many connections with the database.	A valid numeric value. The default is **10**.
MaxIdleTime	The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value.

Connection Retry Settings (with XA support)

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

Table 11 Outbound SQL Server XA eWay Environment Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection with the SQL Server database upon a failure to acquire one.	an integer indicating the number of attempts allowed to establish a connection. The configured default is **0**.

Table 11 Outbound SQL Server XA eWay Environment Connection Retry Settings

Name	Description	Required Value
ConnectionRetryInterval	Specifies the milliseconds of pause before each reattempt to access the database. This setting is used in conjunction with the 'Connection Retries' setting. For example: In the event that the eWay cannot connect to the Database, the eWay will try to reconnect to the database in 5 second intervals, a total of 10 times, when the Connection Retries property is set at 10 and the Connection Retry Interval property is 5000.	A valid numeric value. The default is **1000**.

3.4 Adding JAR Files to the LogicalHost

When using **local:** as the URL value in the Environment properties, both the **ctgclient.jar** and the **ctgserver.jar** files must be installed to the following location prior to running a SQL Server eWay Project:

```
<caps51>\logicalhost\is\lib
```

where <*caps51*> is the directory where the Composite Application Platform Suite is installed.

Note: *If the Logical Host is running, it must be recycled to pickup the new JAR file.*

Chapter 4

Using the SQL Server eWay Database Wizard

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

What's in This Chapter

- **"Using the Database OTD Wizard" on page 34**
- **"Steps to Create a New SQL Server OTD" on page 35**
- **"Editing Existing OTDs" on page 47**

4.1 Using the Database OTD Wizard

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

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

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

4.2 Steps to Create a New SQL Server OTD

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

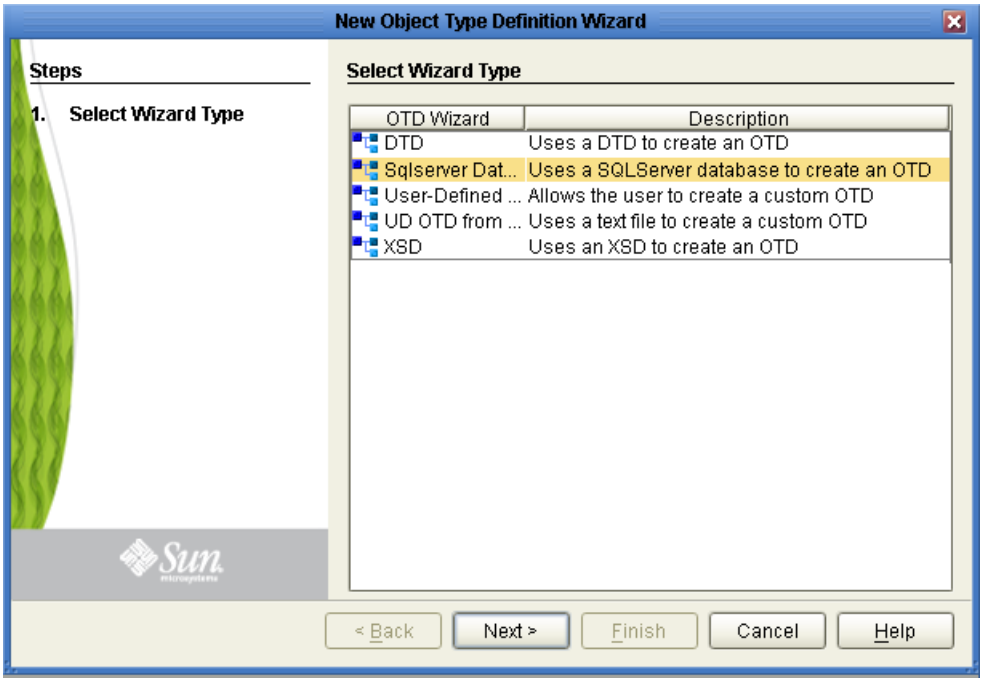
- **Select Wizard Type** on page 35
- **Connect to Database** on page 36
- **Select Database Objects** on page 36
- **Select Procedures** on page 40
- **Add Prepared Statements** on page 43
- **Specify the OTD Name** on page 46
- **Review Selections** on page 46

Select Wizard Type

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

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

Figure 7 OTD Wizard Selection



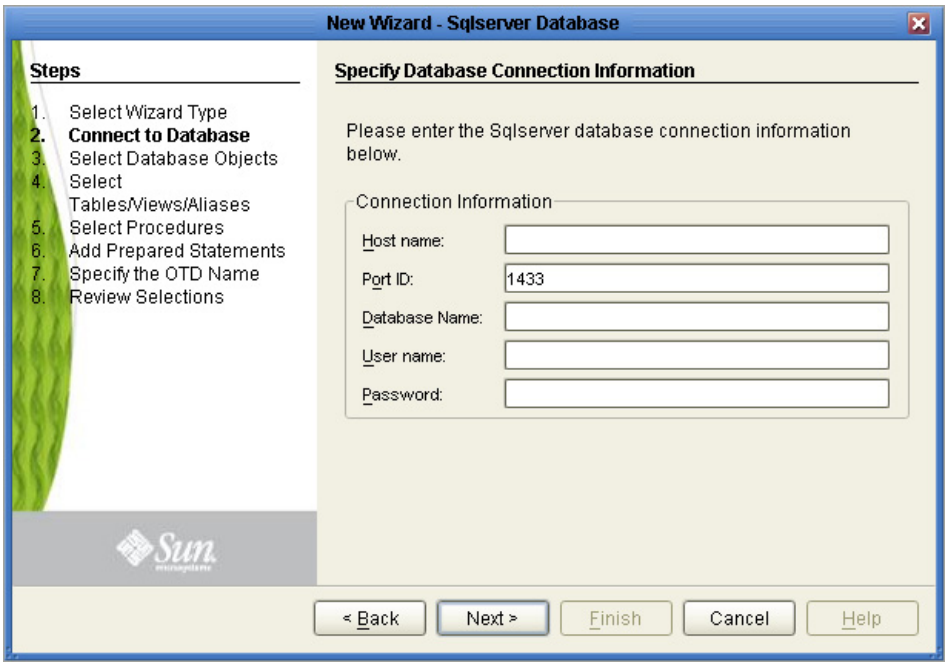
4.2.1 Connect to Database

Enter the SQL Server database connection information in the Connection Information frame.

Required Database Connection Fields include:

- Host name – the database service host name.
- Port ID – the database service connection port ID/number.
- Database name – the name of the SQL Server database.
- User name – a valid SQL Server database username.
- Password – a password for the user name noted above.

Figure 8 Database Connection Information



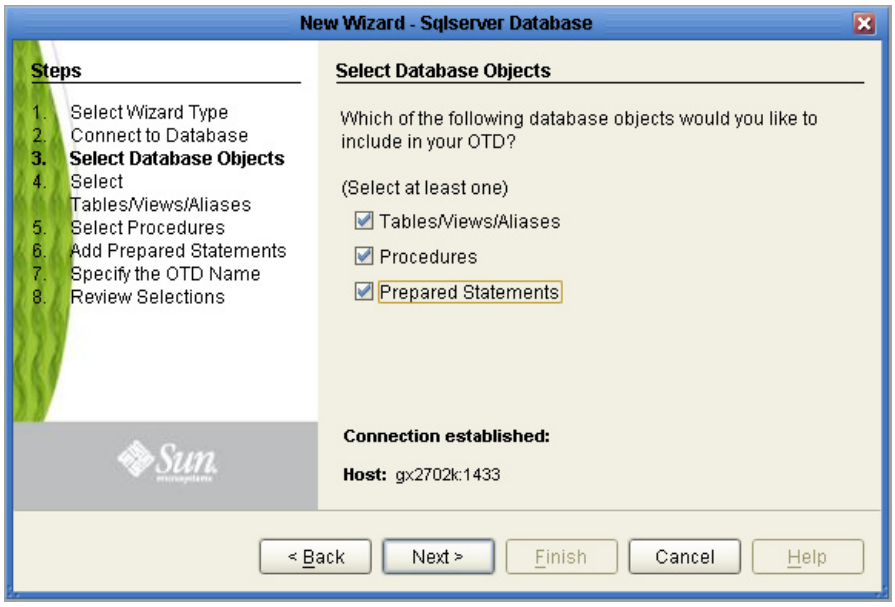
Click **Next**. The **Select Database Objects** dialog box appears

4.2.2 Select Database Objects

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 9).

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

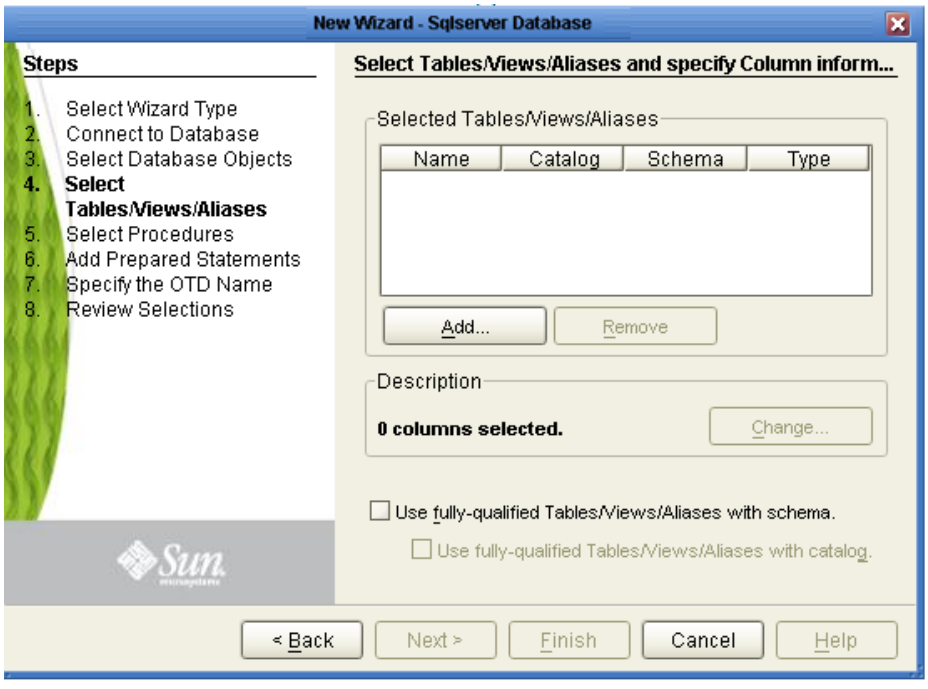
Figure 9 Select Database Objects



4.2.3 Select Table/Views/Aliases

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

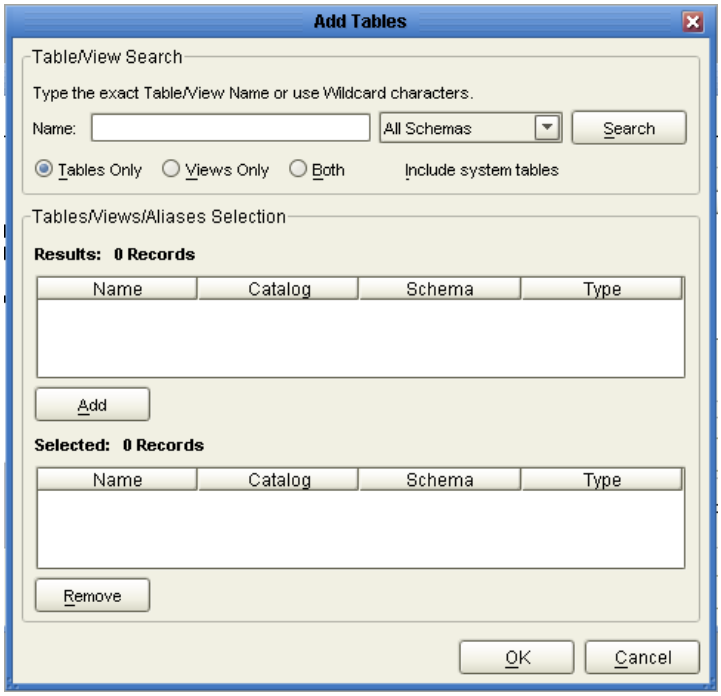
Figure 10 Select Tables/Views/Aliases



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

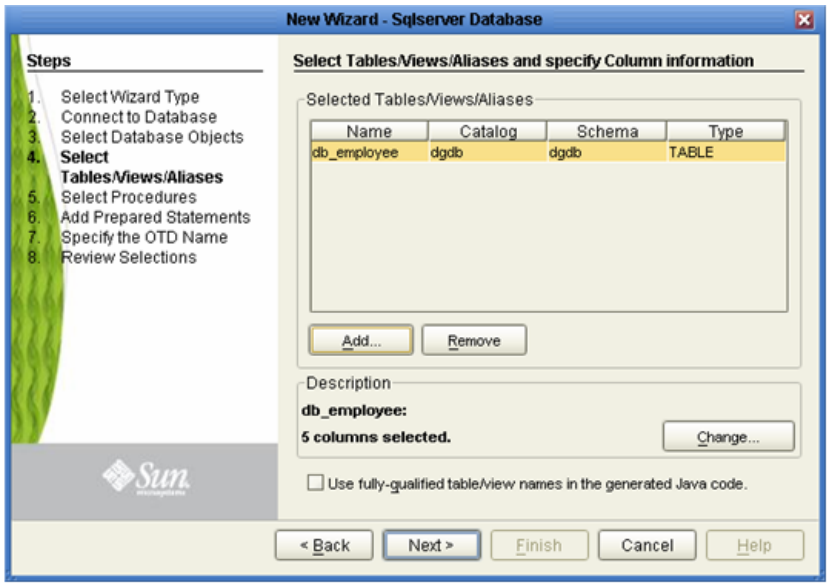
3 From the **Table/View Name** drop down list, select the location of your database table and click **Search** (see Figure 11).

Figure 11 Database Wizard - All Schemes



4 Select the table of choice and click **OK**. The table selected is added to the **Selected Tables/Views** window (see Figure 12).

Figure 12 Selected Tables/Views window with a table selected

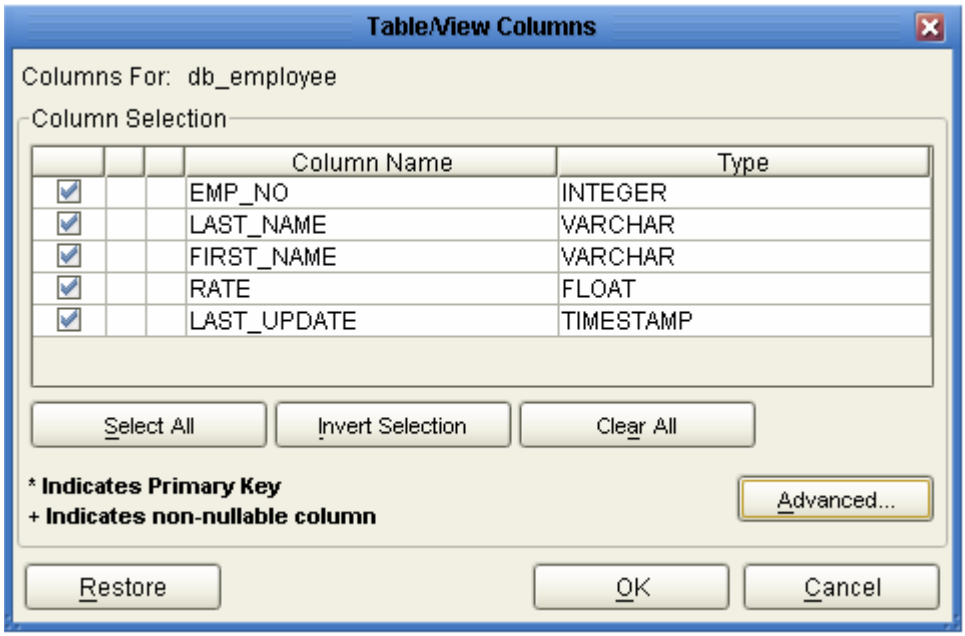


5 In the **Selected Tables/Views** window, review the table(s) you have selected. To make changes to the selected Table or View, click **Change**. If you do not wish to make any additional changes, click **Next** to continue.

6 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 13).

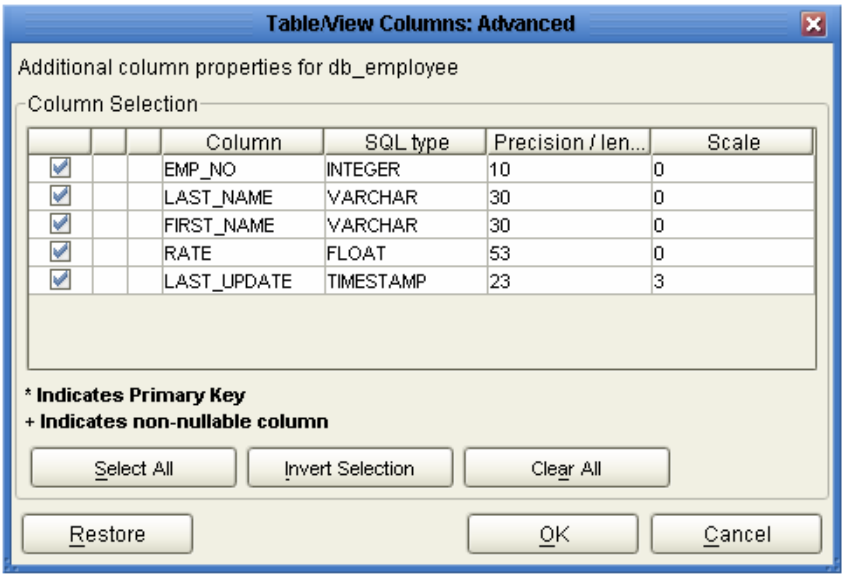
The data type is usually listed as **Other** when the driver cannot detect the data type. In these situations we recommend changing the data type to one that is more appropriate for the type of column data.

Figure 13 Table/View Columns



7 Click **Advanced** to change the data type, precision/length, or scale. Once you have finished your table choices, click **OK**. In general, you will not need to change these settings (see **Figure 14 on page 40**).

Figure 14 Table/View Columns — Advanced

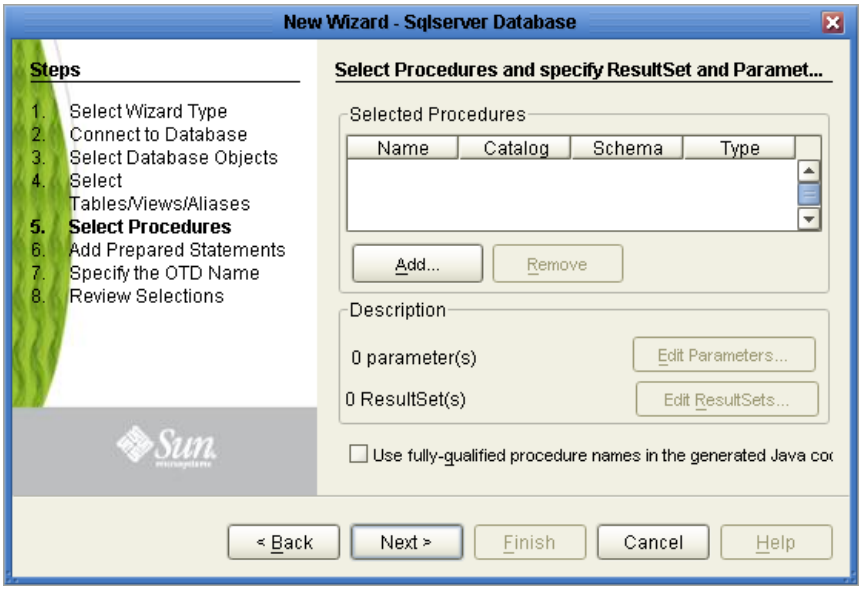


8 When using Prepared Statement packages, select **Use fully qualified table/view names in the generated Java code** (see Figure 12).

4.2.4 Select Procedures

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

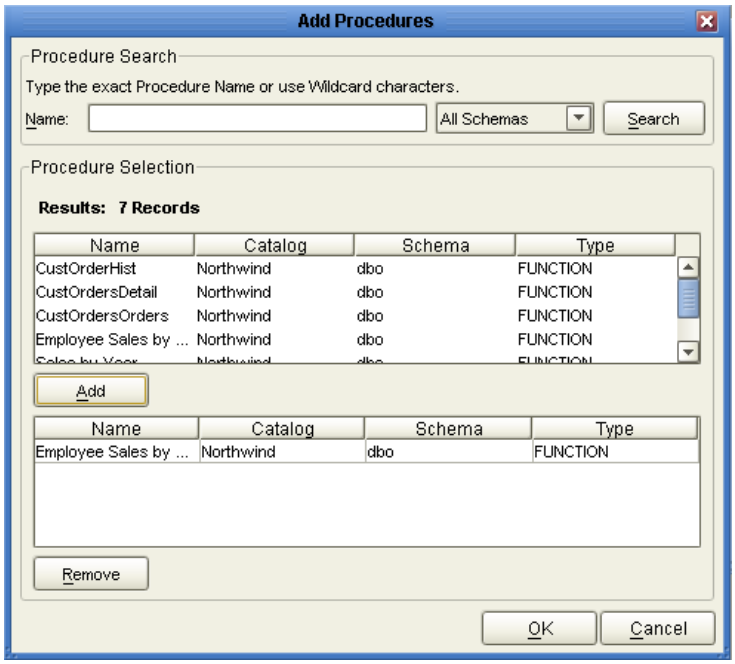
Figure 15 Select Procedures and specify Resultset and Parameter Information



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

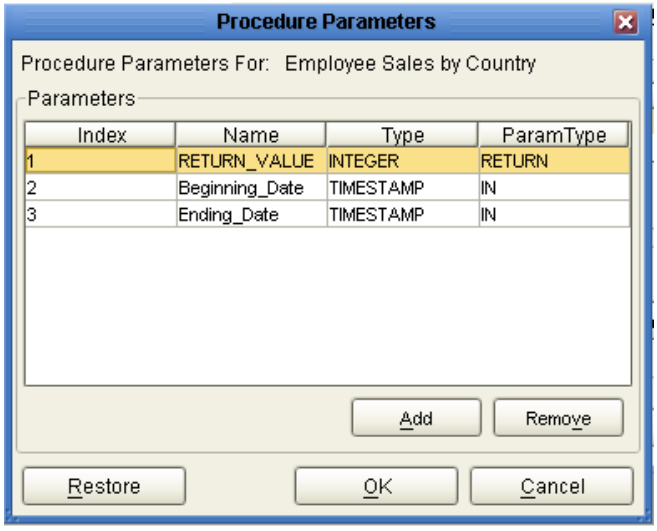
3 In the resulting **Procedure Selection** list box, select a Procedure. Click **OK**.

Figure 16 Add Procedures



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 17).

Figure 17 Procedure Parameters

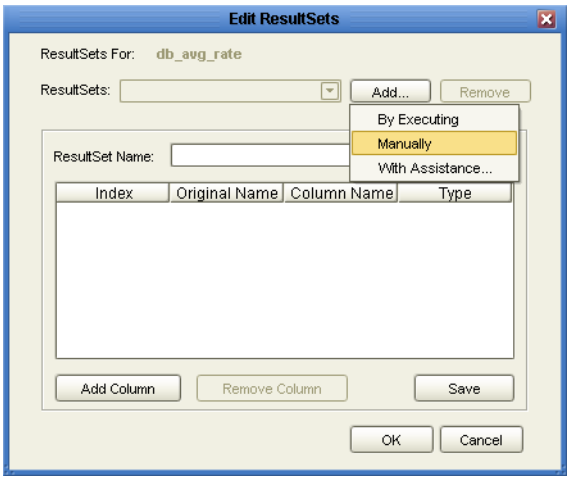


5 To restore the data type, click **Restore**. When finished, click **OK**.

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 18 Edit Resultset



The DBWizard provides thre 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.

4.2.5 Add Prepared Statements

Add a Prepared Statement object to your OTD.

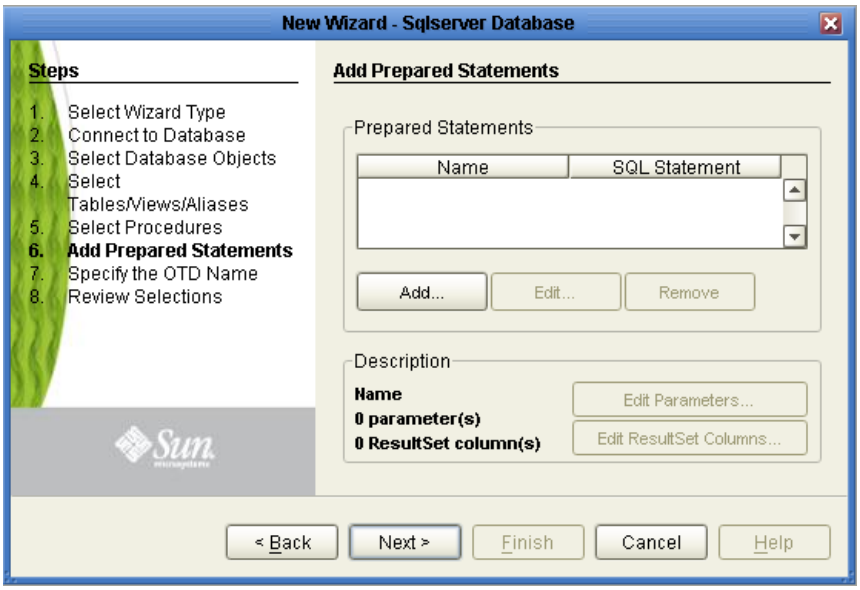
Steps Required to Add Prepared Statements Include:

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.*

You can process a resultset by looping through the next() method. For more information, see **The Query (Select) Operation** *on page 51.*

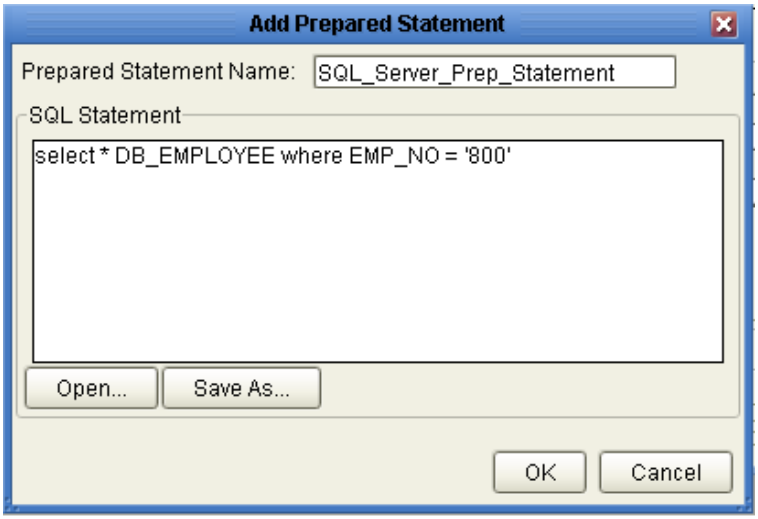
1 On the **Add Prepared Statements** window, click **Add**.

Figure 19 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 20).

Figure 20 Prepared SQL Statement

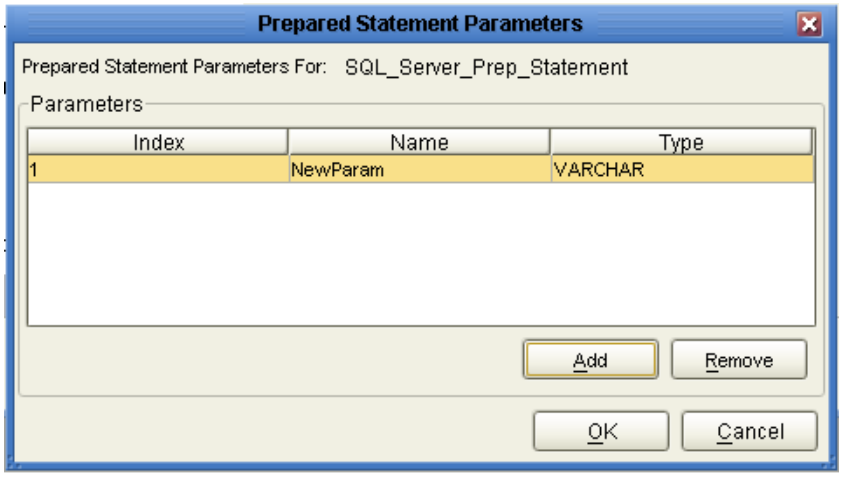


3 The **Add Prepared Statement** window displays the name you assigned to the Prepared Statement. To edit the parameters, click **Edit Parameters**. To change the data type, click in the **Type** field and select a different type from the list.

Note: *When doing a Prepared Statement with two or more tables, where multiple tables have the same column name, you must put the table name qualifier in the Prepared Statement to build the OTD.*

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** (see Figure 21).

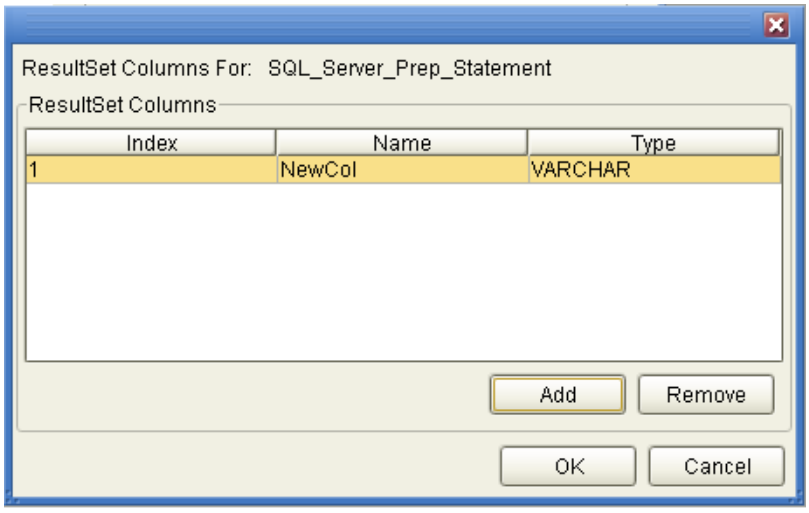
Figure 21 Edit the Prepared Statement Parameters



5 To edit the Resultset Columns, click **Edit Resultset Columns**. Both the Name and Type are editable but it is recommend you do not change the Name. Doing so will cause a loss of integrity between the Resultset and the Database. Click **OK** (see Figure 22).

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

Figure 22 ResultSet Columns



6 On the Add Prepared Statements window, click **OK**.

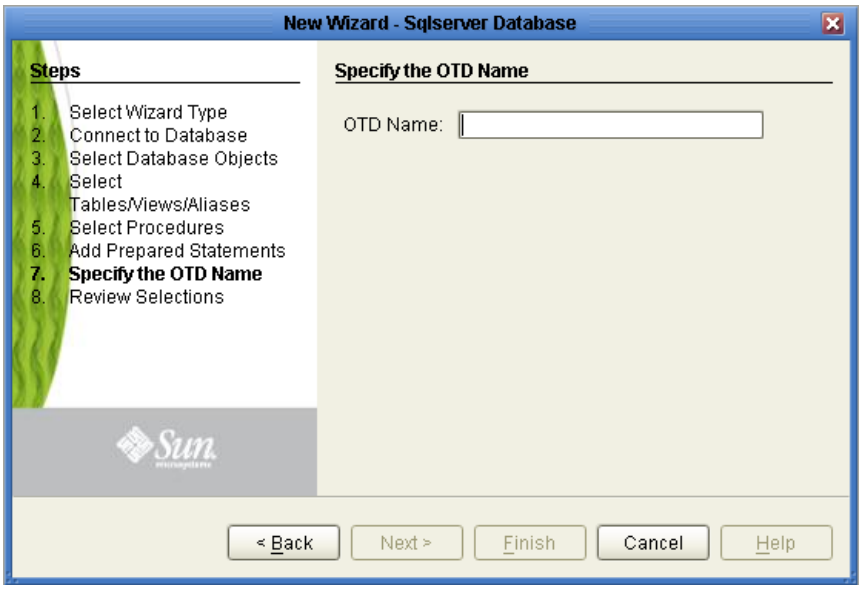
4.2.6 Specify the OTD Name

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 23).

Figure 23 Naming an OTD



2 Click **Next**.

4.2.7 Review Selections

Review the selections made for the new OTD.

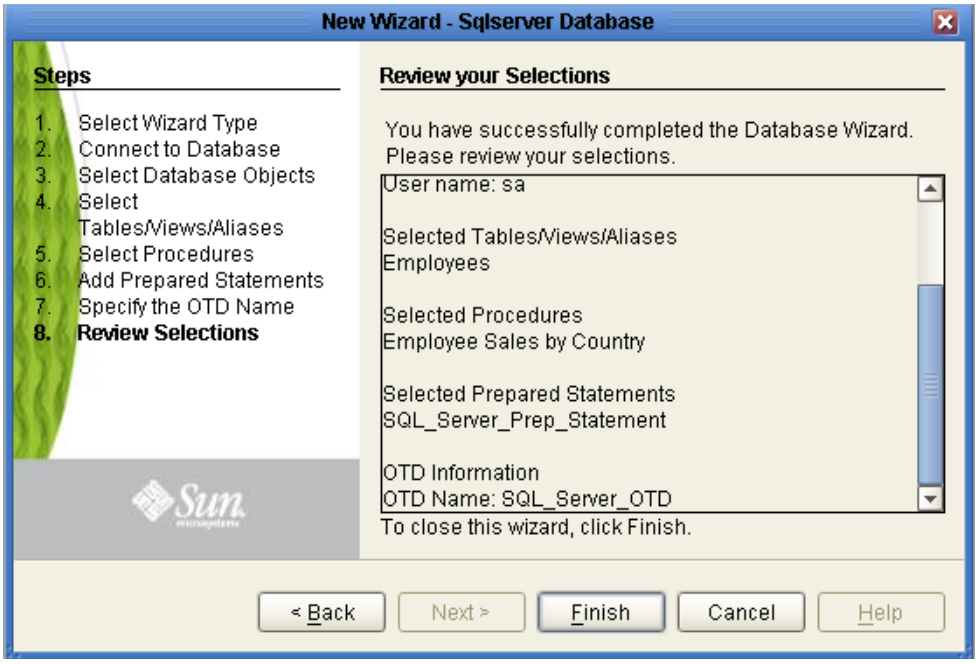
Steps Required to Review Your OTD Selections:

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 24 on page 47**.

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

Figure 24 Database Wizard - Summary



4.3 Editing Existing OTDs

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

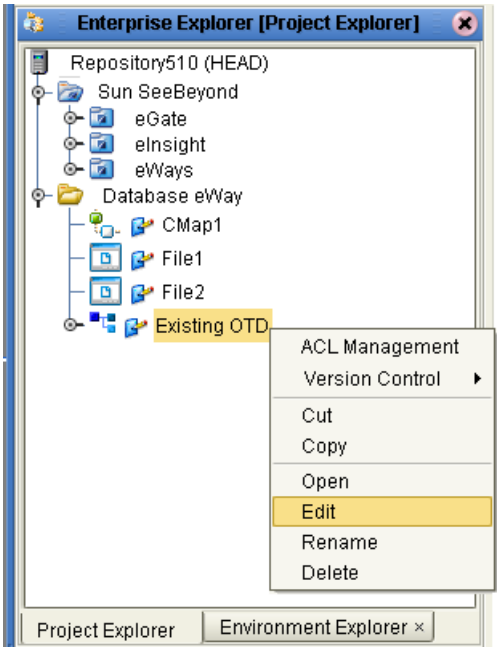
- Add or Remove **Table/Views/Aliases**.
- Change data types by selecting a different one from a list.
- Add or Remove columns from a **Table** object.
- Add or Remove **Prepared Statement** objects.
- Edit **Prepared Statement** objects.
- Add or Remove **Stored Procedure** objects.
- Edit **Stored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

To Edit an Existing OTD

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

1 In the Enterprise Explorer, right-click on the OTD. From the submenu, click **Edit** (see Figure 25). The Database Connection Information Wizard opens.

Figure 25 OTD Edit Menu Item



2 Connect to the database by entering the applicable information in the wizard. Once the connection is established, the Database Wizard opens, allowing you to make modifications to the OTD.

3 Once you have completed editing the OTD, click the **Finish** button to save the changes.

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

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

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

Chapter 5

Using SQL Server Operations

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

What's in This Chapter

- **SQL Server eWay Database Operations (BPEL)** on page 49
- **SQL Server eWay Database Operations (JCD)** on page 51

5.1 SQL Server eWay Database Operations (BPEL)

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

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

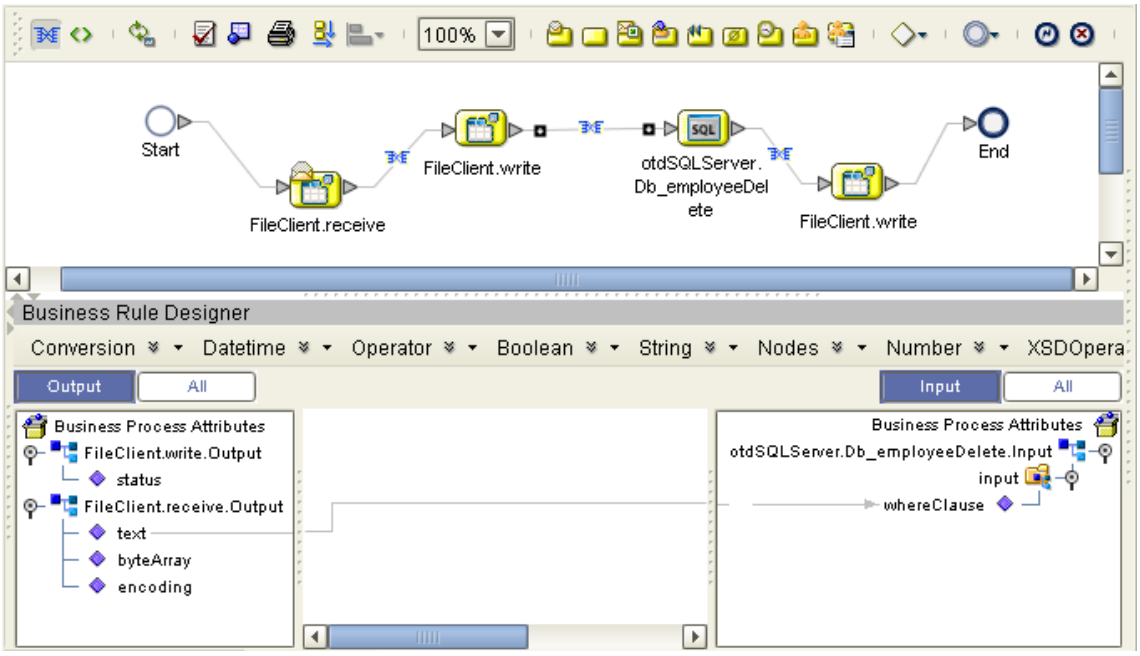
In addition to these outbound operations, the SQL Server eWay also employs the inbound Activity **ReceiveOne** within a Prepared Statement OTD.

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 26 displays the business rules between the **FileClient.write** and **otdSQL Server.Db_employeeDelete** Activities. In this example, the **whereClause** appears on the Input side.

Figure 26 Input and Output Between Activities



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

Table 12 SQL Server Operations

eInsight Operation	Activity Input	Activity Output
SelectAll	where() clause (optional)	Returns all rows that fit the condition of the where() clause.
SelectMultiple	number of rows where() clause (optional)	Returns the number of rows specified that fit the condition of the where() clause, and the number of rows to be 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 SQL Server eWay Database Operations (JCD)

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

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

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

Note: *Refer to the Javadoc for a full description of methods included in the SQL Server 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 prjSQLServer_JCDjcdALL;


public class jcdTableSelect
{

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

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

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

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

    public void receive(
com.stc.connector.appconn.file.FileTextMessage input,
dtd.otdOutputDTD1325973702.DB_Employee otdOutputDTD_DB_Employee_1,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
       FileClient_1.setText( "Selectiong records from db_employee
table via Table Select........" );
        FileClient_1.write();
        otdSQLServer_1.getDb_employee().select( input.getText() );
        while (otdSQLServer_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.shortToString(
otdSQLServer_1.getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdSQLServer_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdSQLServer_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate(
otdSQLServer_1.getDb_employee().getRATE().toString() );
            otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdSQLServer_1.getDb_employee().getLAST_UPDATE(), "yyyy-MM-dd
hh:mm:ss", false, "" ) );
            FileClient_1.setText(
otdOutputDTD_DB_Employee_1.marshalToString() );
            FileClient_1.write();
        }
        FileClient_1.setText( "Table Select Done." );
        FileClient_1.write();
    }

}
```

The Insert Operation

To perform an insert operation on a table

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

   This example inserts an employee record.

```
package prjSQLServer_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,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1,
dtd.otdInputDTD_1206505729.DB_Employee otdInputDTD_DB_Employee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Inserting records in to db_employee
table......" );
        FileClient_1.write();
        otdInputDTD_DB_Employee_1.unmarshalFromString(
input.getText() );
        otdSQLServer_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdSQLServer_1.getDb_employee().setEMP_NO(
typeConverter.stringToShort(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdSQLServer_1.getDb_employee().setLAST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdSQLServer_1.getDb_employee().setFIRST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdSQLServer_1.getDb_employee().setRATE( new
java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1
).getRate() ) );
            otdSQLServer_1.getDb_employee().setLAST_UPDATE(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdSQLServer_1.getDb_employee().insertRow();
        }
        FileClient_1.setText( "Insert Done." );
        FileClient_1.write();
    }

}
```

The Update Operation

To perform an update operation on a table

1. Execute the **update()** method.
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 prjSQLServer_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,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Updating the Rate and Last_update
fields .. " );
        FileClient_1.write();
        otdSQLServer_1.getDb_employee().update( input.getText() );
        while (otdSQLServer_1.getDb_employee().next()) {
         otdSQLServer_1.getDb_employee().setLAST_NAME( "Krishna" );
         otdSQLServer_1.getDb_employee().setFIRST_NAME( "Kishore" );
         otdSQLServer_1.getDb_employee().updateRow();
         }
        FileClient_1.setText( "Update Done." );
        FileClient_1.write();
    }

}
```

The Delete Operation

To perform a delete operation on a table

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

In this example DELETE an employee.

```
package prjSQLServer_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,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Deleting record....." );
        FileClient_1.write();
        otdSQLServer_1.getDb_employee().delete( input.getText() );
        FileClient_1.setText( "Delete Done." );
        FileClient_1.write();
    }
}
```

5.2.2 The Stored Procedure

A Stored Procedure OTD represents a database stored procedure. Fields correspond to the arguments of a stored procedure while methods are the operations that you can apply to the OTD. It allows you to execute a stored procedure. 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.

Below are the steps for executing the Stored Procedure:

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

For example:

```
package Storedprocedure;

public class sp_jce
{

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

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

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

        //@map:Copy java.lang.Integer.parseInt(Employee_no) to
Employee_no
        insert_DB_1.getInsert_new_employee().setEmployee_no(
java.lang.Integer.parseInt(
employeedb_with_top_db_employee_1.getEmployee_no() ) );

        //@map:Copy Employee_lname to Employee_Lname
        insert_DB_1.getInsert_new_employee().setEmployee_Lname(
employeedb_with_top_db_employee_1.getEmployee_lname() );

        //@map:Copy Employee_fname to Employee_Fname
```

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

        //@map:Copy java.lang.Float.parseFloat(Rate) to Rate
        insert_DB_1.getInsert_new_employee().setRate(
java.lang.Float.parseFloat(
employeedb_with_top_db_employee_1.getRate() ) );

        //@map:Copy java.sql.Timestamp.valueOf(Update_date) to
Update_date
        insert_DB_1.getInsert_new_employee().setUpdate_date(
java.sql.Timestamp.valueOf(
employeedb_with_top_db_employee_1.getUpdate_date() ) );

        //@map:Insert_new_employee.execute
        insert_DB_1.getInsert_new_employee().execute();

        //@map:insert_DB_1.commit
        insert_DB_1.commit();

        //@map:Copy "procedure executed" to Text
        FileClient_1.setText( "procedure executed" );

        //@map:FileClient_1.write
        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

SQL Server 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 XSC 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 Result Set (ResultSet_1) and work with the data until you have completed your modifications and updates. Open ResultSet_2, (ResultSet_1 is now closed) and modify. When you have completed your work in ResultSet_2, open any additional ResultSets or close ResultSet_2.
- If you modify the ResultSet generated by the Execute mode of the Database Wizard, you need to 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.

```
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 SQL Server Sample Projects

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

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

What's in This Chapter

- **"About the SQL Server eWay Sample Projects" on page 60**
- **"Steps Required to Run the Sample Projects" on page 62**
- **"Running the SQL Script" on page 63**
- **"Importing a Sample Project" on page 63**
- **"Building and Deploying the prjSQL Server_BPEL Sample Project" on page 64**
- **"Creating the prjSQLServer_JCD Sample Project" on page 88**

6.1 About the SQL Server eWay Sample Projects

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

- **prjSQLServer_JCD:** demonstrates how to select, insert, update, and delete data from a SQL Server database using JCDs.
- **prjSQLServer_BPEL:** demonstrates how to select, insert, update, and delete data from a SQL Server database using a BPEL business process.

Both the **prjSQLServer_JCD** and **prjSQLServer_BPEL** sample Projects demonstrate how to:

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

- Delete an employee record in the db_employee table.

In addition to sample Projects, the **SQLServer510_SAMPLE_projects.zip** file also includes six sample input trigger files and ten sample output files (five per sample).

Sample input files include:

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

Sample output JCD files include:

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

Sample output BPEL files include:

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

6.1.1 Operations Used in the SQL Server Sample Projects

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

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

Assigning Operations in JCD

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

1. Create a Collaboration that contains a database OTD created from the SQL Server database.

2 Right-click the OTD listed in your Collaboration and then select **Select Method to Call** from the shortcut menu.

3 Browse to and select a method to call.

Assigning Operations in BPEL

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

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

2 Drag the operation onto the eInsight Business Process canvas.

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

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

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

6.1.2 About the eInsight Engine and eGate Components

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

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

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

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

6.2 Steps Required to Run the Sample Projects

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

1 Run the SQL script.

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

2 Import the sample Projects.

3 Build, deploy, and run the sample Projects.

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

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

4 Check the output.

6.3 Running the SQL Script

The data used for both the **JCD** and **BPEL** sample Projects are contained within a table called **db_employee**. You create this table by using the SQL statement **SQLServer_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 SQL Server eWay use input files to pass predefined data or conditions into the Collaboration or BPEL business process, which then transforms the database contents, and delivers the result set.

6.4 Importing a Sample Project

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

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

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

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

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

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

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

6.5 Building and Deploying the prjSQL Server_BPEL Sample Project

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

Steps required to create the sample project include:

- **Creating a Project** on page 64
- **Creating the OTDs** on page 64
- **Creating the Business Process** on page 66
- **Creating the Connectivity Map** on page 80
- **Creating an Environment** on page 82
- **Configuring the eWays** on page 83
- **Creating the Deployment Profile** on page 86
- **Creating and Starting the Domain** on page 86
- **Building and Deploying the Project** on page 87

6.5.1 Creating a Project

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

1 Start the Enterprise Designer.

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

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

6.5.2 Creating the OTDs

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

- SQL Server Database OTD
- Inbound DTD OTD

- Outbound DTD OTD

Steps required to create a SQL Server 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 **SQL Server Database OTD Wizard** from the list of OTD Wizards and click **Next**.
3. Enter the connection information for the SQL Server database. Connection fields include:
   - Host name:
   - Port ID:
   - 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 SQL Server 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, we use **otdSQL Server**.
11. Click **Next** and review your settings, then click **Finish** to create the OTD.

Steps required to create inbound and outbound DTD OTDs include:

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

The New Object Type Definition Wizard window appears.

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

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

- otdInputDTD.dtd
- otdOutputDTD.dtd

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

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

6.5.3 Creating the Business Process

Steps required to create the Business Process include:

- Creating the business process flow
- Configuring the modeling elements

Creating the Business Process Flow

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

Steps to create a business process flow include:

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

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

- bpUpdate
- bpDelete
- bpPsSelect
- bpTableSelect

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

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

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

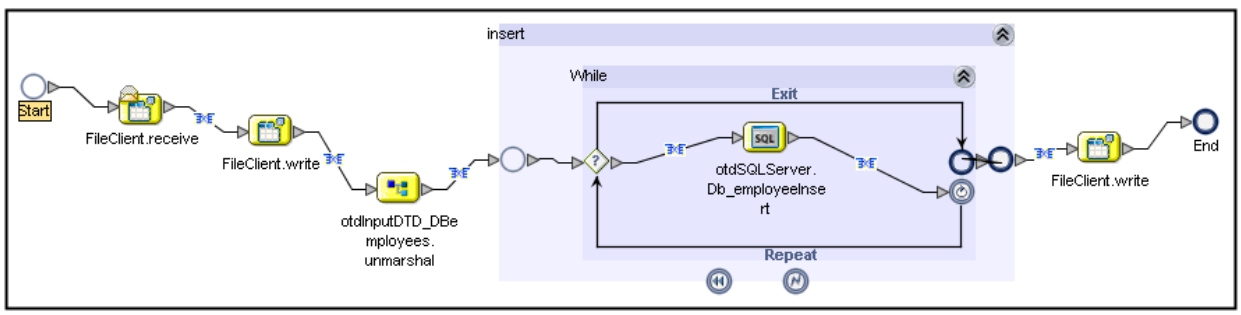
Configuring the bpInsert Modeling Elements

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

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Insert operation. See Figure 27 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 27 bpInsert Business Process

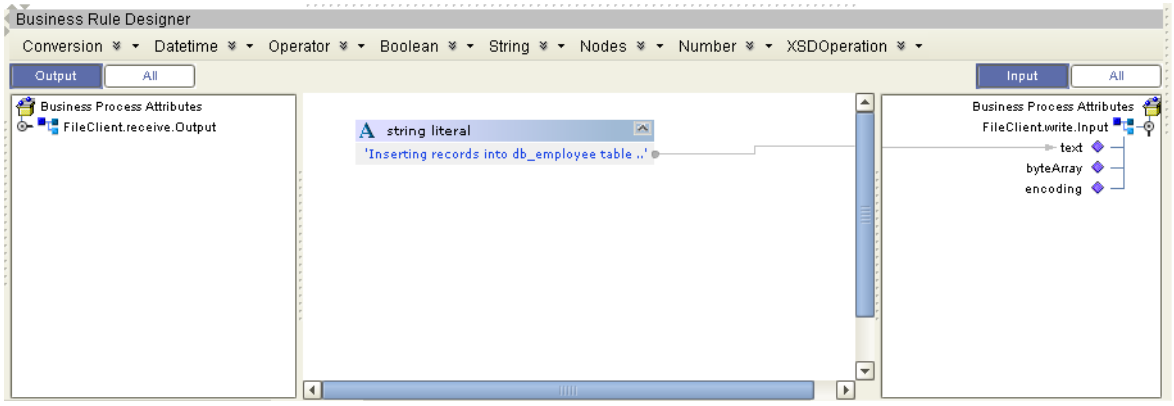


Steps required to configure the bpInsert business process:

Configure the following business rules in the bpInsert business process.

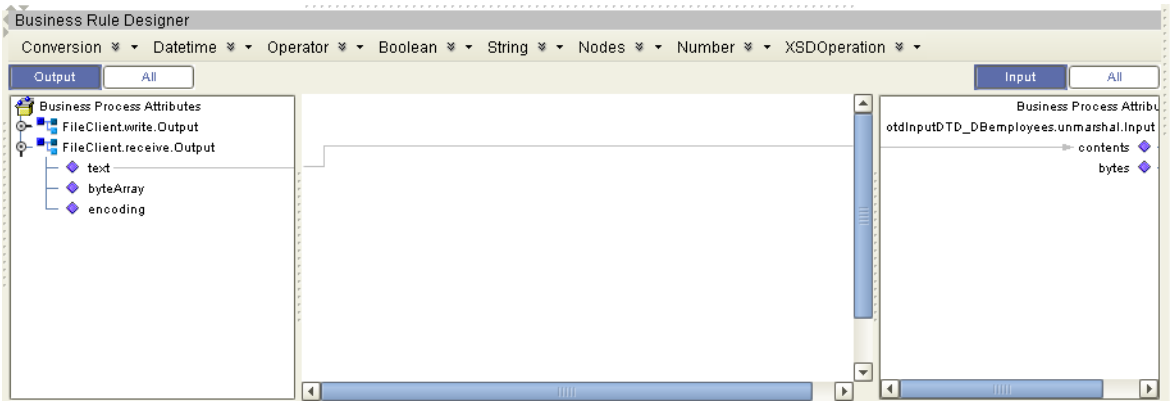
1 Configure the business rule between **FileClient.receive** and **FileClient.write** as seen in Figure 28.

Figure 28 bpInsert Business Rule # 1



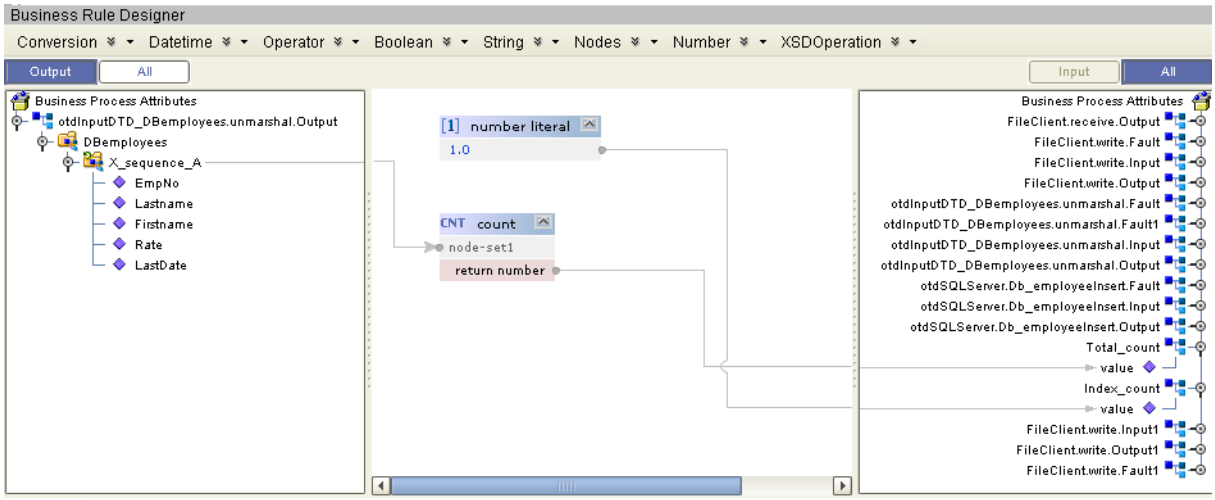
2 Configure the business rule between the **FileClient.write** Activity and **otdInputDTD_DBemployees.unmarshal** Activity as seen in Figure 29.

Figure 29 bpInsert Business Rule # 2



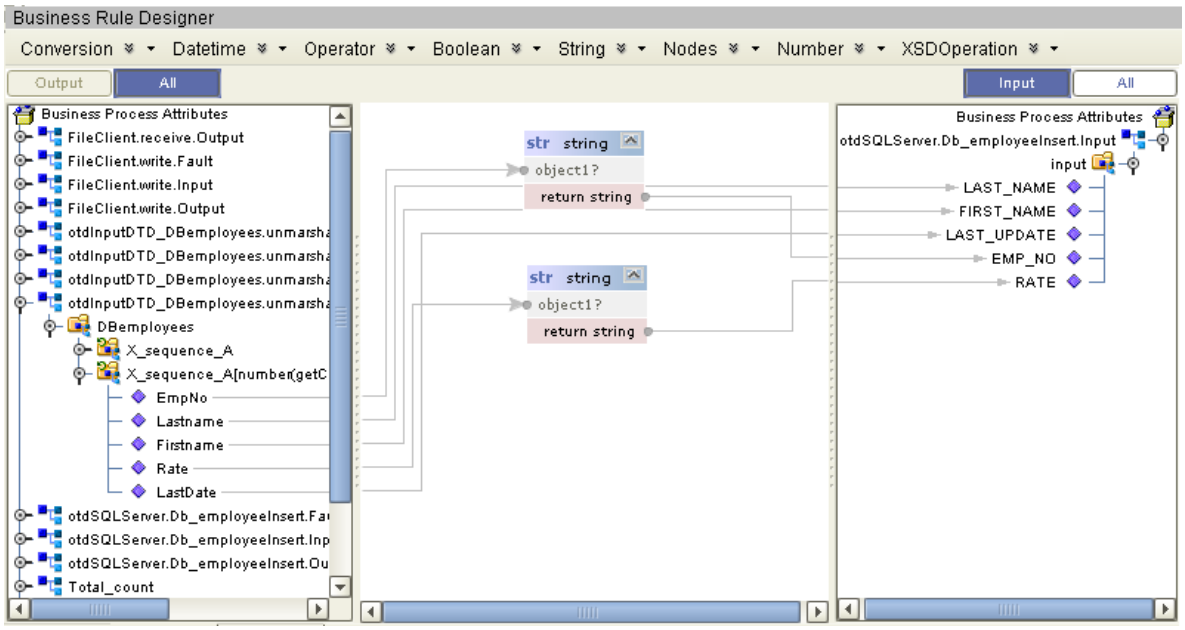
3 Configure the business rule between **otdInputDTD_DBemployees.unmarshal** and the **Insert** (Scope element).

Figure 30 bpInsert Business Rule # 3



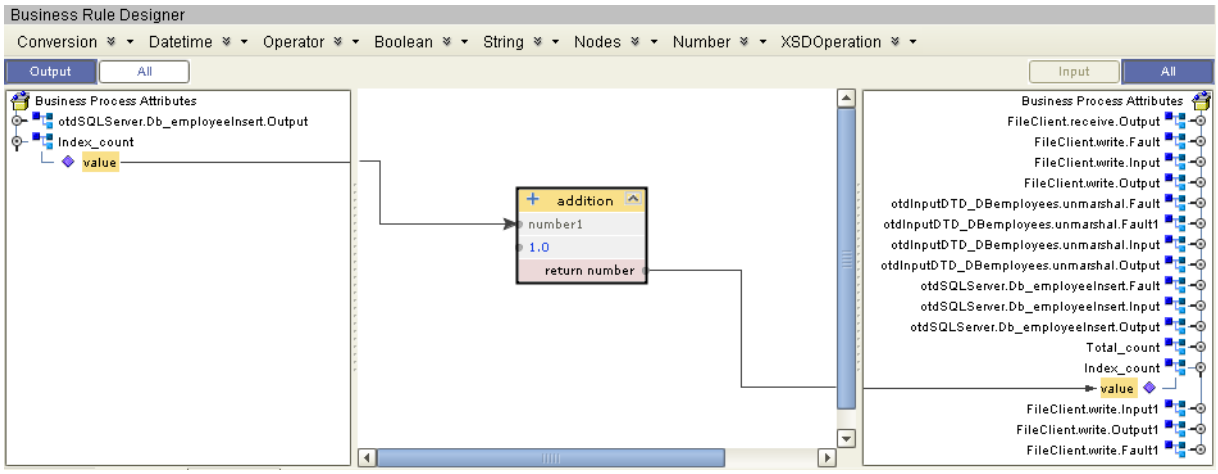
4 Configure the business rule in the **While** statement that connects to the **otdSQL Server.DB_EMPLOYEEInsert** Activity.

Figure 31 bpInsert Business Rule # 4



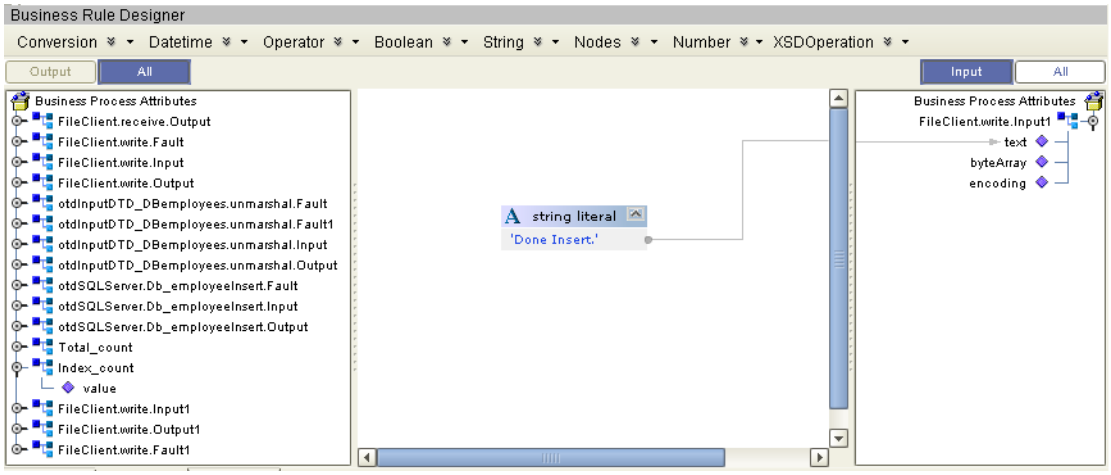
5 Configure the business rule in the **While** statement that connects from the **otdSQL Server.DB_EMPLOYEEInsert** Activity.

Figure 32 bpInsert Business Rule # 5



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

Figure 33 bpInsert Business Rule # 6



Configuring the bpUpdate Modeling Elements

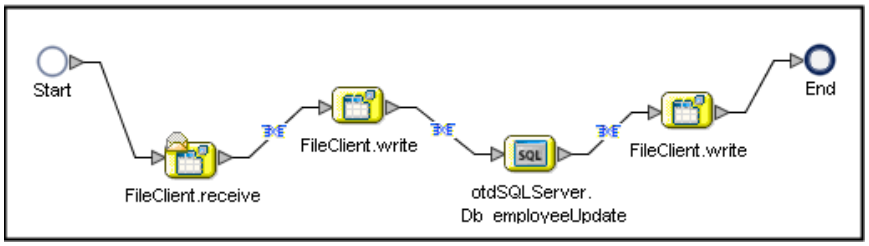
The bpUpdate business process describes how to update a record in the SQL Server database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Update operation. Figure 34 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 34 bpUpdate Business Process

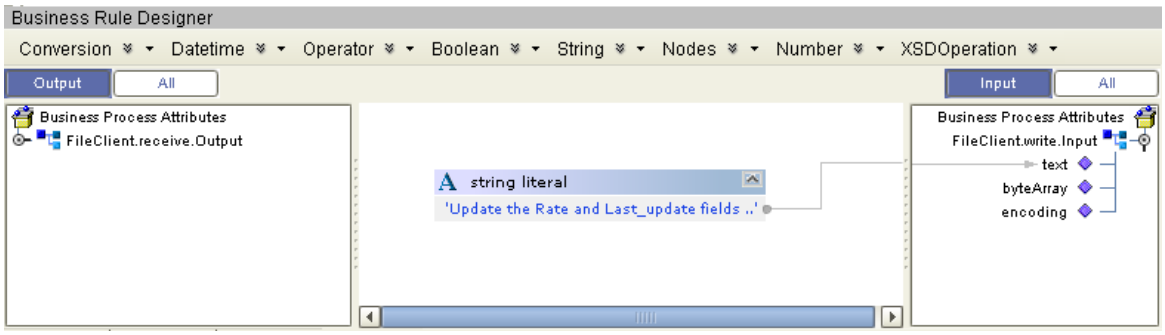


Steps required to configure the bpUpdate business process:

Configure the following business rules in the bpUpdate business process.

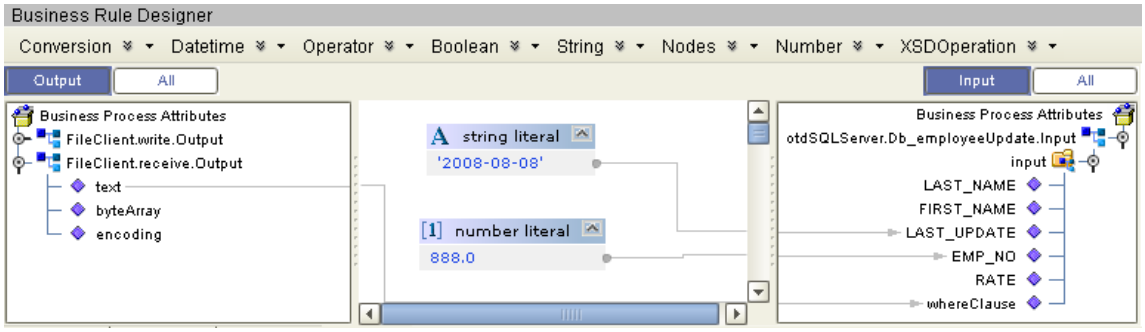
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 35.

Figure 35 bpUpdate Business Rule # 1



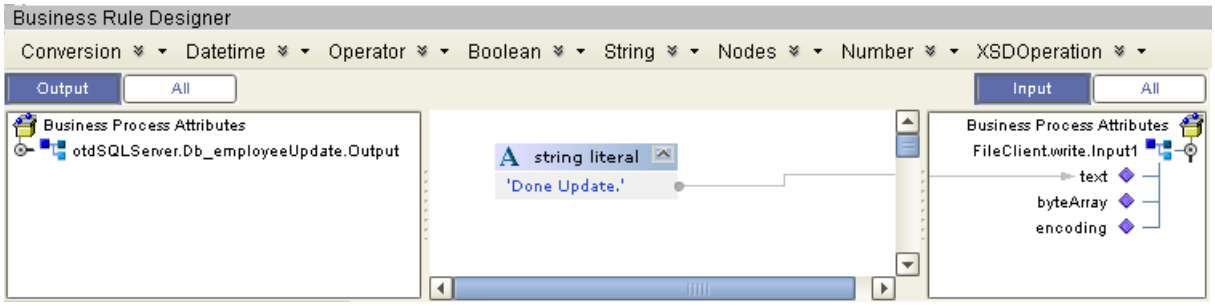
2 Configure the business rule between the **FileClient.write** Activity and **otdSQL Server.DB_EMPLOYEEUpdate** Activity as seen in Figure 36.

Figure 36 bpUpdate Business Rule # 2



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

Figure 37 bpUpdate Business Rule # 3



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

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

Configuring the bpDelete Modeling Elements

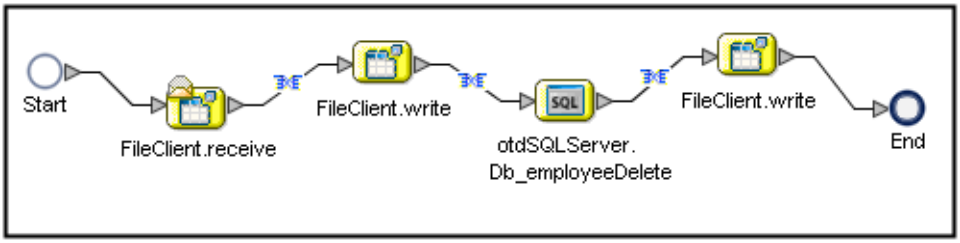
The bpDelete business process describes how to delete a record in the SQL Server database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Delete operation. See Figure 38 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.*

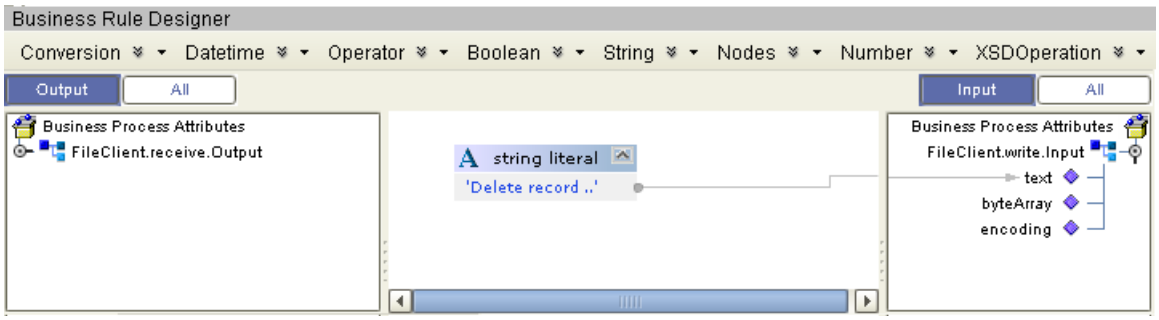
Figure 38 bpDelete Business Process



Steps required to configure the bpDelete business process:

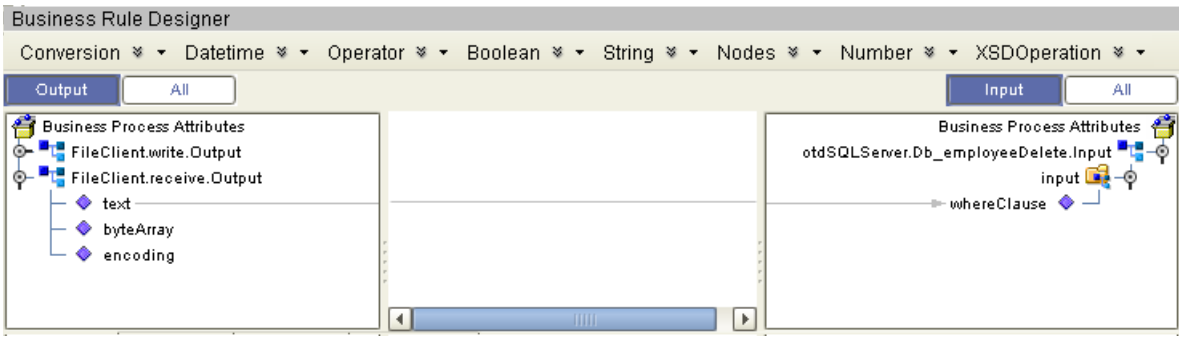
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 39.

Figure 39 bpDelete Business Rule # 1



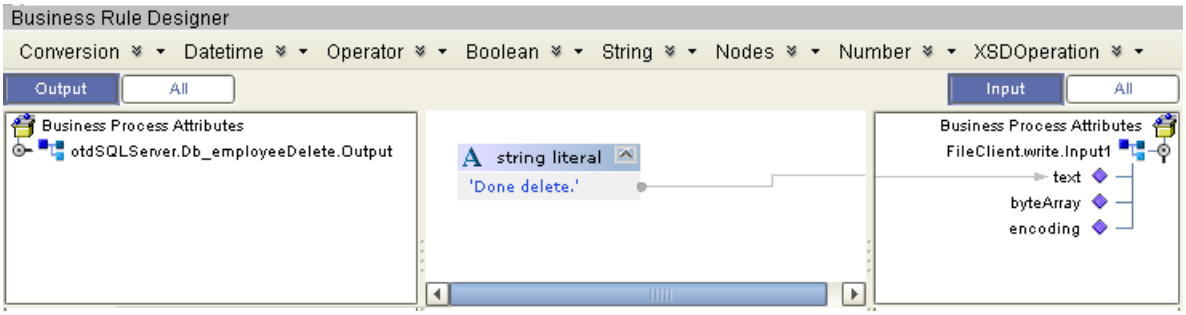
2 Configure the business rule between the **FileClient.write** Activity and **otdSQL Server.DB_EMPLOYEEDelete** Activity as seen in Figure 40.

Figure 40 bpDelete Business Rule # 2



3 Configure the business rule between the **otdSQL Server.DB_EMPLOYEEDelete** Activity and the **FileClient.write** Activity as seen in Figure 41.

Figure 41 bpDelete Business Rule # 3



Configuring the bpTableSelect Modeling Elements

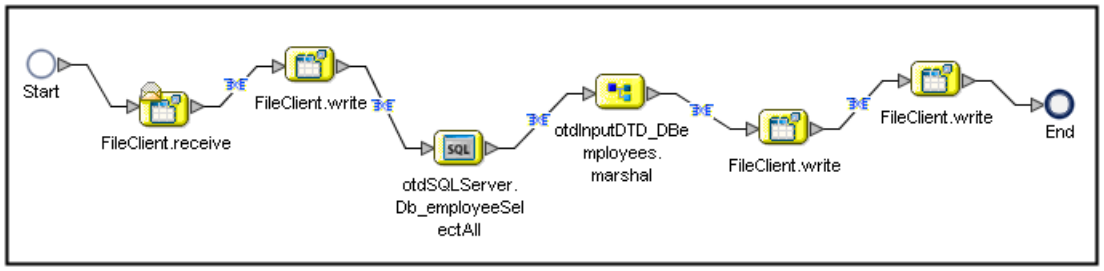
The bpTableSelect business process is describes how to select all records the SQL Server database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 42 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.*

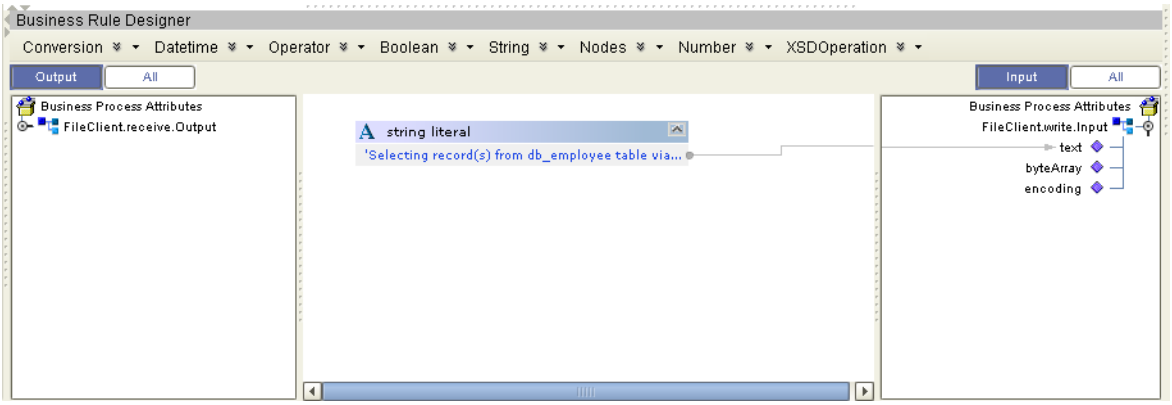
Figure 42 bpTableSelect Business Process



Steps required to configure the bpTableSelect business process:

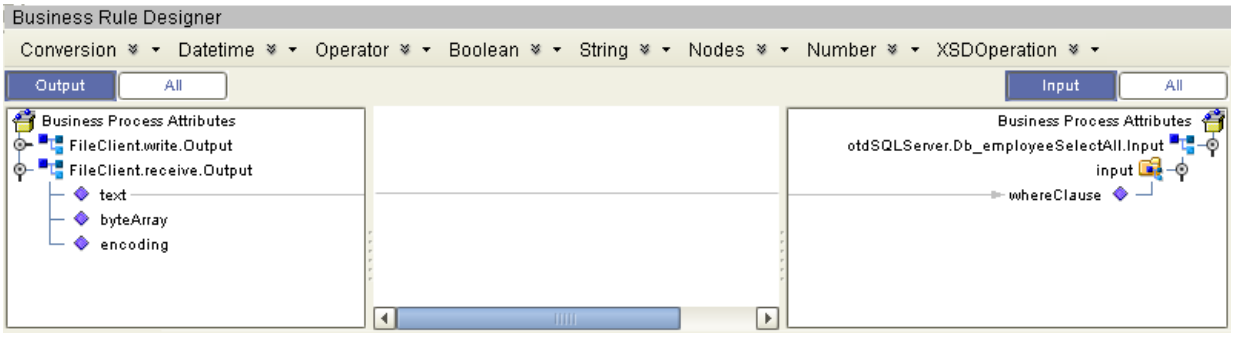
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 43.

Figure 43 bpTableSelect Business Rule # 1



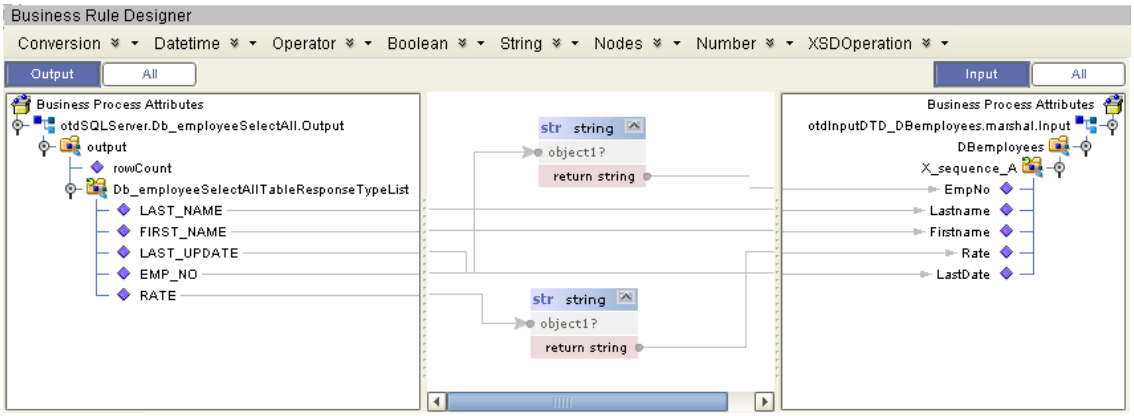
2 Configure the business rule between the **FileClient.write** Activity and **otdSQL Server.DB_EMPLOYEESelectAll** Activity as seen in Figure 44.

Figure 44 bpTableSelect Business Rule # 2



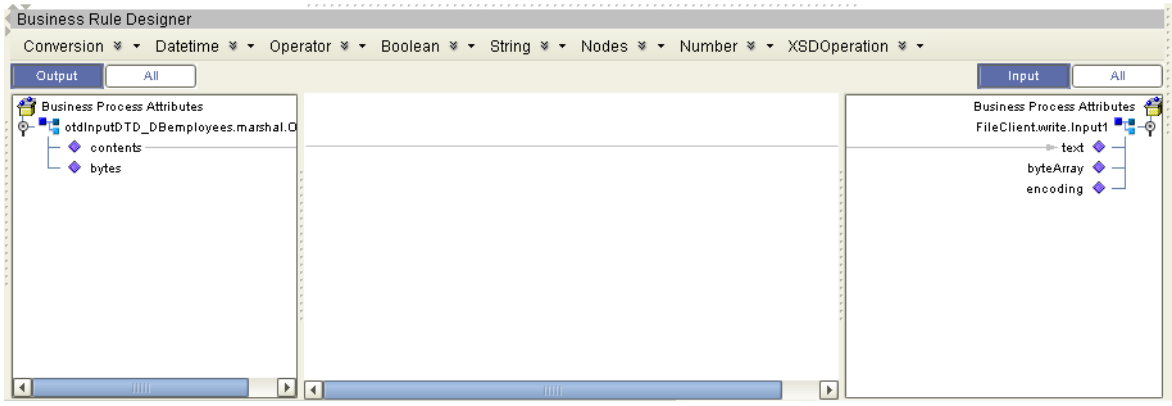
3 Configure the business rule between the **otdSQL Server.DB_EMPLOYEESelectAll** Activity and the **otdInputDTD_DBemployees.marshal** Activity as seen in Figure 45.

Figure 45 bpSelectTable Business Rule # 3



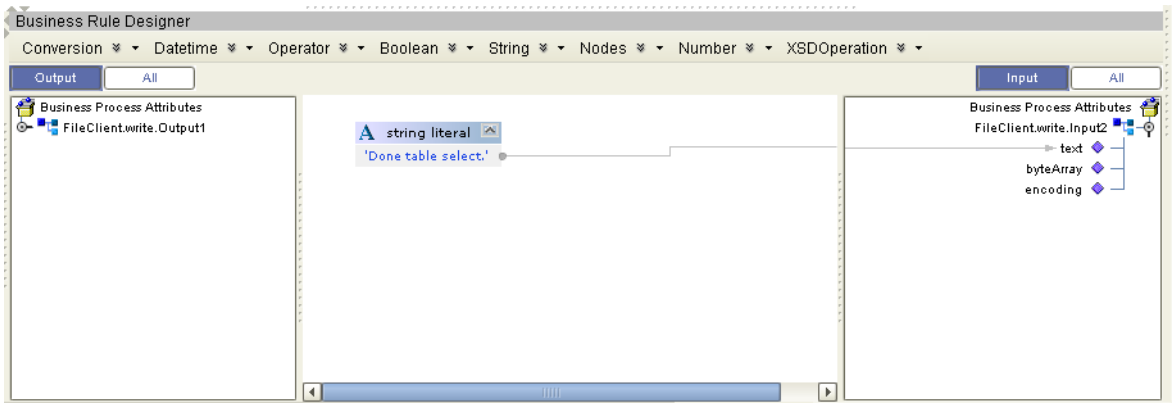
4 Configure the business rule between the **otdInputDTD_DBemployees.marshal** Activity and the **FileClient.write** Activity as seen in Figure 46.

Figure 46 bpTableSelect Business Rule # 4



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

Figure 47 bpTableSelect Business Rule # 5



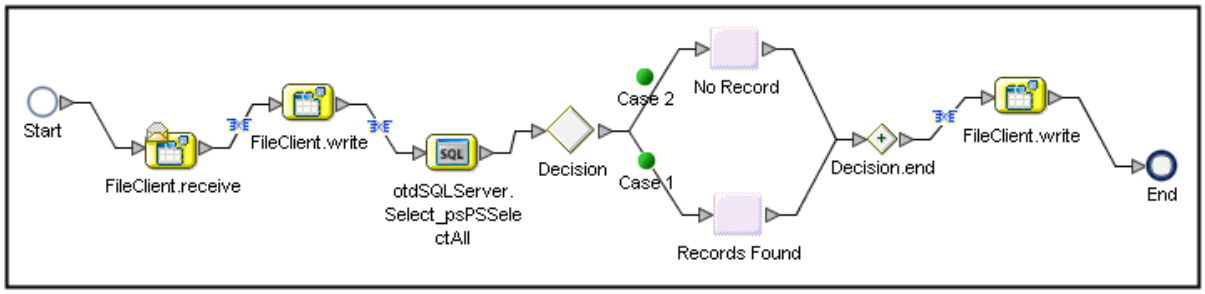
Configuring the bpPsSelect Modeling Elements

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

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 48 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.*

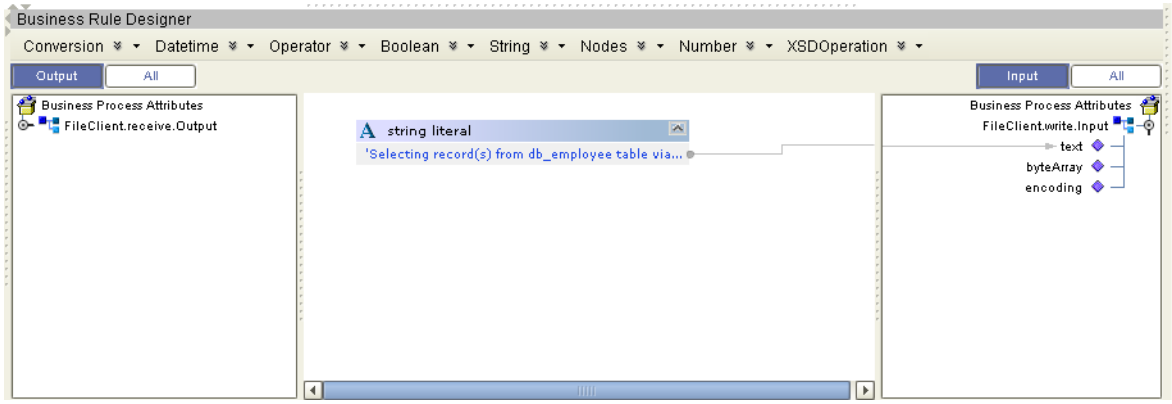
Figure 48 bpPsSelect Business Process



Steps required to configure the bpPsSelect business process:

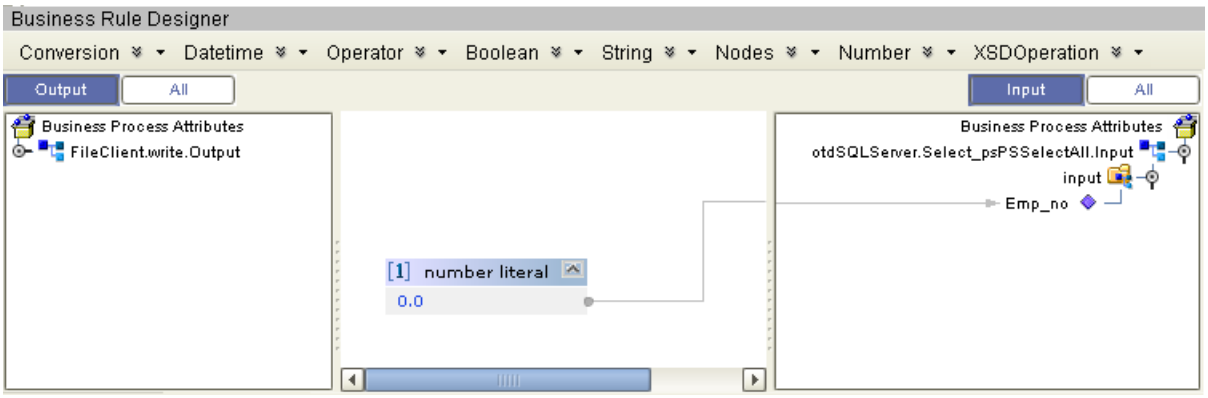
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 43.

Figure 49 bpSelectTable Business Rule # 1



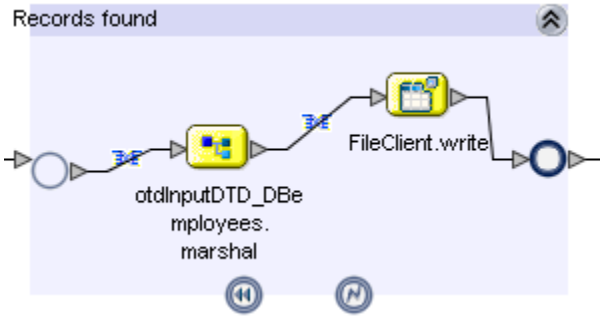
2 Configure the business rule between **FileClient.write** and **otdSQL Server.Select_psPSSelectAll** as seen in Figure 50.

Figure 50 bpSelectTable Business Rule # 2



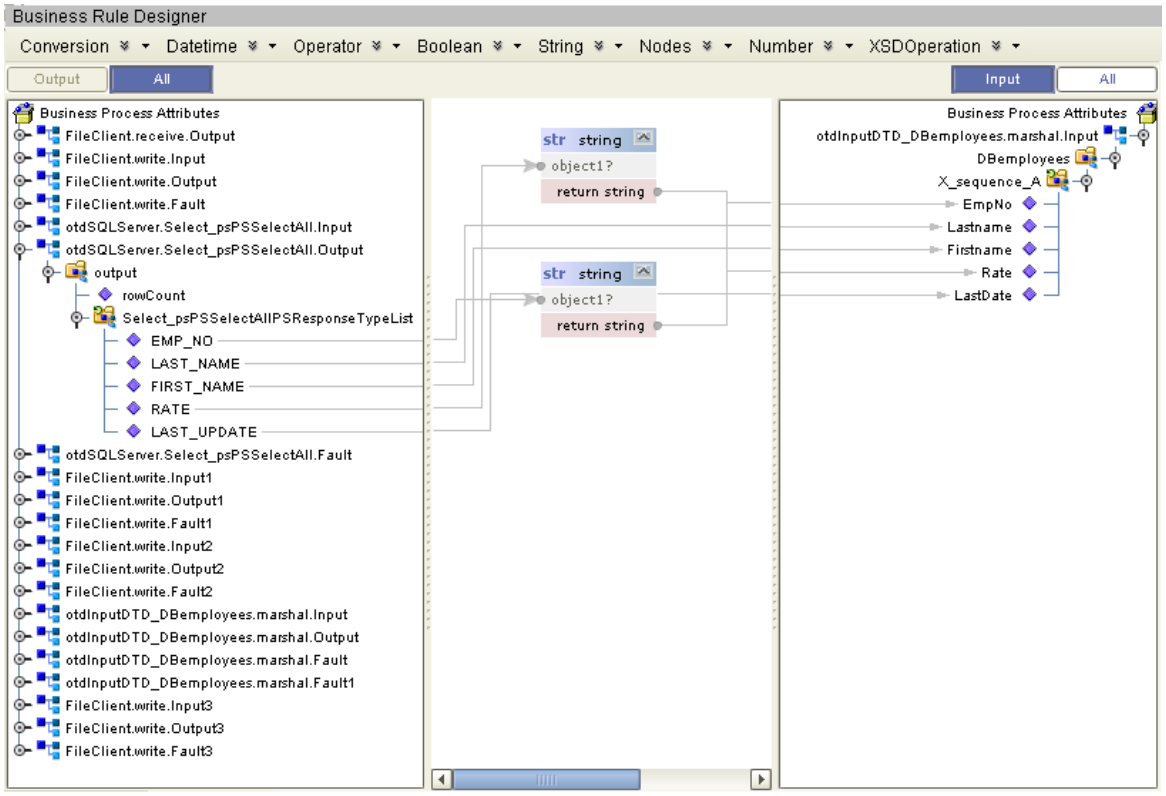
3 Configure **Case 1** of the Decision branching activity. This requires adding business rules between the **otdInputDTD_DBemployees.marshal** and the **FileClient.write** activities within the Scope element.

Figure 51 Activities within Case 1 Scope



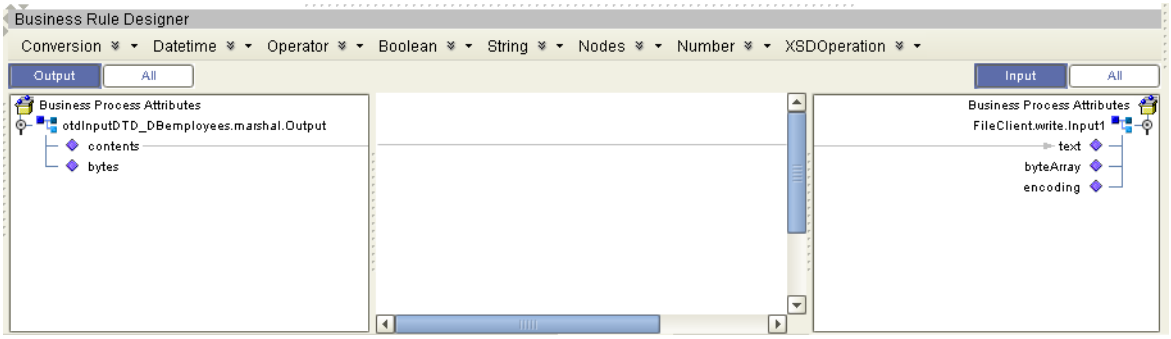
4 Configure the business rule between the start of the Scope element in **Case 1** and the **otdInputDTD_DBemployees.marshal** activity, as seen in Figure 52.

Figure 52 Case 1 Scope Business Rule # 1



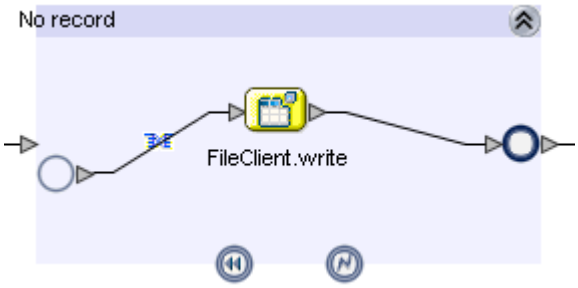
5 Configure the business rule between **otdInputDTD_DBemployees.marshal** and **FileCleint.write** in the Scope element, as seen in Figure 53.

Figure 53 Case 1 Scope Business Rule # 2



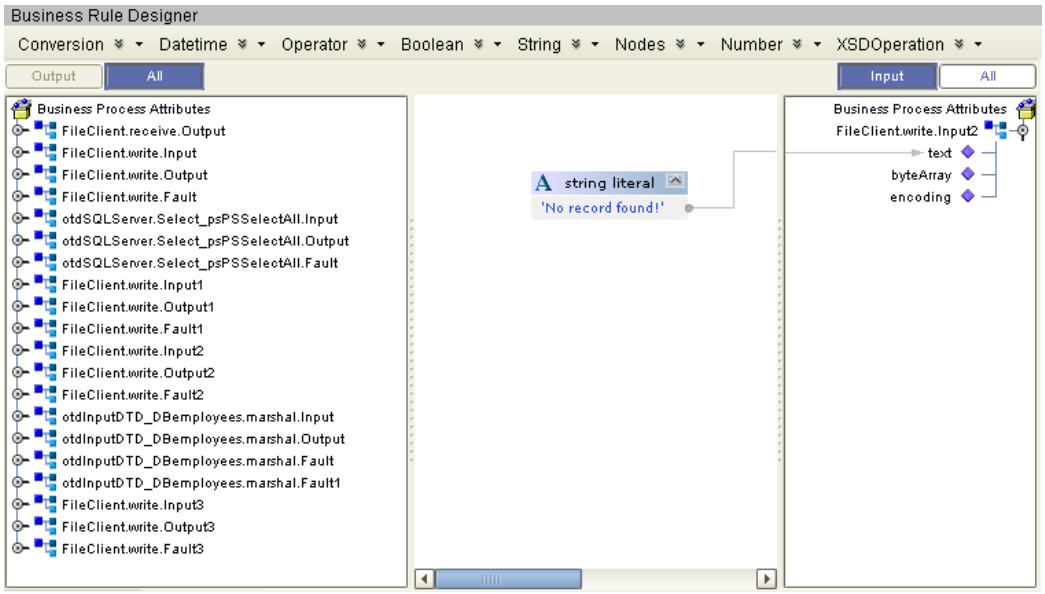
6 Configure Case 2 of the Decision branching activity. This requires adding business rules between the **otdInputDTD_DBemployees.marshal** and the **FileClient.write** activities within the Scope element.

Figure 54 Activities within Case 2 Scope



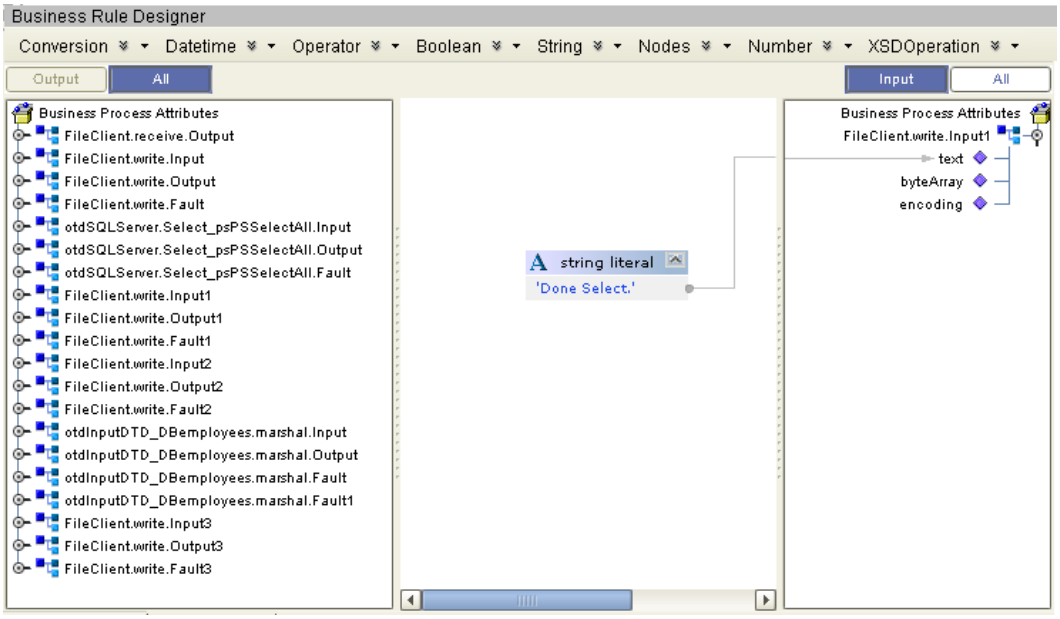
7 Configure the business rule between the start of the Scope element in **Case 2** and the **FileClient.Write** activity, as seen in Figure 55.

Figure 55 Case 2 Scope Business Rule # 1



8 Configure the business rule between the **Decision.end** Element and the **FileClient.write** Activity, as seen in Figure 56.

Figure 56 bpSelectTable Business Rule # 3



6.5.4 Creating the Connectivity Map

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

1. From the Project Explorer tree, right-click the new **prjSQL Server_BPEL** Project and select **New > Connectivity Map** from the shortcut menu.
2. The New Connectivity Map appears, and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled **CMap1**.

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

   - cmDelete
   - cmInsert
   - cmPsSelect
   - cmTableSelect
   - cmUpdate

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

Populating the Connectivity Map

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

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

- File External Application (2)
- SQL Server External Application
- Business Process

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

To Select a SQL Server 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, SQL Server 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
   - SQL Server1 to eaSQL ServerOUT

To Select a SQL Server Business Process

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

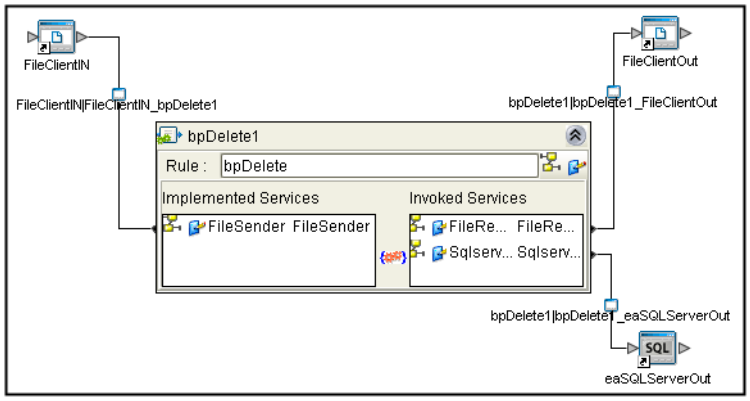
Binding the eWay Components

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

Steps required to bind eWay components together:

1. Open one of the Connectivity Maps and double-click a Business Process, for example the **bpDelete** Business Process in the **cmDelete** Connectivity Map. The **bpDelete** Binding dialog box appears.
2. From the **bpDelete** Binding dialog box, map **FileSender** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **bpDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **bpDelete**.
3. From the **bpDelete** Binding dialog box, map **SQL Server_otdSQL Server** (under Invoked Services) to the **eaSQL ServerOUT** External Application.
4. From the **bpDelete** Binding dialog box, map **FileReceiver** to the **FileClientOUT** External Application, as seen in Figure 57.

Figure 57 Connectivity Map - Associating (Binding) the Project's Components



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

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

6.5.5 Creating an Environment

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

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

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

3 Rename the new Environment to **envSQL ServerProj**.

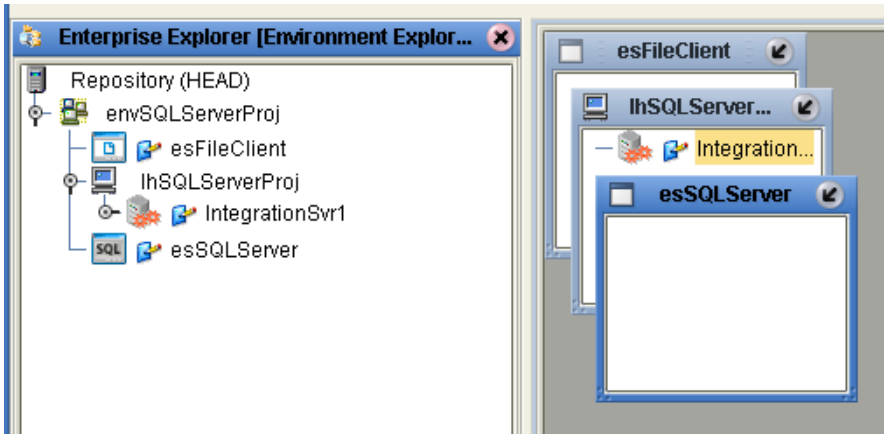
4 Right-click **envSQL ServerProj** and select **New SQL Server External System**. Name the External System **esSQL Server**. Click **OK**. **esSQL Server** is added to the Environment Editor.

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

6 Right-click **envSQL ServerProj** 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 SeeBeyond Java Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see Figure 58).

Figure 58 Environment Editor - envSQL ServerProj



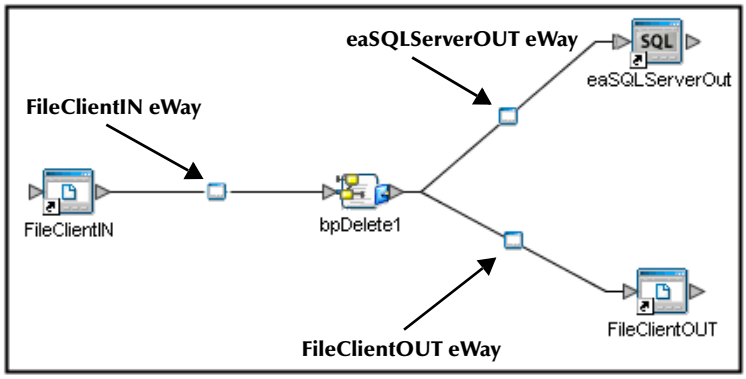
8 Save your current changes to the Repository.

6.5.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the The **prjSQL Server_BPEL** sample Project use three eWays that are represented as a nodes between the External Applications and the Business Process, as seen in Figure 59.

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

Figure 59 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 13. Click **OK** to close the Properties Editor.

Table 13 FileClientIN eWay Property Settings

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

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

Table 14 FileClientOUT eWay Property Settings

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

Steps to Configure the Environment Explorer Properties

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

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

Table 15 SQL Server eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound SQL Server 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 SQL Server eWay Environment configuration.

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

Table 16 File eWay Environment Properties

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

6.5.7 Configuring the Integration Server

You must set your 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 **SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.

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

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

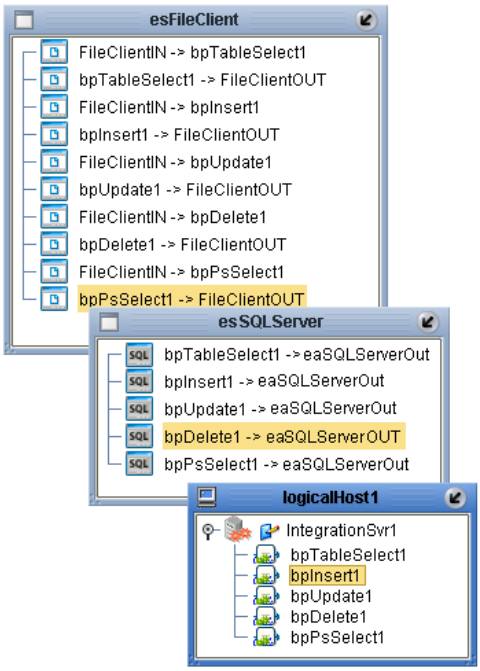
For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

6.5.8 Creating the Deployment Profile

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

1. From the Enterprise Explorer's Project Explorer, right-click the **prjSQL Server_BPEL** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpSQL Server_BPEL**). Select **envSQL ServerProj** 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 60.

Figure 60 Deployment Profile



6.5.9 Creating and Starting the Domain

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

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

Steps required to create and start the domain:

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

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

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

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

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

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

6.5.10 Building and Deploying the Project

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

Build the Project

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

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

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

Deploy the Project

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

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

Deploying a Project to an HP NonStop Server

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

6.5.11 Running the Sample Project

Perform the following steps to run your deployed sample Project:

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

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

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

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

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

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

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

2 Verify the output data by viewing the sample output files. See **About the SQL Server eWay Sample Projects** on page 60 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.

6.6 Creating the prjSQLServer_JCD Sample Project

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

Steps required to create the sample project include:

- **Creating a Project** on page 88
- **Creating the OTDs** on page 89
- **Creating a Connectivity Map** on page 90
- **Creating the Collaboration Definitions (Java)** on page 91
- **Binding the eWay Components** on page 100
- **Creating an Environment** on page 101
- **Configuring the eWays** on page 102
- **Creating and Starting the Domain** on page 106
- **Building and Deploying the Project** on page 107
- **Running the Sample** on page 107

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

6.6.2 Creating the OTDs

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

- SQL Server Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

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

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

- Host name:
- Port ID:
- Database:
- 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 SQL Server database. The **Add Tables** window appears.

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

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

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

- Prepared Statement Name: Select_ps

- SQL Statement:

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

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

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

Steps required to create inbound and outbound DTD OTDs include:

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

   The New Object Type Definition Wizard window appears.
2. Select **DTD** from the list of OTD Wizards and click **Next**.
3. Browse to and then select a DTD file. For this example, select one of the following DTD files from the sample Project, and then click **Next**.
   - otdInputDTD.dtd
   - otdOutputDTD.dtd
4. The file you select appears in the Select Document Elements window. Click **Next**.
5. Click **Finish** to complete the DTD based OTD. Repeat this process again to create the second DTD file.

6.6.3 Creating a Connectivity Map

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

Steps required to create a new Connectivity Map:

1. From the Project Explorer tree, right-click the new **prjSQLServer_JCD** Project and select **New > Connectivity Map** from the shortcut menu.
2. The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree labeled **CMap1**.

   Create four additional Connectivity Maps—**CMap2**, **CMap3**. **CMap4**, and **CMap5**— and rename them as follows:
   - cmDelete
   - cmInsert
   - cmPsSelect
   - cmTableSelect
   - cmUpdate

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

Populating the Connectivity Map

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

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

- File External Application (2)
- SQL Server External Application
- Service

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

Steps required to select a SQL Server 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, **SQL Server** 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
   - SQL Server1 to eaSQLServerOUT
4. Rename each Connectivity Map Service to match the intended operation, as for example:
   - jcdDelete
   - jcdInsert
   - jcdPsSelect
   - jcdTableSelect
   - jcdUpdate

6.6.4 Creating the Collaboration Definitions (Java)

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

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

- jcdDelete
- jcdInsert
- jcdPsSelect
- jcdTableSelect
- jcdUpdate

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 **prjSQLServer_JCD** > **otdALL** > **otdSQLServer**. The **otdSQLServer** OTD is added to the Selected OTDs field.
5. Click the **Up One Level** button to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.
6. Click **Finish**. The Collaboration Editor with the new **jcdDelete** Collaboration appears in the right pane of the Enterprise Designer.

jcdInsert Collaboration

Steps required to create the jcdInsert Collaboration:

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

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

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

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

jcdPsSelect Collaboration

Steps required to create the jcdPsSelect Collaboration:

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

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

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

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

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

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

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

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

jcdTableSelect Collaboration

Steps required to create the jcdTableSelect Collaboration:

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

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

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

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

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

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

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

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

jcdUpdate Collaboration

Steps required to create the jcdUpdate Collaboration:

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

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

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

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

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

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

6.6.5 Create the Collaboration Business Rules

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

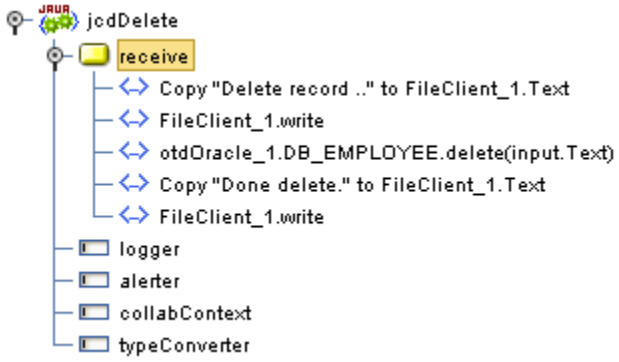
Creating the jcdDelete Business Rules

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

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

The **jcdDelete** Collaboration contains the Business Rules displayed in Figure 61.

Figure 61 jcdDelete Business Rules



Creating the jcdInsert Business Rules

The **jcdInsert** Collaboration implements the Input Web Service Operation to read the **TriggerInsert.in**. file. It then unmarshals data from the input data into the **otdInputDTD_DBEmployees** OTD, calls the **otdSQLServer** 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 62.

Figure 62 jcdInsert Business Rules

Sample code from the jcdInsert Includes:

```
package prjSQLServer_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,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1,
dtd.otdInputDTD_1206505729.DB_Employee otdInputDTD_DB_Employee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

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

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

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

        otdInputDTD_DB_Employee_1.unmarshalFromString(
input.getText() );

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

        otdSQLServer_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdSQLServer_1.getDb_employee().setEMP_NO(
typeConverter.stringToShort(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdSQLServer_1.getDb_employee().setLAST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdSQLServer_1.getDb_employee().setFIRST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdSQLServer_1.getDb_employee().setRATE( new
java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1
).getRate() ) );
            otdSQLServer_1.getDb_employee().setLAST_UPDATE(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdSQLServer_1.getDb_employee().insertRow();

\\ Writes a message to confirm an inserted records.

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

Creating the jcdPsSelect Business Rules

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

The **jcdPsSelect** Collaboration contains the Business Rules displayed in Figure 63.

Figure 63 jcdPsSelect

Sample code from the jcdPsSelect Includes:

```
package prjSQLServer_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,
otdSQLServer.OtdSQLServerOTD otdSQLServer_1,
dtd.otdOutputDTD1325973702.DB_Employee otdOutputDTD_DB_Employee_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

\\ Writes out a message stating records are being selected

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

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

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

        FileClient_1.write();
        otdSQLServer_1.getSelect_ps().setEMPNO(
typeConverter.stringToShort( "0", "#", false, 0 ) );
        otdSQLServer_1.getSelect_ps().executeQuery();
        if (otdSQLServer_1.getSelect_ps().resultsAvailable()) {
            while
(otdSQLServer_1.getSelect_ps().get$Select_psResults().next()) {
                otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.intToString(
otdSQLServer_1.getSelect_ps().get$Select_psResults().getEMP_NO(),
"#", false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastname(
otdSQLServer_1.getSelect_ps().get$Select_psResults().getLAST_NAME()
);
                otdOutputDTD_DB_Employee_1.setFirstname(
otdSQLServer_1.getSelect_ps().get$Select_psResults().getFIRST_NAME()
);
                otdOutputDTD_DB_Employee_1.setRate(
typeConverter.doubleToString(
otdSQLServer_1.getSelect_ps().get$Select_psResults().getRATE(),
"#.000000;-#.000000", false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdSQLServer_1.getSelect_ps().get$Select_psResults().getLAST_UPDATE()
, "yyyy-MM-dd hh:mm:ss", false, "" ) );
                FileClient_1.setText(
otdOutputDTD_DB_Employee_1.marshalToString() );
                FileClient_1.write();
            }
        }

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

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

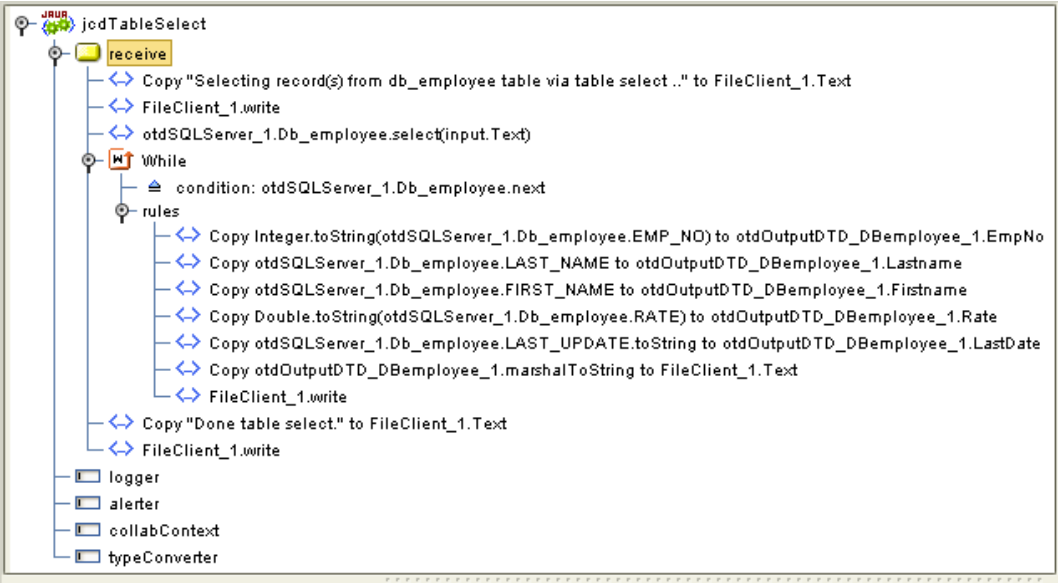
Creating the jcdTableSelect Business Rules

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

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

The **jcdTableSelect** Collaboration contains the Business Rules displayed in Figure 64.

Figure 64 jcdTableSelect



Sample code from the jcdTableSelect Includes:

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

public void receive( com.stc.connector.appconn.file.FileTextMessage
input, dtd.otdOutputDTD1325973702.DB_Employee
otdOutputDTD_DB_Employee_1, otdSQLServer.OtdSQLServerOTD
otdSQLServer_1, com.stc.connector.appconn.file.FileApplication
FileClient_1 )
        throws Throwable
    {

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

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

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

        otdSQLServer_1.getDb_employee().select( input.getText() );
        while (otdSQLServer_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.shortToString(
otdSQLServer_1.getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdSQLServer_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdSQLServer_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate(
otdSQLServer_1.getDb_employee().getRATE().toString() );
```

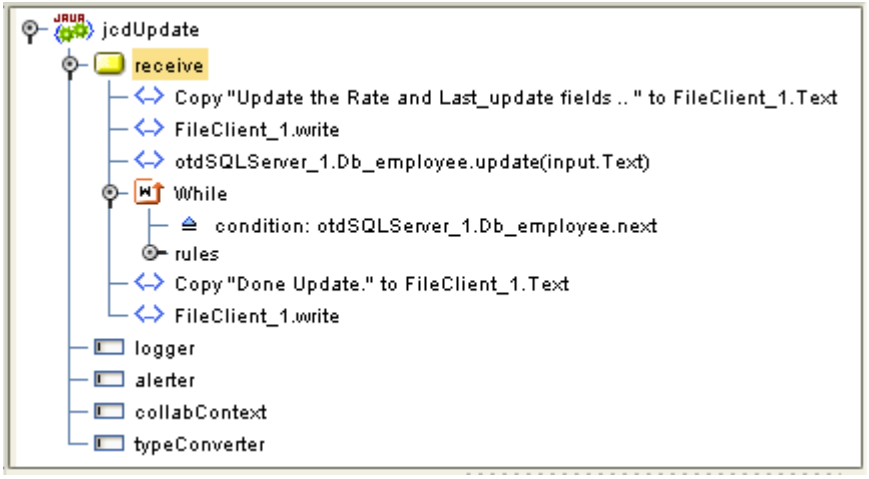
```
            otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdSQLServer_1.getDb_employee().getLAST_UPDATE(), "yyyy-MM-dd
hh:mm:ss", false, "" ) );

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

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

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

        FileClient_1.setText( "Table Select Done." );
        FileClient_1.write();
    }
}
Creating the jcdUpdate Business Rules
```

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

Note: *The where clause in the business rule reads the trigger value as a 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 65.

Figure 65 jcdUpdate



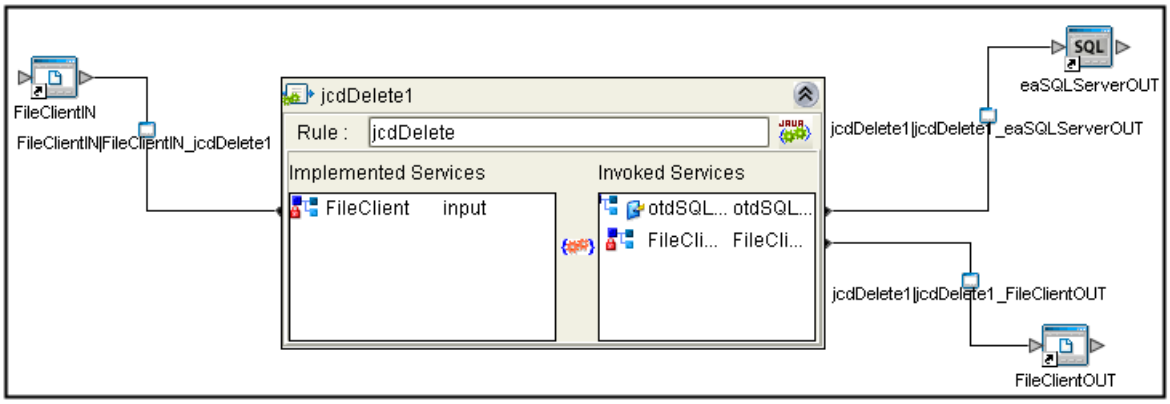
6.6.6 Binding the eWay Components

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 **bpDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **bpDelete**.
5. From the **bpDelete** Binding dialog box, map **otdSQLServer_1** (under Invoked Services) to the **eaSQLServerOUT** External Application.
6. From the **bpDelete** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 57.

Figure 66 Connectivity Map - Associating (Binding) the Project's Components



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

6.6.7 Creating an Environment

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

Steps required to create an Environment:

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

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

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

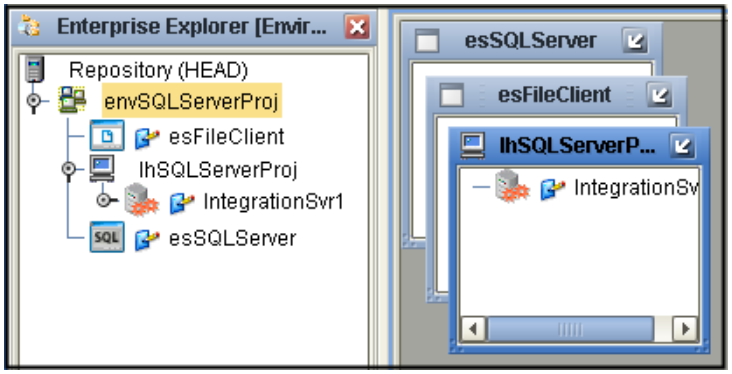
4 Right-click **envSQLServerProj** and select **New SQLServer External System**. Name the External System **esSQLServer.** Click **OK**. **esSQLServer** is added to the Environment Editor.

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

6 Right-click **envSQLServerProj** 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 Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see Figure 58).

Figure 67 Environment Editor - envSQLServerProj



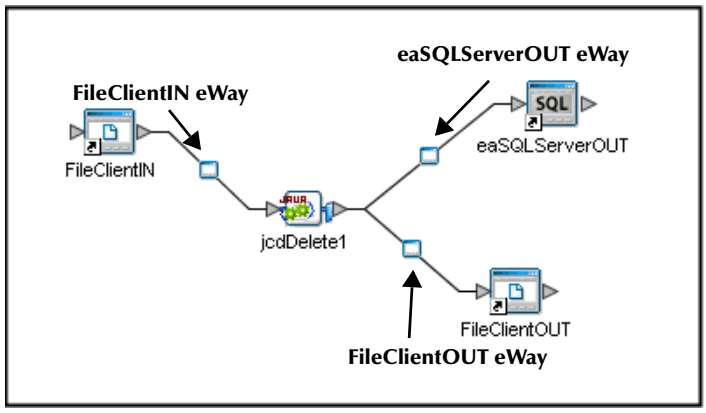
8 Save your current changes to the Repository.

6.6.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjSQLServer_BPEL** sample Project uses three eWays that are represented as nodes between the External Applications and the Business Process, as seen in Figure 59.

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

Figure 68 eWays in the cmDelete Connectivity Map



Configuring the eWay Properties

Steps required to configure the eWay properties:

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

Table 17 FileClientIN eWay Property Settings

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

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

Table 18 FileClientOUT eWay Property Settings

Connectivity Map	Property Name	Required Value
cmDelete	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.in
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 File External System (**esSQLServer** in this sample), and select **Properties**. The Properties Editor opens to the SQL Server eWay Environment configuration.
2. Modify the SQL Server eWay Environment configuration properties for your system, as seen in Table 19, and click **OK**.

Table 19 SQL Server eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound SQL Server eWay > JDBC Connector settings	ServerName	Enter the 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 SQL Server eWay Environment configuration.
4. Modify the File eWay Environment configuration properties for your system, as seen in Table 20, and click **OK**.

Table 20 File eWay Environment Properties

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

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

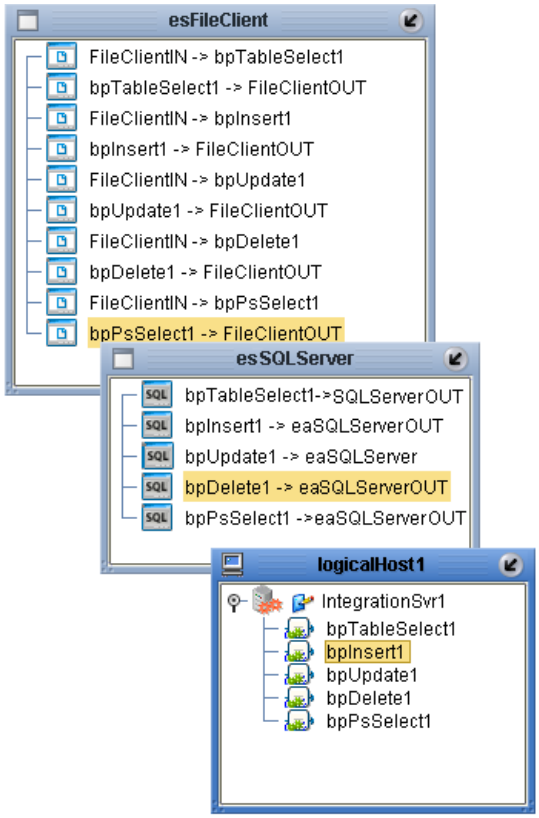
6.6.9 Creating the Deployment Profile

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

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer's Project Explorer, right-click the **prjSQLServer_BPEL** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpSQLServer_BPEL**). Select **envSQLServerProj** 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 60.

Figure 69 Deployment Profile



6.6.10 Creating and Starting the Domain

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

Steps required to create and start the domain:

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

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

6.6.11 Building and Deploying the Project

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

Build the Project

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

Deploy the Project

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

Deploying a Project to an HP NonStop Server

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

6.6.12 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
   - TriggerUpdate.in

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

Verify the output data by viewing the sample output files. See **About the SQL Server eWay Sample Projects** on page 60 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

Common DataType Conversions

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

What's in this Appendix:

- **Common DataType Conversions** on page 109

A.1 Common DataType Conversions

Table 1 lists common datatype conversions used for the SQL Server.

Table 1 The SQL Server eWay Datatype Conversions

SQL Server Data Type	OTD/Java Data Type	JCD Class Browser Fields	Sample Data
BigInt	Long	Long: **java.lang.Long.parseLong(String)**	123
Int	Int	Integer: **java.lang.Integer.parseInt(String)**	123
tinyInt	Byte	Byte: **java.lang.Byte.parseByte(String)**	123
SmallInt	Short	Short: **java.lang.Short.parseShort(String)**	123
Number	BigDecimal	BigDecimal: **java.math.BigDecimal(String)**	145.78
Decimal	BigDecimal	BigDecimal: **java.math.BigDecimal(String)**	145.78
Bit	Boolean	Boolean: **java.lang.Boolean.getBoolean(String)**	true or false
Real	Float	Float: **java.lang.Foat.parseFloat(String)**	3468.494

SQL Server Data Type	OTD/Java Data Type	JCD Class Browser Fields	Sample Data
Float	Double	Double: **java.lang.Double.parseDouble(String)**	3468.494
Money	BigDecimal	Call a NewConstructor BigDecimal: **java.math.BigDecimal(String)**	2456.95
Smallmoney	BigDecimal	Call a NewConstructor BigDecimal: **java.math.BigDecimal(String)**	2456.95
Smalldatetime	TimeStamp	TimeStamp: **java.sql.TimeStamp.valueOf(String)**	2003-09-28 11:35:00
Timestamp	Binary	N/A (Used by the Database Internally)	N/A
DateTime	TimeStamp	Date: **java.sql.TimeStamp.valueOf(String)**	2003-09-28 11:35:42
Varchar	String	Direct Assign	Any Characters
Char	String	Direct Assign	Any Characters
Text	String	Direct Assign	Any Characters
Binary(1)	Byte[]	String: java.lang.String.getBytes()	0 or 1

Index

A

Activity Input and Output **49**
Add Prepared Statements **43**
Automap **86**, **105**

B

binding
  dialog box **82**, **101**
BPEL operations **49**

C

Collaboration
  editor **91**
Configuring
  Environment Explorer properties **20**
Configuring DataDirect Drivers **13**
Connect to Database **36**
Connection Retry Settings **29**
Connectivity Map
  Inbound properties **24**
  Outbound properties **23**
conventions, text **9**

D

Database Operations
  BPEL **49**
  JCD **51**
database OTD wizard
  review selections **46**
  steps to create **35**
DataDirect 3.4 Drivers for XA Mode **13**
Delete Operation **54**
Deployment Profile
  Automap **86**, **105**

E

Editing Existing OTDs **47**
Environment Configuration Properties **25**
Environment Explorer properties **18**
eWays
  creating **19**
  Plug-Ins **15**
  properties **18**
Executing Stored Procedures **55**
External Application **18**

I

Insert Operation **52**
Installation
  eWay Product Files **11**
  Plug-Ins **15**
  Samples and Javadocs **13**
  Stored Procedures for JTA **14**
installation **11**
Installing
  Repository on UNIX **11**
  sample Projects and Javadocs **13**

J

Javadocs, installing **13**
JCD operations **51**
JDBC Connector Settings
  with XA support **30**

O

Object Type Definition (OTD)
  Editing an Existing OTD **47**
Operations
  BPEL **49**
  Delete **54**
  Insert **52**
  JCD **51**
  Query (Select) **51**
  Update **53**
OTD Editing **47**
OTD Wizard **34**
Outbound Connectivity Map Properties **23**

P

Plug-Ins
  Installing **15**
prepared statement
  batch operations **59**
  executing **58**
Project
  importing **63**
properties **18**
  ConnectionRetryInterval **30**, **33**
  Connectivity Map properties

modifying **19**
DriverProperties **28**, **31**
Environment properties
modifying **20**
MaxPoolSize **29**, **32**
MinPoolSize **29**, **32**
modifying **22**
Connectivity Map properties **19**
Properties Editor **22**

Q

Query (Select) Operation **51**

R

ResultSet
collaboration usability for a stored procedure **57**
ResultSet methods
available **56**
enableResultSetandUpdateCounts **56**
enableResultSetOnly **56**
enableUpdateCountsOnly **56**
getUpdateCount **56**
next **56**
resultsAvailable **56**

S

sample projects, installing **13**
screenshots **10**
Select Database Objects **36**
Select Procedures **40**
Select Table/Views **37**
Select Wizard Type **35**
SQL **51**
SQL operations, table **51**
SQL Server eWay Database Wizard **34**
SQL Server External Application **18**
Stored Procedure
manipulating ResultSet and update count **56**
Stored Procedures **55**
Executing **55**
Stored Procedures for JTA **14**
Sybase eWay Project
running sample projects **107**

T

Table
SQL operations **51**
text conventions **9**

U

update count **56**
Update Operation **53**