SUN SEEBEYOND

eWAY™ ADAPTER FOR SYBASE USER'S GUIDE

Release 5.1.2

Part Number: 819-7382-10

Version 20061006152439

Contents

Chapter 1

Introducing the Sybase eWay 7

About Sybase 7

About the Sybase eWay 7

What's New in This Release 8

What's In this Document 9

Scope 9

Intended Audience 9

Text Conventions 9

Related Documents 10

Sun Microsystems, Inc. Web Site 10

Documentation Feedback 10

Chapter 2

Installing the Sybase eWay 11

Installing the Sybase eWay 11

Installing the Sybase eWay on a Java CAPS system 11

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

After Installation 12

Extracting the Sample Projects and Javadocs 13

ICAN 5.0 Project Migration Procedures 13

Installing eWay Enterprise Manager plug-ins 15

Viewing Alert Codes 15

Chapter 3

Setting Properties of the Sybase eWay 18

Creating and Configuring a Sybase eWay 18

Configuring the eWay Connectivity Map Properties 19

Configuring the eWay Environment Properties 22

eWay Connectivity Map Properties 23
Configuring the Outbound eWay Properties 23
Configuring the Outbound XA eWay Properties 24
Configuring the Outbound Sybase non-Transactional eWay Properties 24
eWay External Properties 25
Inbound Sybase eWay 25
Outbound Sybase eWay 26
JDBC Connector Settings 26
Connection Retry Settings 28
Outbound XA Sybase eWay 29
JDBC Connector Settings 29
Connection Retry Settings 31
Outbound Sybase non-Transactional eWay 31
JDBC Connector Settings 31
Connection Retry Settings 33

Chapter 4
Using the Sybase eWay Database Wizard 34
About the Database OTD Wizard 34
Steps to Create a New Sybase OTD 34
Select Wizard Type 35
Connect to Database 35
Select Database Objects 36
Select Tables/Views/Aliases 37
Select Procedures 40
Add Prepared Statement 44
Specify the OTD Name 46
Review Selections 47
Steps to Edit an Existing Sybase OTD 48

Chapter 5
Using Sybase Operations 49
Sybase eWay Database Operations (BPEL) 49
Activity Input and Output 49
Sybase eWay Database Operations (JCD) 51
The Table 51
The Query (Select) Operation 52
The Insert Operation 53
The Update Operation 54
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 59
Batch Operations 59

Chapter 6

Implementing the Sybase eWay Sample Projects 60

About the Sybase eWay Sample Projects 60
Operations Used in the Sybase Sample Projects 61
Assigning Operations in JCD 62
Assigning Operations in BPEL 62
About the eInsight Engine and eGate Components 62

Steps Required to Run the Sample Projects 63

Running the SQL Script 63

Importing a Sample Project 64

Building and Deploying the prjSybase_BPEL Sample Project 64
Creating a Project 64
Creating the OTDs 65
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 74
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 82
Configuring the eWay Properties 83
Configuring the Environment Explorer Properties 84
Creating the Deployment Profile 85
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 prjSybase_JCD Sample Project 88
Creating a Project 88
Creating the OTDs 88
Creating a Connectivity Map 90
Populating the Connectivity Map 90
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 96
Creating the jcdTableSelect Business Rules 98

Creating the jcdUpdate Business Rules 100
Binding the eWay Components 101
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

Index 111

Chapter 1

Introducing the Sybase eWay

Welcome to the *Sun SeeBeyond eWay™ Adapter for Sybase*. This document includes information about installing, configuring, and using the Sun SeeBeyond eWay™ Adapter for Sybase eWay, referred to as the Sybase eWay throughout this guide.

What's in This Chapter

- **About Sybase** on page 7
- **About the Sybase eWay** on page 7
- **What's New in This Release** on page 8
- **What's In this Document** on page 9
- **Sun Microsystems, Inc. Web Site** on page 10
- **Documentation Feedback** on page 10

1.1 About Sybase

Sybase Adaptive Server Enterprise (ASE) is a powerful data management platform for high performance business applications.

Sybase ASE's highly reliable data management technology provides a powerful data management platform that supports the demanding needs of mission critical enterprises, accelerating application development, securing critical company and customer data, and easing data administration tasks.

1.2 About the Sybase eWay

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

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

- Query or Update 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.

Though SQL coding is not required, the Sybase eWay also supports a full set of SQL functions for advanced users.

The Sybase 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 Sybase eWay includes the following new features:

What's New in 5.1.2

There are no new features in this release.

What's New in 5.1.1

- **Sybase ASE 15:** Support for Adaptive Server Enterprise 15.

What's New in 5.1

- **Version control:** An enhanced version control system allows you to effectively manage changes to the eWay components.
- **Multiple Drag-and-Drop Component Mapping from the Deployment Editor:** The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- **Support for Runtime LDAP Configuration:** Configuration properties now support LDAP key values.
- **Connection Retry Support:** Allows you to specify the number of attempts to reconnect, and the interval between retry attempts, in the event of a connection failure.
- **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.
- **Relaunchable OTD Support:** An OTD can be rebuilt and saved (under the same name) then relaunched back to the same Java Collaboration or BPEL. This allows you to change the metadata in an OTD without having to completely recreate the business logic from scratch.
- **Connectivity Map Generator:** Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.
- **DataDirect JDBC 3.5 Driver:** The Sybase eWay is now packaged with the DataDirect JDBC 3.5 driver.

1.4 What's In this Document

This document includes the following chapters:

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

1.4.1 Scope

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

This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the Sybase eWay Javadoc files from the Java Composite application Platform Suite Installer.

1.4.2 Intended Audience

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

1.4.3 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

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

1.4.4 Related Documents

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

- *Sun SeeBeyond eGate™ Integrator User's Guide*
- *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 Sybase eWay

This chapter describes how to install the Sybase eWay on a Java CAPS system.

What's in This Chapter

- **Installing the Sybase eWay** on page 11
- **ICAN 5.0 Project Migration Procedures** on page 13
- **Installing eWay Enterprise Manager plug-ins** on page 15

2.1 Installing the Sybase eWay

The Java Composite Application Platform Suite Installer, referred to throughout this guide as the Suite Installer, is a web-based application that is used to select and upload core products, composite applications, and add-on files (eWays) during the installation process. The following section describes how to install the components required for this eWay.

Refer to the readme for the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

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

2.1.1 Installing the Sybase eWay on a Java CAPS system

Follow the directions for installing the Sun Java Composite Application Platform Suite (Java CAPS).

After you have installed eGate™ or eInsight™, do the following:

1. From the Suite Installer, click the Administration tab, and then click the link to install additional products.
2. Select the products for your Java Composite Application Platform Suite, and include the following:

- **FileeWay** (the File eWay is used by most sample Projects)
- **SybaseeWay**

To upload the Sybase eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, select the following:

- **SybaseeWayDocs**

3 Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.

4 From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.

5 Once your product's installation is finished, continue installing the Java Composite Application Platform Suite as instructed in the *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 Java CAPS installation include:

1 Complete steps 1 through 4 above.

2 Once your product's installation is finished, open the Enterprise Designer and select **Update Center** from the Tools menu. The **Update Center Wizard** appears.

3 For Step 1 of the wizard, simply click **Next**.

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

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

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

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

After Installation

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

2.1.2 Extracting the Sample Projects and Javadocs

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

Steps to extract the Javadoc include:

1. Click the Documentation tab of the Suite Installer, then click the Add-ons tab.
2. Click the Sybase eWay Intelligent Adapter link. Documentation for the Sybase eWay appears in the right pane.
3. Click the icon next to **Javadoc** and extract the ZIP file.
4. Open the index.html file to view the Javadoc.

Steps to extract the Sample Projects include:

1. Click the Documentation tab of the Suite Installer, then click the Add-ons tab.
2. Click the Sybase eWay Intelligent Adapter link. Documentation for the Sybase eWay appears in the right pane.
3. Click the icon next to **Sample Projects** and extract the ZIP file. Note that the **Sybase_eWay_Sample.zip** file contains two additional ZIP files for each sample Project.

   Refer to **Importing a Sample Project** on page 64 for instructions on importing the sample Project into your repository via the Enterprise Designer.

2.2 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0.x Projects to the Java Composite Application Platform Suite, version 5.1.2

To migrate your ICAN 5.0.x Projects, do the following:

Export the Project

1. Before you export your Projects, save your current ICAN 5.0.x Projects to your Repository.
2. From the Project Explorer, right-click your Project and select **Export** from the shortcut menu. The Export Manager appears.
3. Select the Project that you want to export in the left pane of the Export Manager and move it to the Selected Projects field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.
4. In the same manner, select the Environment that you want to export in the left pane of the Export Manager and move it to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.

5 Browse to select a destination for your Project ZIP file and enter a name for your Project in the **ZIP file** field.

6 Click **Export** to create the Project ZIP file in the selected destination.

Install Java CAPS 5.1.2

1 Install **Java CAPS 5.1.2**, including all eWays, libraries, and other components used by your ICAN 5.0 Projects.

2 Start Java CAPS 5.1.2 Enterprise Designer.

Import the Project

1 From Java CAPS 5.1.2 Enterprise Designer's Project Explorer tree, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.

2 Browse to and select your exported Project file.

3 Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository when the Project was created, that are not installed on the Java CAPS 5.1.2 Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.

4 Close the Import Manager after the Project is successfully imported.

Deploy the Project

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

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

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

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

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

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

   To reconfigure your File eWays, do the following:

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

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

3 Deploy your Projects.

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

2.3 Installing eWay Enterprise Manager plug-ins

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

The *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 Suite Installer's **DOWNLOADS** tab. The plugin required for Sybase is listed as the **Sybase eWay Enterprise Manager Plug-in**.

The following steps are required to install eWay plug-ins into the Enterprise Manager:

1 From the **Enterprise Manager's** Explorer toolbar, click the **Configuration** icon.

2 Click the **Web Applications Manager** tab, go to the **Auto-Install from Repository** sub-tab, and connect to your Repository.

3 Select the application plug-ins you require, and click **Install**. The application plug-ins are installed and deployed.

Alternately, you can install eWay plug-ins using the following steps:

1 From the 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 the **Configuration** icon.

3 Click the **Web Applications Manager** tab and go to the **Manage Applications** sub-tab.

4 Browse for and select the WAR file for the application plug-in that you downloaded, and click **Deploy**. The plug-in is installed and deployed.

Viewing Alert Codes

You can view 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 the **Configuration** icon.
3. Click the **Web Applications Manager** tab and go to the **Manage Alert Codes** tab. Your installed eWay alert codes display under the **Results** section.

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

Table 2 Alert Codes for the Sybase eWay

Alert Code\Description	Description Details	User Actions
DBCOMMON-CONNECT-FAILED000001=Failed to connect to database {0} on host {1}. Reason: The Pooled connection could not be allocated: [{2}]	Occurs during the initial database connection establishment.	▪ Database is down; start your database. ▪ External configuration information is invalid. You may need to verify the following: ◆ Server name ◆ Database name ◆ User ◆ Password ◆ Port
DBCOMMON-CONNECT-FAILED000002=Operation failed because of a database connection error. Reason: [{0}]	Occurs while retrieving a connection from the database or the connection pool.	▪ Verify that the database has not terminated with unexpected errors.
DBCOMMON-CONNECT-FAILED000005=Connection handle not usable. Reason:[{0}]	The connection in the pool is stale and is not usable.	▪ Probably a database restart occurred causing the connection to be stale, retry the operation after the database is up.
DBCOMMON-XARESOURCE-FAILED000001=Unable to get XAResource for the database. Reason: [{0}]	Could not obtain XAResource for the connection.	▪ Check if the database supports XA and has been configured for Distributed Transaction Support.
DBCOMMON-XACONNECT-FAILED000001=Failed to connect to database {0} on host {1}. The XA connection could not be allocated: Reason [{2}]	Occurs during the initial database connection establishment.	▪ Check if the database is configured for XA and if the database is running. ▪ External configuration information is invalid. You may need to verify the following: ◆ Server name ◆ Database name ◆ User ◆ Password ◆ Port

Alert Code\Description	Description Details	User Actions
DBCOMMON-XASTART-FAILED000001=Unable to perform XAStart for the connection. Reason: [{0}]	A connection error has occurred which caused XASTART to fail.	▪ Check if the database is running, and there are no network issues.
DBCOMMON-XAEND-FAILED000001=XAEnd failed. Reason: [{0}]	Error occurred during commit on XA connection.	▪ Look for the detailed error mentioned in the alert for the appropriate action.
DBCOMMON-CANNOT-GET-ISOLATION-LEVEL=Unable to get isolationLevel for the transaction. Reason: [{0}]	Could not read transaction isolation information of the connection.	▪ Transaction isolation is one of the following constants: ♦ Connection.TRANSACTION_READ_UNCOMMITTED ♦ Connection.TRANSACTION_READ_COMMITTED ♦ Connection.TRANSACTION_REPEATABLE_READ ♦ Connection.TRANSACTION_SERIALIZABLE ♦ Connection.TRANSACTION_NONE ***Note:*** Confirm with the vendor that the getIsolation() method of the connection is implemented correctly.

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

Chapter 3

Setting Properties of the Sybase eWay

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

What's in This Chapter

- **Configuring the Outbound eWay Properties** on page 23
- **Configuring the Outbound XA eWay Properties** on page 24
- **Inbound Sybase eWay** on page 25
- **Outbound Sybase eWay** on page 26
- **Outbound XA Sybase eWay** on page 29
- **Outbound Sybase non-Transactional eWay** on page 31

3.1 Creating and Configuring a Sybase eWay

All eWays contain a unique set of default configuration parameters. After the eWays are established and a Sybase External System is created in the Project's Environment, the eWay parameters are modified for your specific system. The Sybase eWay configuration parameters are modified from two locations:

- From the **Connectivity Map**—which contains parameters specific to the Sybase external system, and may vary from other eWays (of the same type) in the Project.
- From the **Environment Explorer** tree—which contains global parameters that commonly apply to all external systems (of the same type) in the Project. Saved parameters are shared by all eWays in the Sybase External System Properties window.

Note: *You must set configuration parameters for the Sybase eWay in both locations.*

Note: *Properties set from the Collaboration will override the corresponding properties in the eWay's configuration file. Any properties that are not overridden retain their configured default settings.*

3.2 Configuring the eWay Connectivity Map Properties

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

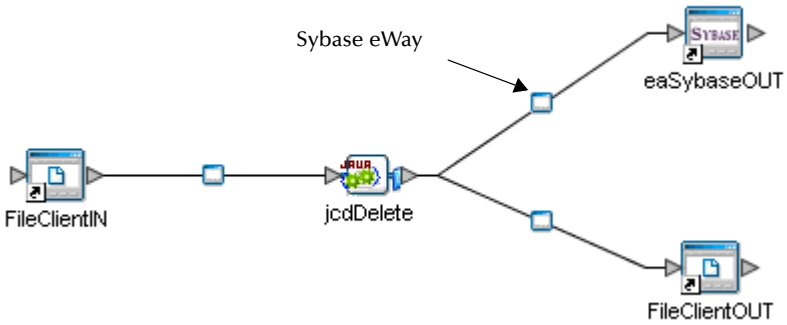
Transaction support levels supported by the Sybase eWay include:

- Outbound Sybase non-Transactional eWay
- Outbound Sybase eWay
- Outbound Sybase XA eWay

To configure the eWay properties and set the type of transaction support:

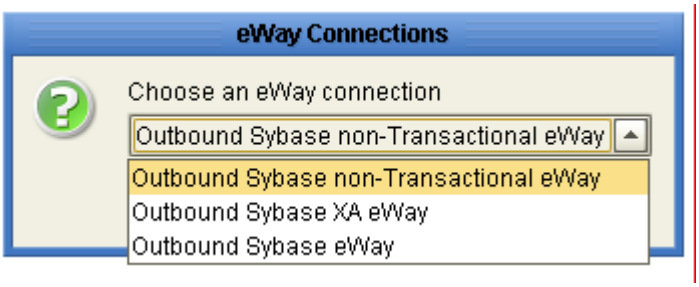
1 On the Enterprise Designer's Connectivity Map, double-click the Sybase eWay icon. The eWay Connections window appears.

Figure 1 Connectivity Map with Components



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

Figure 2 Template window



The choices to make are as follows:

- **Outbound Sybase 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 Sybase 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 Sybase 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.

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 5.1.2. 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 5.1.2, 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:

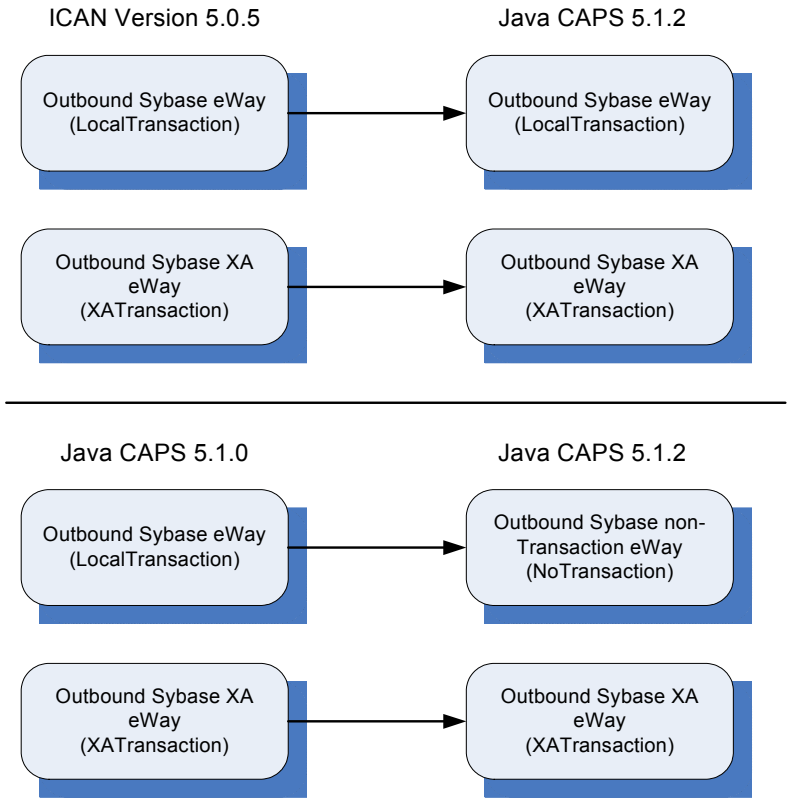
A In the Connectivity Map, delete the link to the database external application, then reconnect the link and select NoTransaction.

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 5.1.2, and 5.1.0 and 5.1.2, respectively. **Note that there are no changes when migrating from ICAN version 5.0.5 and Java CAPS 5.1.2.**

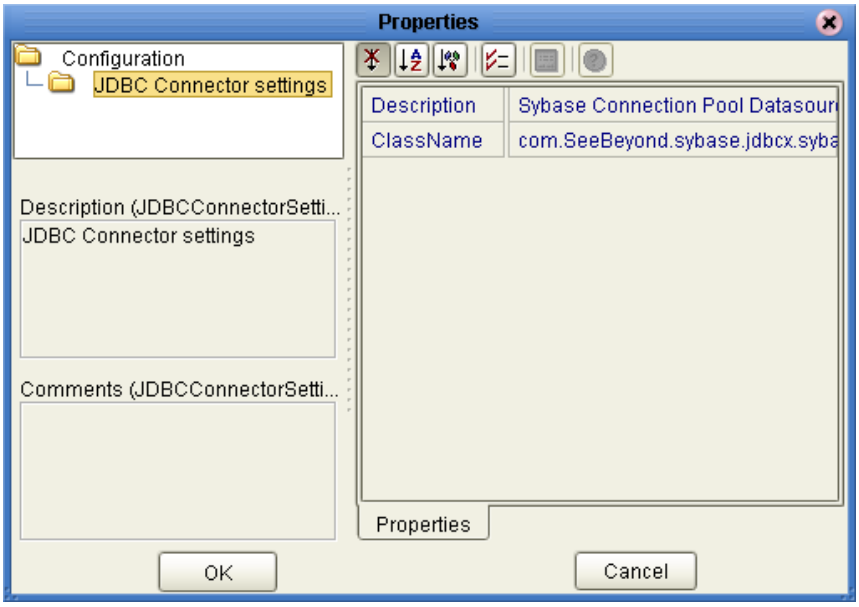
Figure 3 Transaction Support Levels



Under the scenario noted above, if you want 5.1.2 behavior for a LocalTransaction, then set your eWay connection to be Outbound Sybase non-Transactional eWay (NoTransaction).

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

Figure 4 Outbound eWay Properties



3.3 Configuring the eWay Environment Properties

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

Available External System properties include:

- Inbound Sybase eWay
- Outbound Sybase eWay
- Outbound Sybase XA eWay
- Outbound Sybase non-Transactional eWay

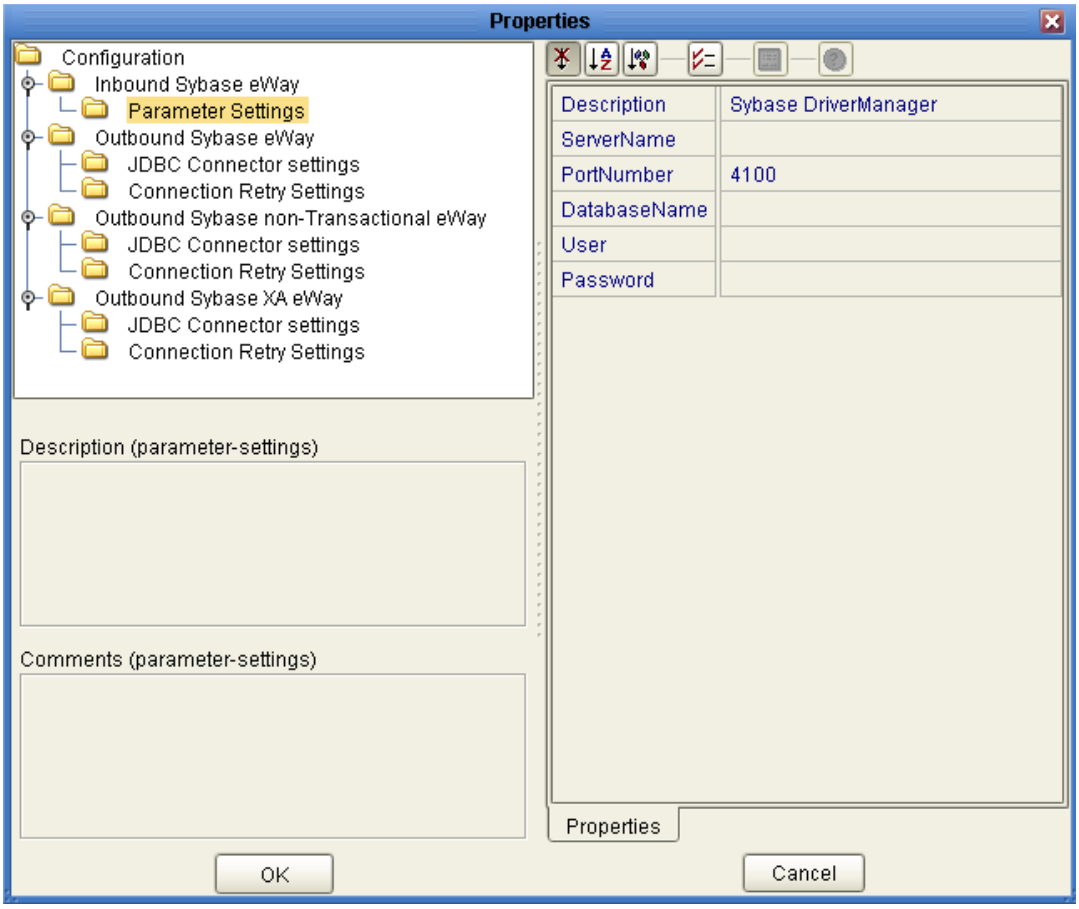
To Configure the Environment Properties:

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

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

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

Figure 5 Sybase eWay Environment Configuration



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

5 Click on any property field to make it editable.

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

3.4 eWay Connectivity Map Properties

The eWay Connectivity Map consists of the following properties categories.

- JDBC Connector Settings

3.4.1 Configuring the Outbound eWay Properties

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

Table 3 Outbound eWay—JDBC Connector Settings

Name	Description	Required Value
Description	Sybase Connection Pool Datasource.	A valid string.
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.jdbcx.sybase.sybaseDataSource.** ***Note:*** Do not change this value.

3.4.2 Configuring the Outbound XA eWay Properties

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

Table 4 Outbound eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	A valid string.
ClassName	Displays the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.	A valid class name. The default is: **com.SeeBeyond.jdbcx.sybase.sybaseDataSource**. ***Note:*** Do not change this value.

3.4.3 Configuring the Outbound Sybase non-Transactional eWay Properties

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

Table 5 Outbound eWay—JDBC Connector Settings

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

Table 5 Outbound eWay—JDBC Connector Settings

Name	Description	Required Value
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.jdbcx.sybase.sybaseDataSource**. ***Note:*** Do not change this value.

3.5 eWay External Properties

The eWay External System consists of the following properties categories.

- **Inbound Sybase eWay** on page 25
- **Outbound Sybase eWay** on page 26
- **Outbound XA Sybase eWay** on page 29
- **Outbound Sybase non-Transactional eWay** on page 31

3.5.1 Inbound Sybase eWay

The Inbound Sybase eWay includes the following configuration section:

- Parameter Settings

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

Table 6 Inbound Sybase eWay—Parameter Settings

Name	Description	Required Value
Description	The description of the database.	A valid string.
ServerName	Specifies the host name of the external database server.	Any valid string. ***Note:*** The Sybase eWay does not support using "localhost" as the server name.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **4100**.
DatabaseName	Specifies the name of the database instance.	Any valid string.

Table 6 Inbound Sybase eWay—Parameter Settings

Name	Description	Required Value
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

3.5.2 Outbound Sybase eWay

The Outbound Sybase eWay includes the following configuration sections:

- JDBC Connector Settings
- Connection Retry Settings

JDBC Connector Settings

The following Parameter Settings are used by the external database.

Table 7 Outbound Sybase eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	A valid string.
ServerName	This setting specifies the host name of the external database server.	Any valid string. ***Note:*** The Sybase eWay does not support using "localhost" as the server name.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **4100**.
DatabaseName	Specifies the name of the database instance.	Any valid string.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

Table 7 Outbound Sybase eWay—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 7 Outbound Sybase eWay—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 a longer connection time 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 you set depends on the transaction volume and response time of the application. If the pool size is too big, you may end up with too many connections with the database.	A valid numeric value. The default is **10**.
MaxIdleTime	Specifies the maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value. The default is **0**.

Connection Retry Settings

The following Parameter Settings are used by the external database.

Table 8 Outbound Sybase eWay—Connection Retry Settings

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

Table 8 Outbound Sybase eWay—Connection Retry Settings

Name	Description	Required Value
ConnectionRetryInterval	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 10 times in 5 seconds intervals when the Connection Retries is 10 and the Connection Retry Interval is 5000.	A valid numeric value. The default is **1000**.

3.5.3 Outbound XA Sybase eWay

The Outbound XA Sybase eWay includes the following configuration sections:

- JDBC Connector Settings
- Connection Retry Settings

JDBC Connector Settings

The following Parameter Settings are used by the external database.

Table 9 Outbound XA Sybase eWay—JDBC Connector Settings

Name	Description	Required Value
Description	The description of the database.	Any valid string.
ServerName	Specifies the host name of the external database server.	Any valid string. ***Note:*** The Sybase eWay does not support using "localhost" as the server name.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **4100**.
DatabaseName	Specifies the name of the database instance.	Any valid string.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.

Table 9 Outbound XA Sybase eWay—JDBC Connector Settings

Name	Description	Required Value
DriverProperties	If you choose to not to use the JDBC driver that is shipped with this eWay, you will need to add the drivers properties to the eWay. Often times the DataSource implementation will need to execute additional methods to assure a connection. 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 **#**.
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.	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.	A valid numeric value. The default is **10**.

Table 9 Outbound XA Sybase eWay—JDBC Connector Settings

Name	Description	Required Value
MaxIdleTime	Specifies the number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value. The default is **0**.

Connection Retry Settings

The following Parameter Settings are used by the external database.

Table 10 Outbound XA Sybase eWay—Connection Retry Settings

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

3.5.4 Outbound Sybase non-Transactional eWay

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

- JDBC Connector Settings
- Connection Retry Settings

JDBC Connector Settings

The following Parameter Settings are used by the external database.

Table 11 Outbound Sybase non-Transactional eWay—JDBC Connector Settings

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

Table 11 Outbound Sybase non-Transactional eWay—JDBC Connector Settings

Name	Description	Required Value
ServerName	Specifies the host name of the external database server.	Any valid string. ***Note:*** The Sybase eWay does not support using "localhost" as the server name.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1521**. ***Important:*** The default port number displayed in this field is incorrect. The value should be **4100**.
DatabaseName	Specifies the name of the database instance.	Any valid string.
User	Specifies the user name the eWay uses to connect to the database.	Any valid string.
Password	Specifies the password used to access the database.	Any valid string.
DriverProperties	If you choose to not to use the JDBC driver that is shipped with this eWay, you will need to add the drivers properties to the eWay. Often times the DataSource implementation will need to execute additional methods to assure a connection. 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 11 Outbound Sybase non-Transactional eWay—JDBC Connector Settings

Name	Description	Required Value
Delimiter	This is the delimiter character to be used in the DriverProperties prompt.	The default is **#**.
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.	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.	A valid numeric value. The default is **10**.
MaxIdleTime	Specifies the number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.	A valid numeric value. The default is **0**.

Connection Retry Settings

The following Parameter Settings are used by the external database.

Table 12 Outbound Sybase non-Transactional eWay eWay—Connection Retry Settings

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

Chapter 4

Using the Sybase eWay Database Wizard

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

What's in This Chapter

- **"About the Database OTD Wizard" on page 34**
- **"Steps to Create a New Sybase OTD" on page 34**
- **"Steps to Edit an Existing Sybase OTD" on page 48**

4.1 About the Database OTD Wizard

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

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

The Sybase eWay also supports Double-Byte Character Set (DBCS) table and column names. The DBCS is a set of characters in which each character is represented by 2 bytes. The Korean language requires double-byte character sets.

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

4.2 Steps to Create a New Sybase OTD

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

- **Select Wizard Type** on page 35
- **Connect to Database** on page 35
- **Select Database Objects** on page 36
- **Select Tables/Views/Aliases** on page 37

- **Select Procedures** on page 40
- **Add Prepared Statement** on page 44
- **Specify the OTD Name** on page 46
- **Review Selections** on page 47

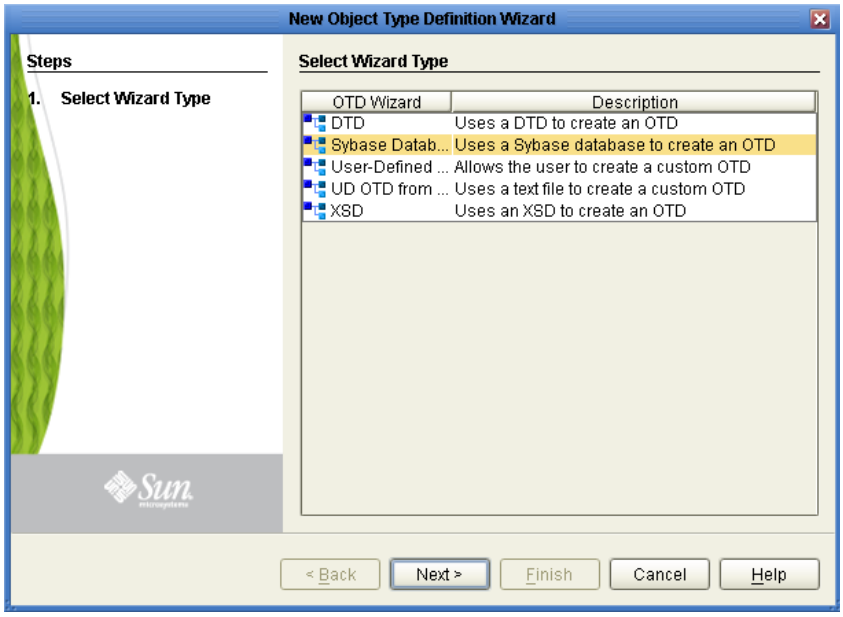
4.2.1 Select Wizard Type

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

Steps Required to Select the Sybase Database OTD Wizard Include:

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 **Sybase Database** and click **Next**. See **Figure 6**.

Figure 6 OTD Wizard Selection



4.2.2 Connect to Database

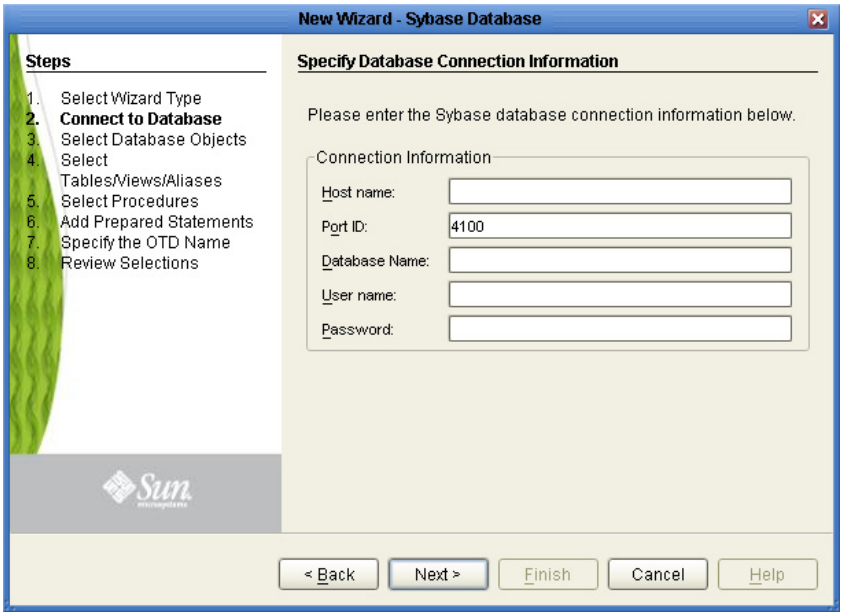
Enter the Sybase database connection information in the Connection Information frame.

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 Sybase database.

- User name – a valid Sybase database username.
- Password – a password for the user name noted above.

Figure 7 Database Connection Information



4.2.3 Select Database Objects

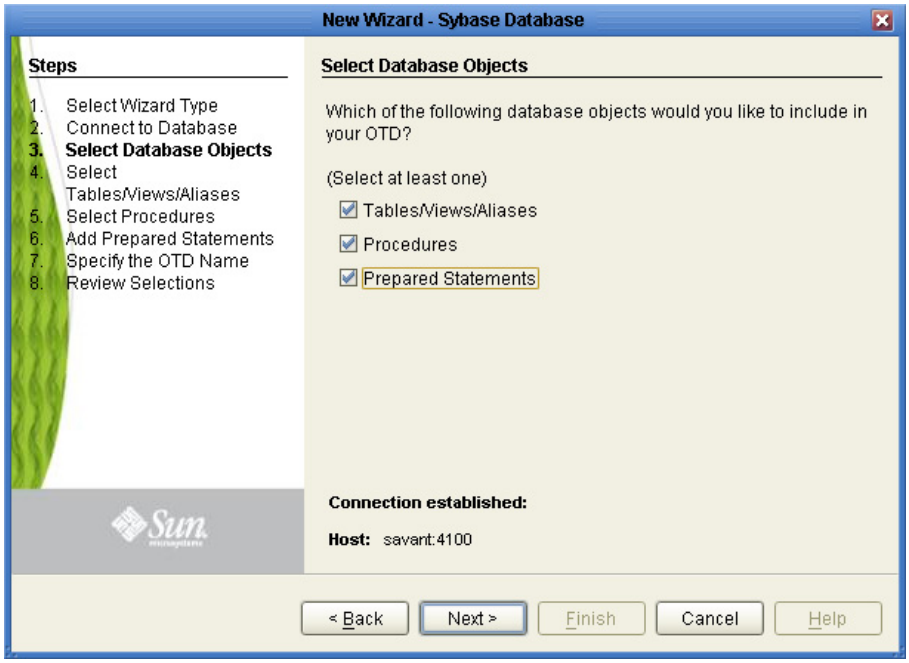
Select the type of Sybase database objects you want included in the OTD.

Steps Required to Select Database Objects Include:

1 When selecting Database Objects, you can select any combination of **Tables**, **Views**, **Procedures**, or **Prepared Statements** you would like to include in the .otd file. Click **Next** to continue. See **Figure 8**.

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

Figure 8 Select Database Objects



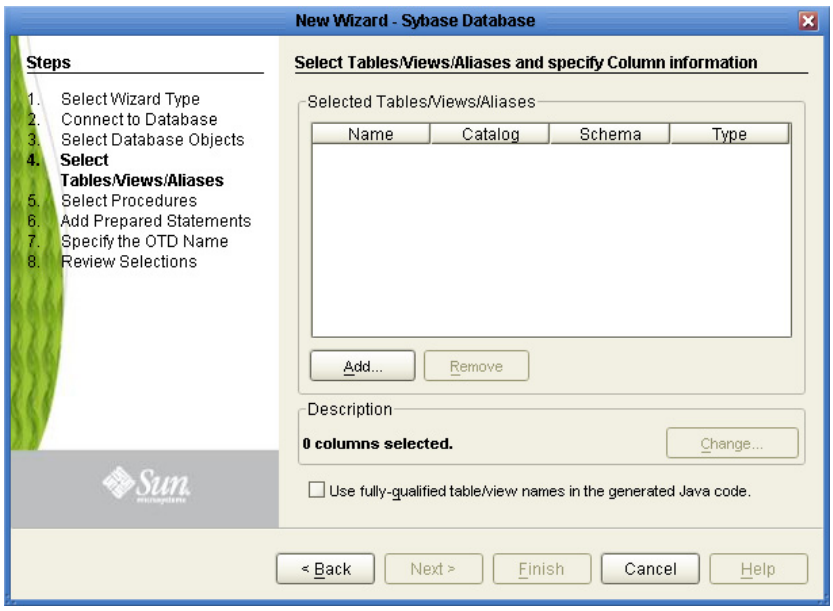
4.2.4 Select Tables/Views/Aliases

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

Steps Required to Select Table/Views/Aliases Include:

1 In the **Select Tables/Views/Aliases** window, click **Add**. See **Figure 9**.

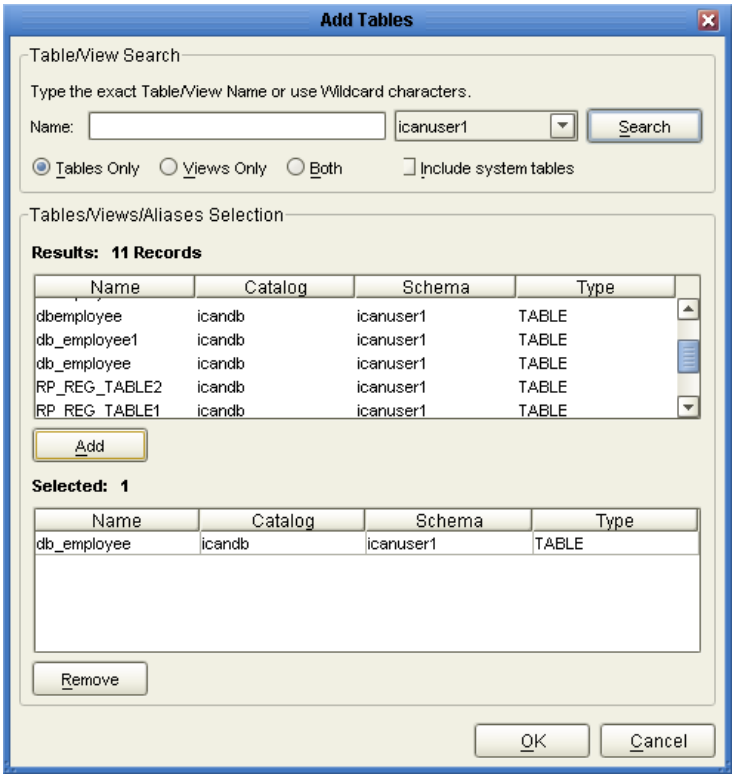
Figure 9 Select Tables/Views/Aliases



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 10**.

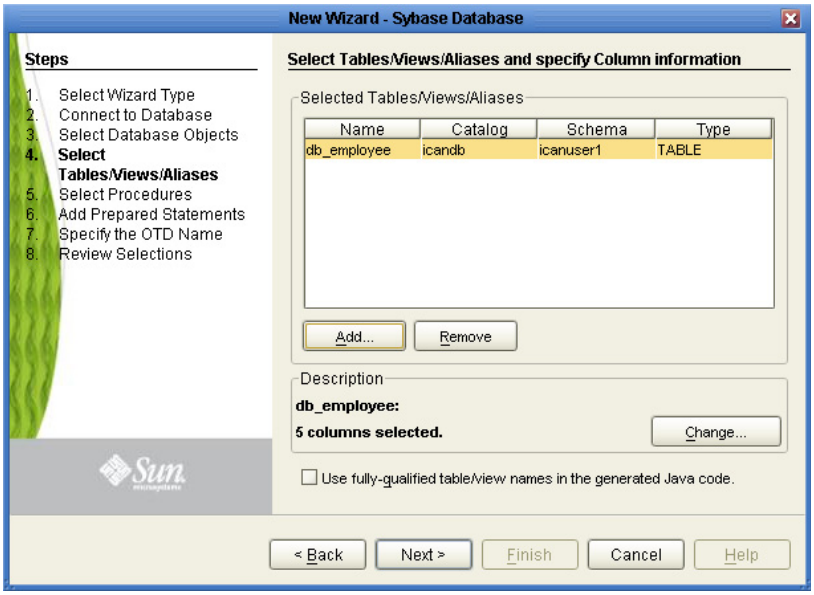
Figure 10 Database Wizard - All Schemes



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

The table selected is added to the **Selected Tables/Views** window. See **Figure 11**.

Figure 11 Selected Tables/Views/Aliases window with a table selected

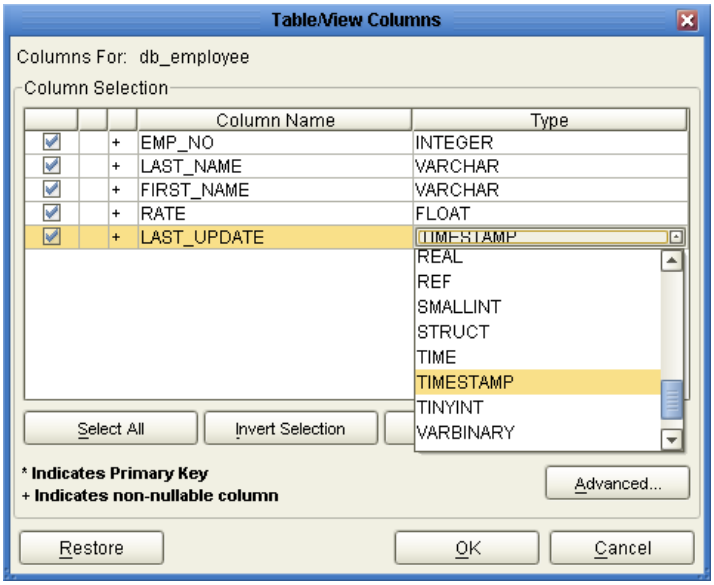


5 In the **Selected Tables/Views/Aliases** window, review the table(s) you have selected. To make changes to the selected Table or View, click Change. If you do not wish to make any additional changes, click **Next** to continue.

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 12**.

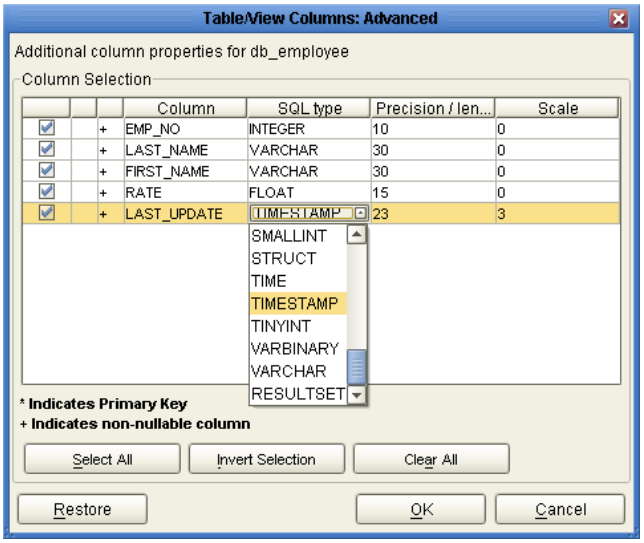
The data type is usually listed as **Other** when the driver cannot detect the data type. In these situations we recommend changing the data type to one that is more appropriate for the type of column data.

Figure 12 Table/View Columns



7 Click **Advanced** to change the data type, percision/length, or scale. Once you have finished your table choices, click **OK**. In general, you will not need to make any changes. See **Figure 13**.

Figure 13 Table/View Columns — Advanced



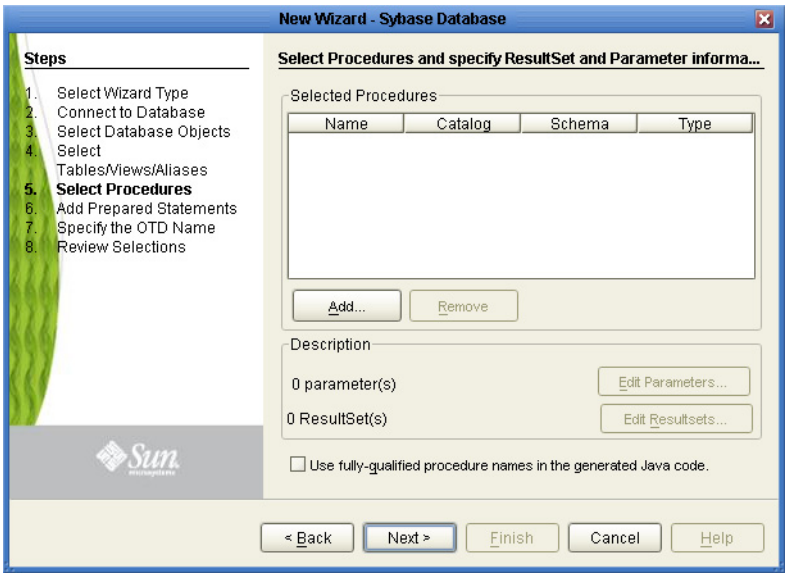
4.2.5 Select Procedures

Select the type of stored procedures required in your OTD.

Steps Required to Select Stored Procedures Include:

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

Figure 14 Select Procedures and specify Resultset and Parameter Information



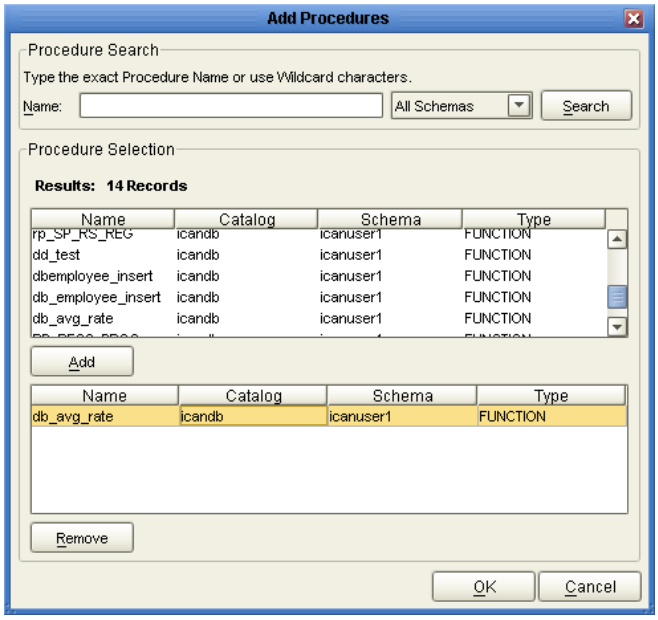
Note: *A fully qualified Procedure Name consists of two parts: Schema Name, and the Stored Procedure Table Name.*

Code that is generated in a Java Collaboration Definition appears fully qualified when the Use fully-qualified procedure names in the generated Java code checkbox is selected. When this checkbox is not selected, then only the Stored Procedure Table Name appears.

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.

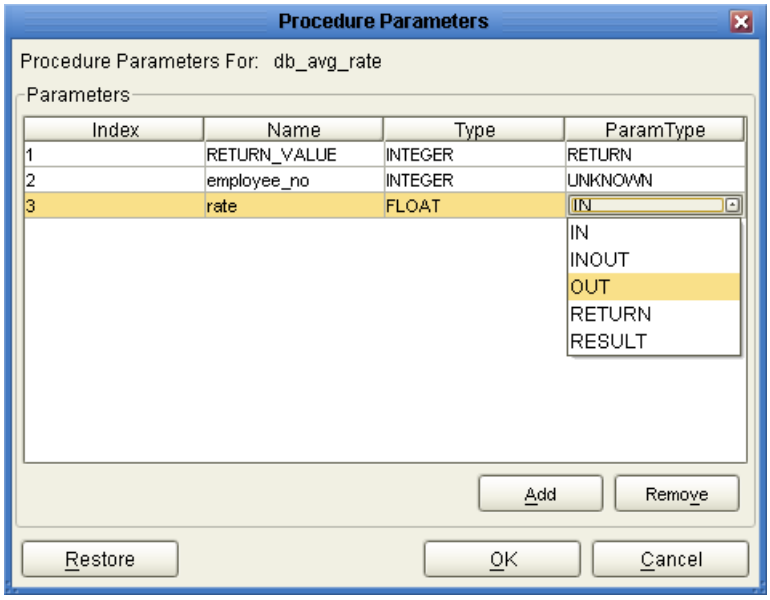
3 In the resulting **Procedure Selection** list box, select a Procedure. Click **OK**.

Figure 15 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 16**.

Figure 16 Procedure Parameters

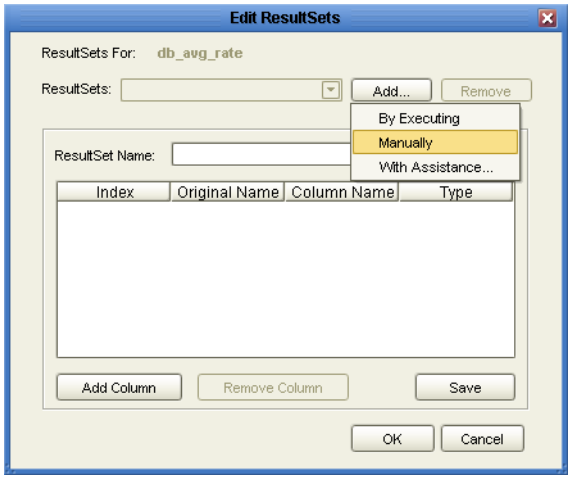


5 To restore the data type, click **Restore**. When finished, click **OK**.

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 17 Edit Resultset



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

By Executing Mode	“By Executing” mode executes the specified Stored Procedure with default values to generate the ResultSet(s). Depending on the business logic of the Stored Procedure, zero or more ResultSets can be returned from the execution. In the case that there are multiple ResultSets and "By Executing" mode does not return all ResultSets, one should use the other modes to generate the ResultSet nodes.
With Assistance Mode	"With Assistance" mode allows users to specify a query and execute it to generate the ResultSet node. To facilitate this operation, the DBWizard tries to retrieve the content of the specified Stored Procedure and display it. However, content retrieval is not supported by all types of Stored Procedures. We can roughly classify Stored Procedures into two types: SQL and external. SQL Stored Procedures are created using CREATE PROCEDURE SQL statements while external Stored Procedures are created using host languages (e.g. Java). Since external Stored Procedures do not store their execution plans in the database, content retrieval is impossible. When using "Assist" mode, highlight the execute statement up to and including the table name(s) before executing the query.
Manually Mode	"Manually" mode is the most flexible way to generate the result set nodes. It allows users to specify the node name, original column name and data type manually. One drawback of this method is that users need to know the original column names and data types. This is not always possible. For example, the column name of 3*C in this query. `SELECT A, B, 3*C FROM table T` is generated by the database. In this case, "With Assistance" mode is a better choice. If you modify the ResultSet generated by the “Execute” mode of the Database Wizard you need to make sure the indexes match the Stored Procedure. This assures your ResultSet indexes are preserved.

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

Note: *In some situations, stored procedures that uses a Select statement may not return a resultset when a conditional statement is used. In this case, no data is returned when the next() method is called, even if the resultset available() method returns “true”. This result is consistent with the type of driver we use.*

Note: *When you use Insert, Update, and Delete operations, in addition to using Select, the stored procedure will return results for each operation used. We recommend*

invoking the enableResultSetsOnly() method if you only want to return a resultset for the Select statement.

4.2.6 Add Prepared Statement

Add a Prepared Statement object to your OTD.

Steps Required to Add Prepared Statements Include:

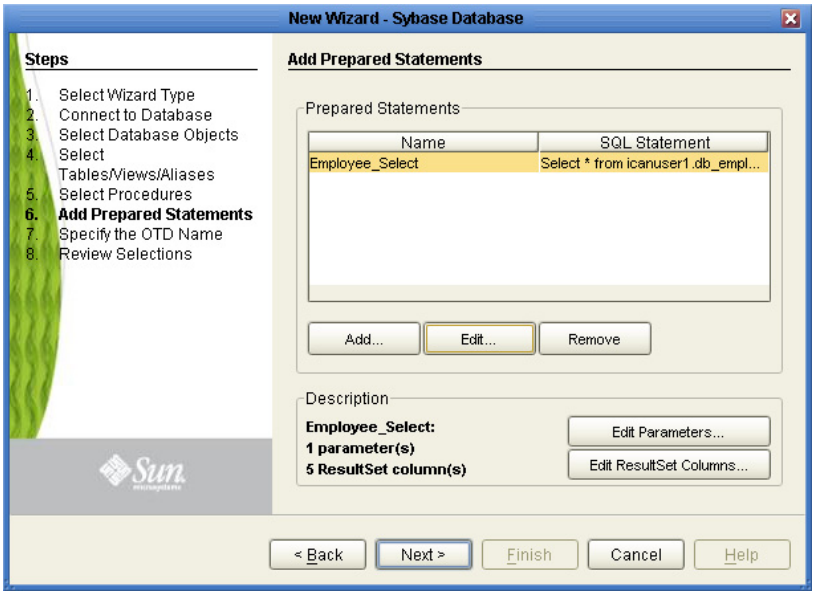
Note: *When using a Prepared Statement, the 'ResultsAvailable()' method always returns true. Although this method is available, you should not use it with a 'while' loop since it creates an infinate loop at runtime, exhausting all CPU activity.*

You can process a resultset by looping through the next() method. For more information, see **The Query (Select) Operation** *on page 52.*

Note: *If the Prepared Statement is wrong, then the ResultSet Columns count is zero.*

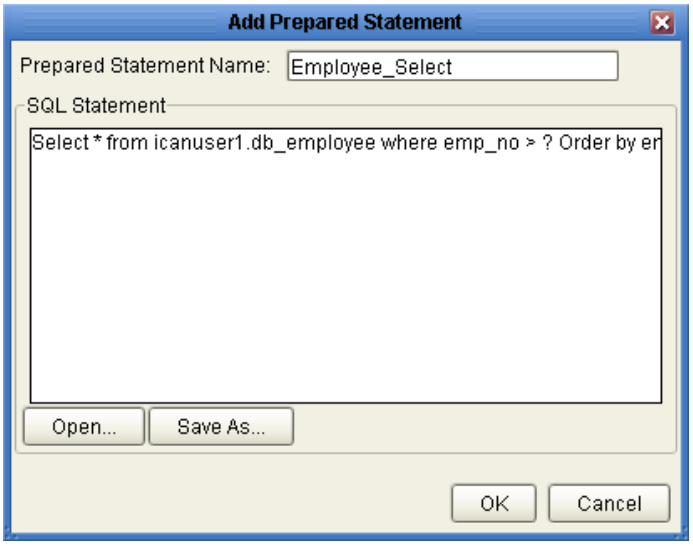
1 On the **Add Prepared Statements** window, click **Add**.

Figure 18 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 19**.

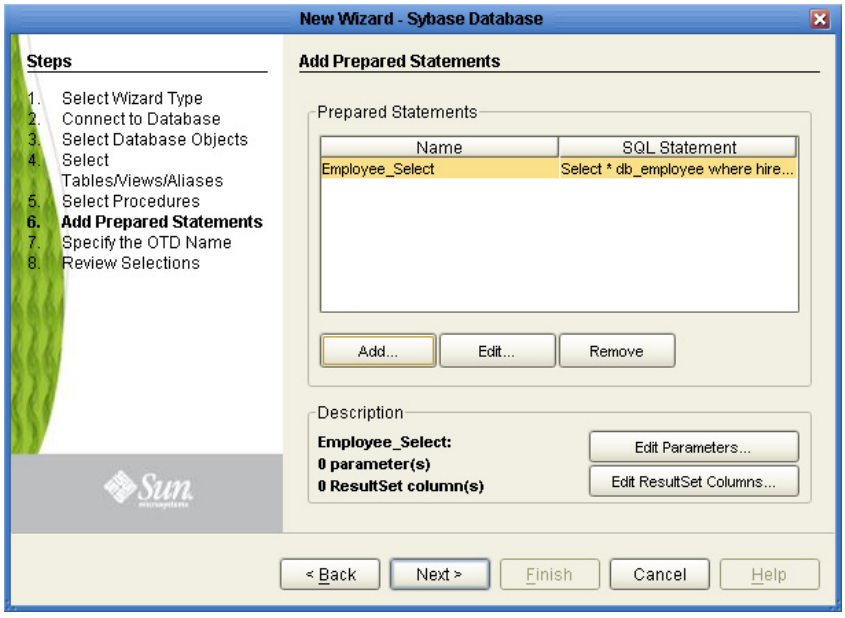
Figure 19 Prepared SQL Statement



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

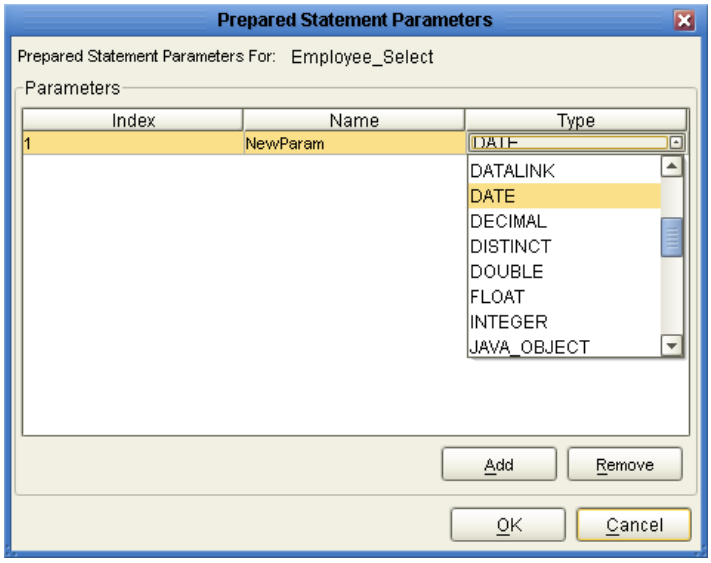
4 Click **Add** if you want to add additional parameters to the Statement or highlight a row and click **Remove** to remove it. Click **OK**. See **Figure 20**.

Figure 20 Edit the Prepared Statement Parameters



5 To edit Resultset Columns, click **Edit Resultset Columns**. The ResultSet Columns window appears, see **Figure 21**.

Figure 21 ResultSet Columns



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

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

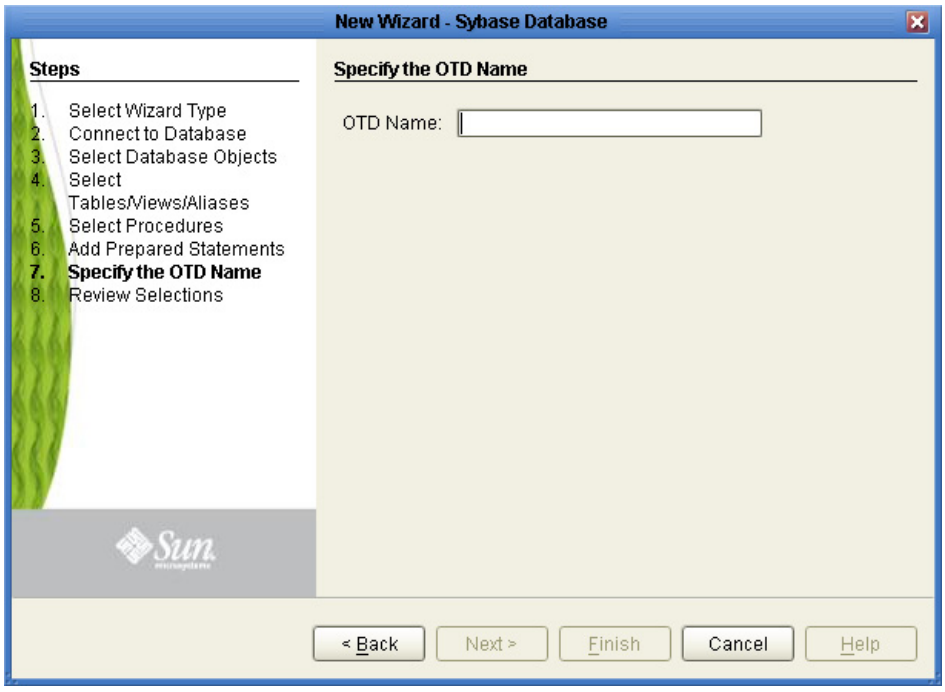
4.2.7 Specify the OTD Name

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

Steps Required to Specify the OTD Name:

1 Enter a name for the OTD. The OTD contains the selected tables and the package name of the generated classes. See **Figure 22**.

Figure 22 Naming an OTD



2 Click **Next**.

4.2.8 Review Selections

Review the selections made for the new OTD.

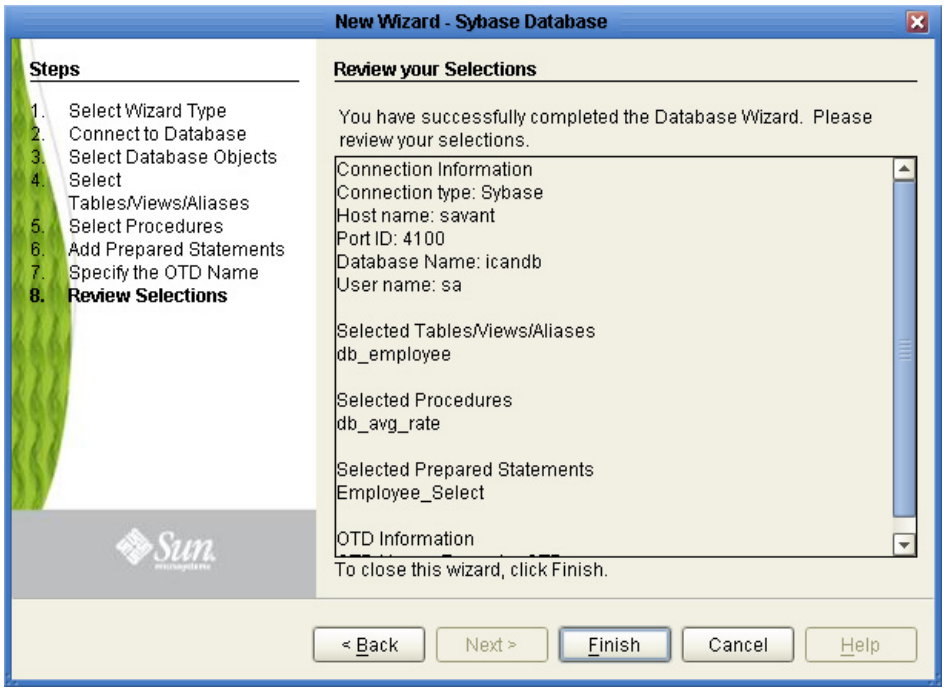
Steps Required to Review Your OTD Selections:

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

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

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

Figure 23 Database Wizard - Summary



4.3 Steps to Edit an Existing Sybase OTD

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

Steps to Edit the OTD from the Enterprise Designer Include:

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

   The Version Control - Check Out window appears.

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

   The Sybase Database Connection Information wizard appears.

4. Enter the connection information as described in **"Connect to Database" on page 35**, and click **Next**.
5. Step through each of the wizard steps and click **Finish** to save your changes.

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

Chapter 5

Using Sybase Operations

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

What's in This Chapter

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

5.1 Sybase eWay Database Operations (BPEL)

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

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

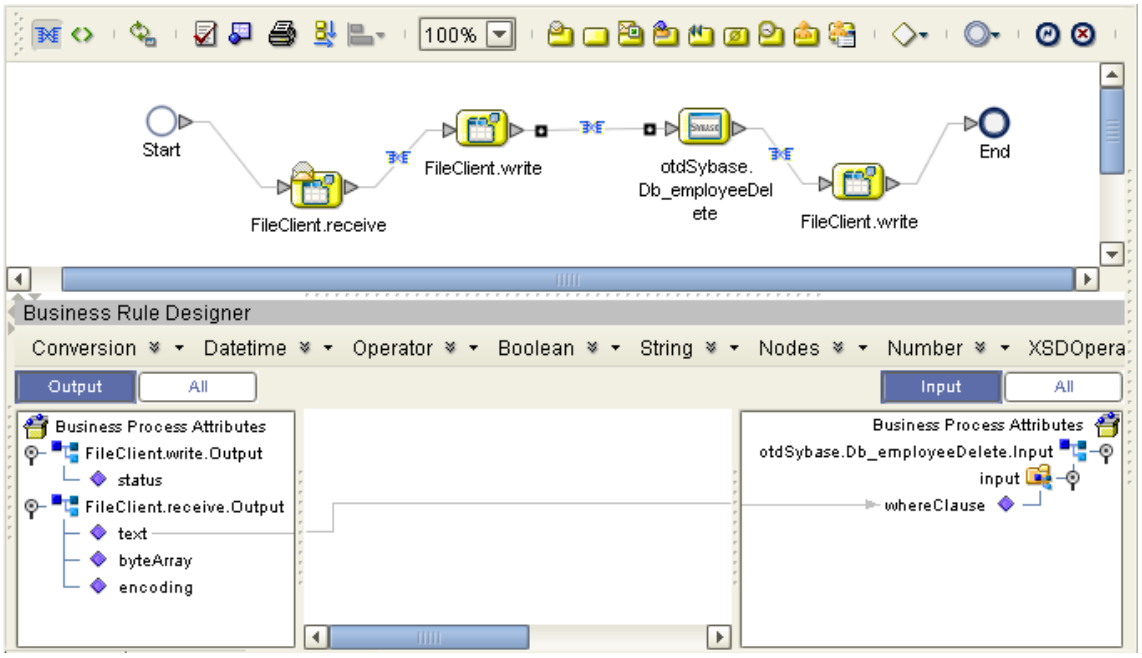
In addition to these outbound operations, the Sybase eWay also employs the inbound Activity **ReceiveOne** within a Prepared Statement OTD.

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

Figure 24 Input and Output Between Activities



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

Table 13 Sybase 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. For example: If the number of rows that meet the condition are 5 and the number of available rows are 10, then only 5 rows will be returned. Alternately, if the number of rows that meet the condition are 20, but if the number of available rows are 10, then only 10 rows are returned.
SelectOne	where() clause (optional)	Returns the first row that fits the condition of the where() clause.
Insert	definition of new item to be inserted	Returns status.

eInsight Operation	Activity Input	Activity Output
Update	where() clause	Returns status.
Delete	where() clause	Returns status.

5.2 Sybase 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 Sybase 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 prjSybase_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,
otdSybase.OtdSybaseOTD otdSybase_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Selecting records from db_employee
table via Table Select........" );
        FileClient_1.write();
        otdSybase_1.getDb_employee().select( input.getText() );
        while (otdSybase_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.shortToString(
otdSybase_1.getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdSybase_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdSybase_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate(
otdSybase_1.getDb_employee().getRATE().toString() );
            otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdSybase_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 prjSybase_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,
otdSybase.OtdSybaseOTD otdSybase_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() );
        otdSybase_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdSybase_1.getDb_employee().setEMP_NO(
typeConverter.stringToShort(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdSybase_1.getDb_employee().setLAST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdSybase_1.getDb_employee().setFIRST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdSybase_1.getDb_employee().setRATE( new
java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1
).getRate() ) );
            otdSybase_1.getDb_employee().setLAST_UPDATE(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdSybase_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 prjSybase_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,
otdSybase.OtdSybaseOTD otdSybase_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Updating the Rate and Last_update
fields .. " );
        FileClient_1.write();
        otdSybase_1.getDb_employee().update( input.getText() );
        while (otdSybase_1.getDb_employee().next()) {
            otdSybase_1.getDb_employee().setLAST_NAME( "Krishna" );
            otdSybase_1.getDb_employee().setFIRST_NAME( "Kishore" );
            otdSybase_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 prjSybase_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,
otdSybase.OtdSybaseOTD otdSybase_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Deleting record....." );
        FileClient_1.write();
        otdSybase_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)
- An outbound parameter (OUTGLOBALPRODUCTID)

These inbound and outbound parameters are generated by the DataBase Wizard and are represented in the resulting OTD as nodes. Within the Transformation Designer, you can drag values from the input parameters, execute the call, collect data, and drag the values to the output parameters.

Steps for executing the Stored Procedure include:

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

For example:

```
package Storedprocedure;

public class sp_jce
{

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

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

    public void receive(
com.stc.connector.appconn.file.FileTextMessage
input,com.stc.connector.appconn.file.FileApplication
```

```
FileClient_1,employeedb.Db_employee
employeedb_with_top_db_employee_1,insert_DB.Insert_DBOTD insert_DB_1
)
    throws Throwable
    {
employeedb_with_top_db_employee_1.unmarshalFromString(
input.getText() );

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

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

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

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

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

insert_DB_1.getInsert_new_employee().execute();

insert_DB_1.commit();

FileClient_1.setText( "procedure executed" );

FileClient_1.write();
    }

}
```

Manipulating the ResultSet and Update Count Returned by Stored Procedure

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

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

Sybase 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 the Sybase eWay Sample Projects

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

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

What's in This Chapter

- **About the Sybase 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 prjSybase_BPEL Sample Project** on page 64
- **Creating the prjSybase_JCD Sample Project** on page 88

6.1 About the Sybase eWay Sample Projects

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

- **prjSybase_JCD:** demonstrates how to select, insert, update, and delete data from a Sybase database using JCDs.
- **prjSybase_BPEL:** demonstrates how to select, insert, update, and delete data from a Sybase database using a BPEL business process.

Both the **prjSybase_JCD** and **prjSybase_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 **Sybase510_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 Sybase Sample Projects

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

- Insert
- Update
- Delete
- Select (BPEL “SelectAll” 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 Sybase 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 **SybaseeWayDocs.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 **Sybase_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

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

The sample Projects provided with the Sybase 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 package. To import a sample eWay Project to the Enterprise Designer do the following:

1 Extract the samples from the Suite Installer to a local file.

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

The following provides step-by-step instructions for manually creating the **prjSybase_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 82
- **Creating the Deployment Profile** on page 85
- **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 Enterprise Designer.

1 Start the Enterprise Designer.

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

3 Right-click **Project1** and select rename from the shortcut menu. Rename the Project (for this sample, **prjSybase_BPEL**).

6.5.2 Creating the OTDs

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

- Sybase Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

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

3 Enter the connection information for the Sybase 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 Sybase 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 icanuser1.db_employee where emp_no > ? Order
by emp_no
```

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

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

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

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

Steps required to create inbound and outbound DTD OTDs:

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

The New Object Type Definition Wizard window appears.

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

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

- otdInputDTD.dtd
- otdOutputDTD.dtd

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

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

6.5.3 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 ▪ otdSybase.DB_EMPLOYEEInsert (inside a Scope) ▪ otdInputDTD_DBemployees.unmarshal
bpUpdate	▪ FileClient.receive ▪ FileClient.write ▪ otdSybase.DB_EMPLOYEEUpdate ▪ FileClient.write
bpDelete	▪ FileClient.receive ▪ FileClient.write ▪ otdSybase.DB_EMPLOYEEDelete ▪ FileClient.write
bpPsSelect	▪ FileClient.receive ▪ FileClient.write ▪ otdSybase.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 ▪ otdSybase.DB_EMPLOYEESelectAll ▪ otdInputDTD_DBemployees.marshal ▪ FileClient.write ▪ FileClient.write

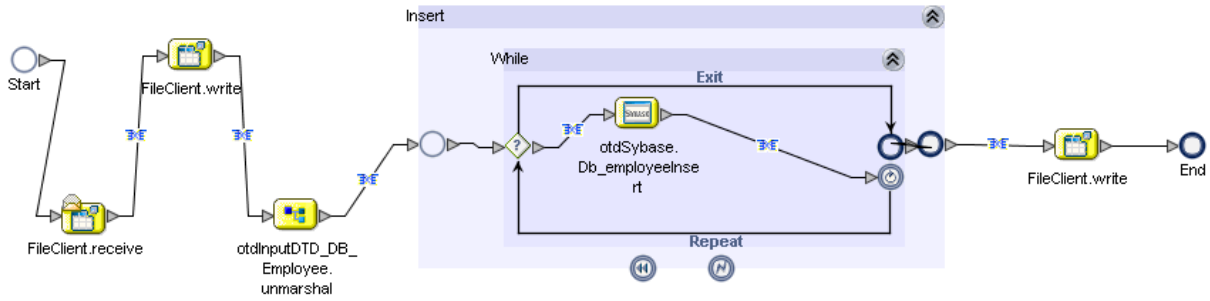
Configuring the bpInsert Modeling Elements

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 25 for an illustration of how all the modeling elements appear when connected.

Note: *Review the Sun SeeBeyond 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.*

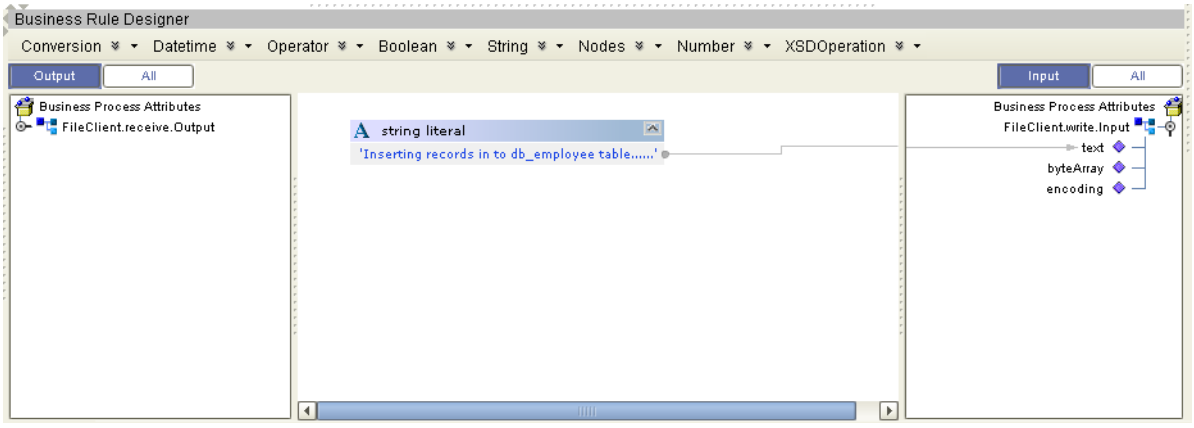
Figure 25 bpInsert Business Process



Steps required to configure the bpInsert business process:

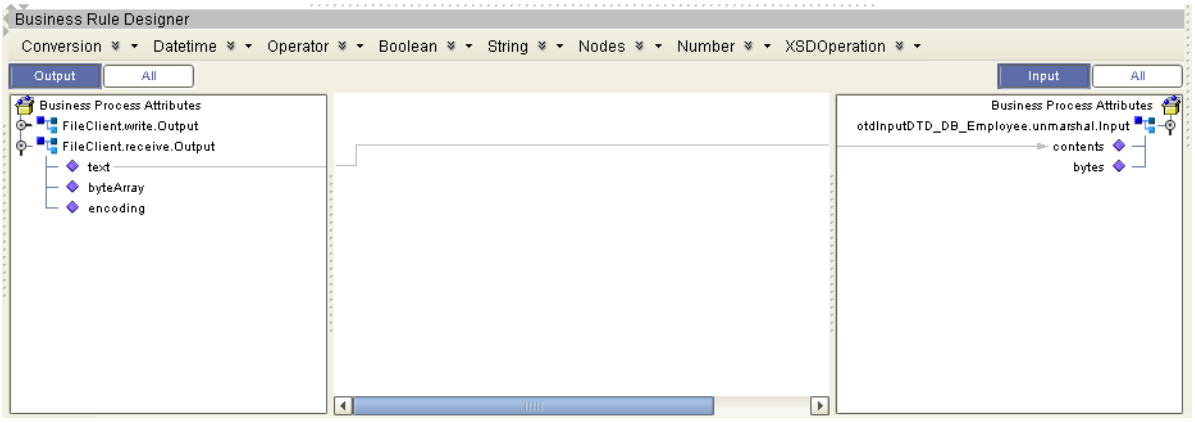
1 Configure the business rule between **FileClient.receive** and **FileClient.write** as seen in Figure 26.

Figure 26 bpInsert Business Rule # 1



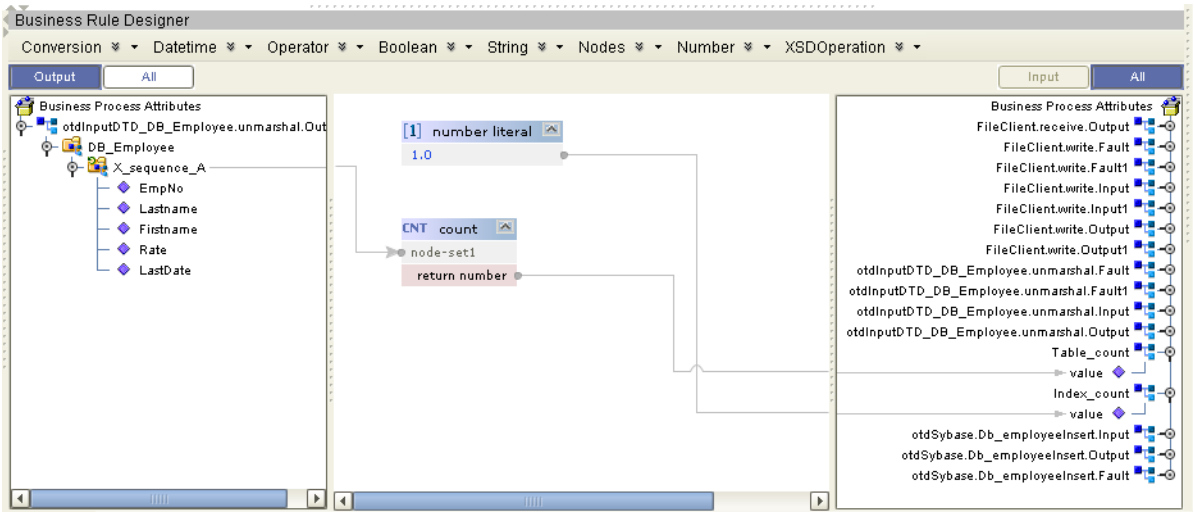
2 Configure the business rule between the **FileClient.write** Activity and **otdInputDTD_DBemployees.unmarshal** Activity as seen in Figure 27.

Figure 27 bpInsert Business Rule # 2



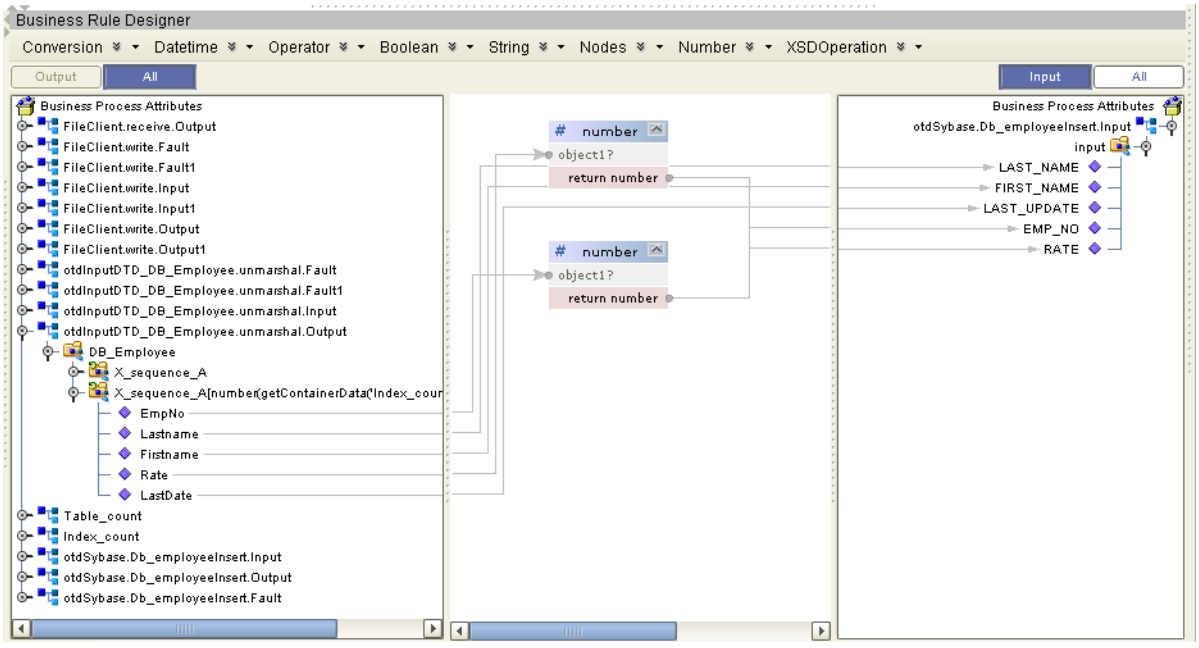
3 Configure the business rule between **otdInputDTD_DBemployees.unmarshal** and the **Insert** (Scope element).

Figure 28 bpInsert Business Rule # 3



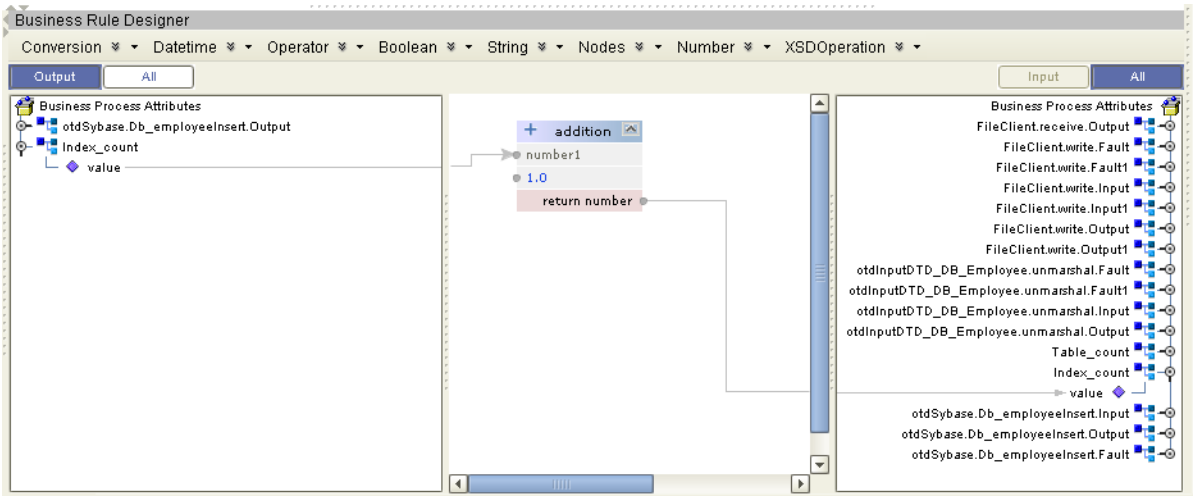
4 Configure the business rule in the **While** statement that connects to the **otdSybase.DB_EMPLOYEEInsert** Activity.

Figure 29 bpInsert Business Rule # 4



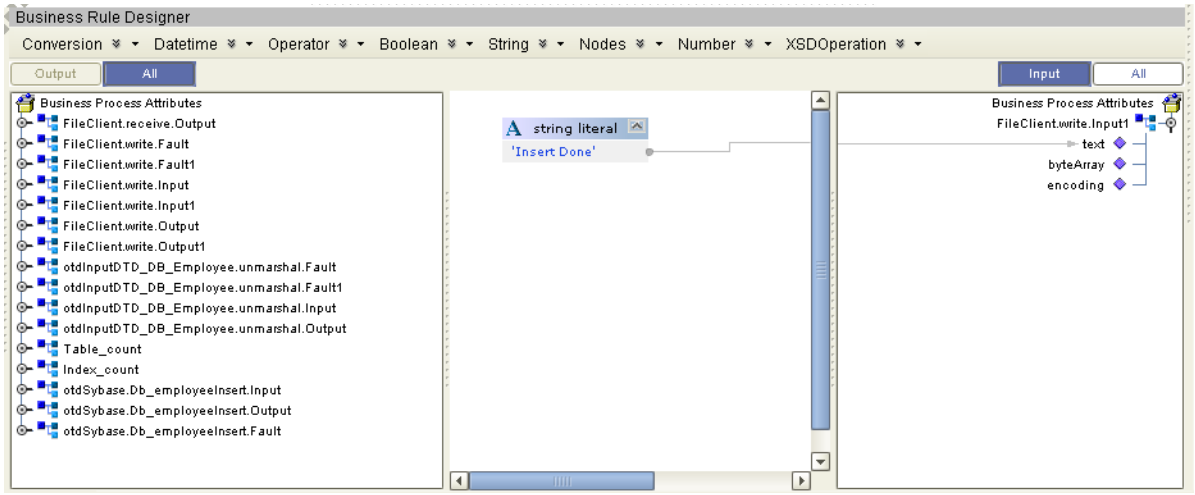
5 Configure the business rule in the **While** statement that connects from the **otdSybase.DB_EMPLOYEEInsert** Activity.

Figure 30 bpInsert Business Rule # 5



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

Figure 31 bpInsert Business Rule # 6



Configuring the bpUpdate Modeling Elements

The bpUpdate business process describes how to update a record in the Sybase 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 32 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.*

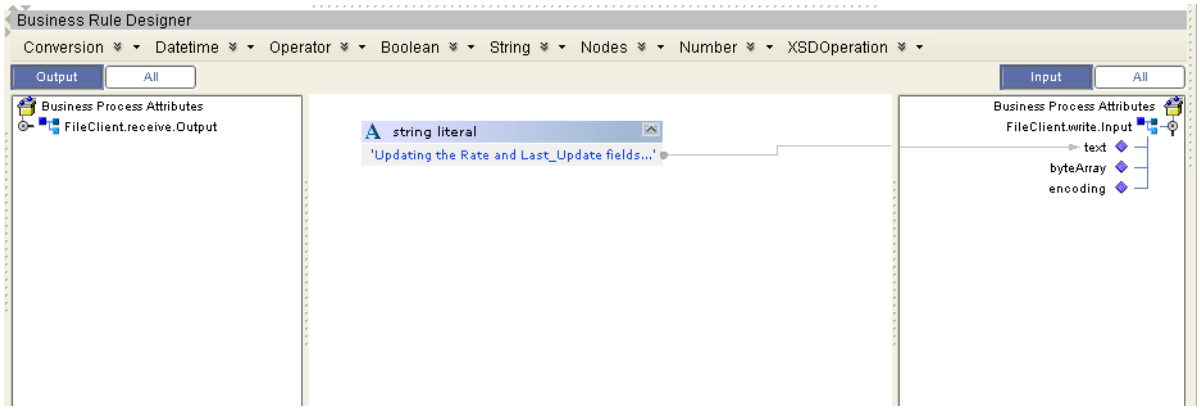
Note: *Review the eInsight Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

Figure 32 bpUpdate Business Process

Steps required to configure the bpUpdate business process:

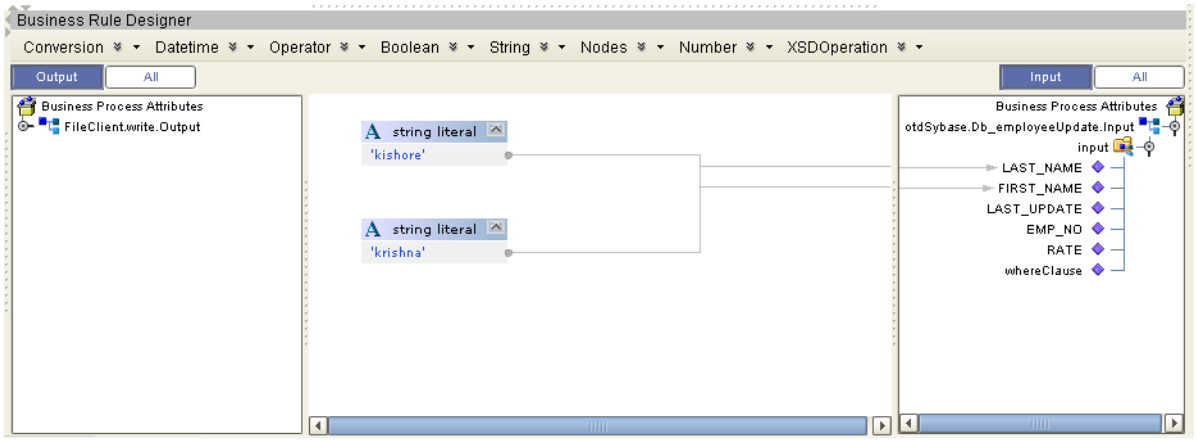
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 33.

Figure 33 bpUpdate Business Rule # 1



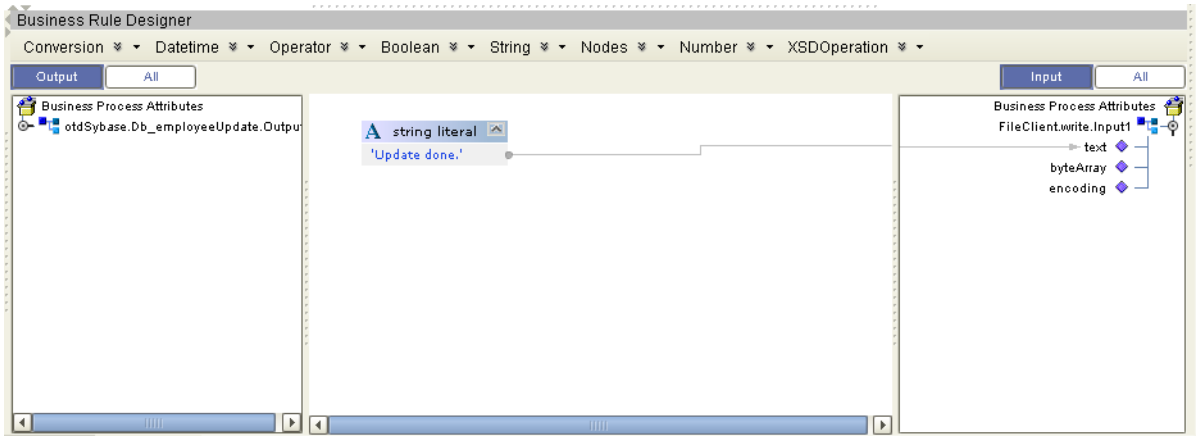
2 Configure the business rule between the **FileClient.write** Activity and **otdSybase.DB_EMPLOYEEUpdate** Activity as seen in Figure 34.

Figure 34 bpUpdate Business Rule # 2



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

Figure 35 bpUpdate Business Rule # 3



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

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

Configuring the bpDelete Modeling Elements

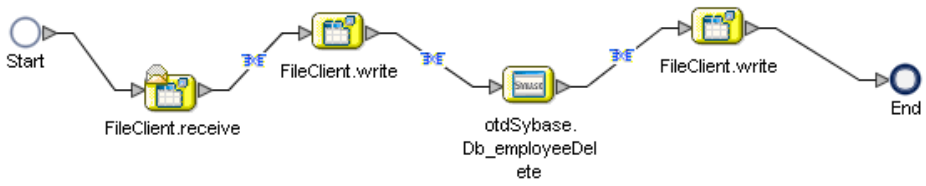
The bpDelete business process describes how to delete a record in the Sybase database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Delete operation. See Figure 36 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 Sun SeeBeyond 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.*

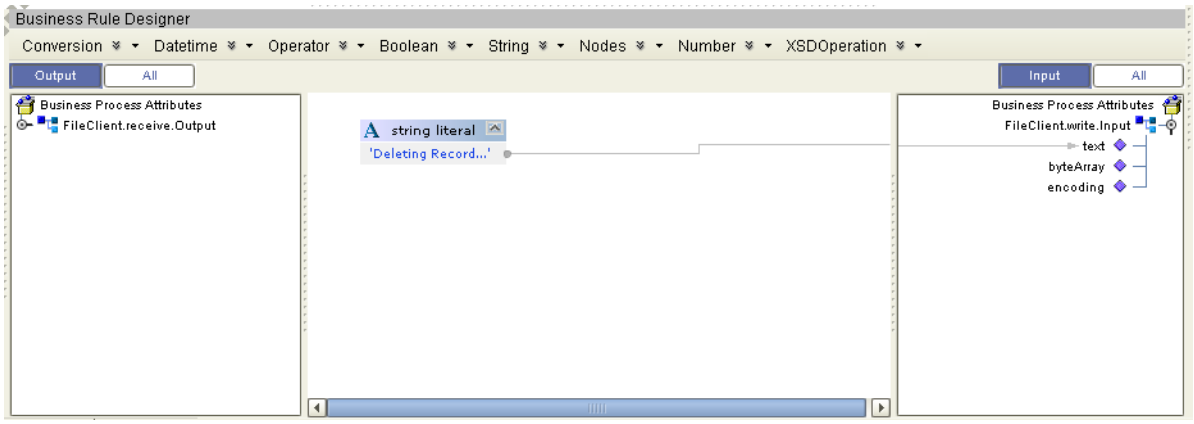
Figure 36 bpDelete Business Process



Steps required to configure the bpDelete business process:

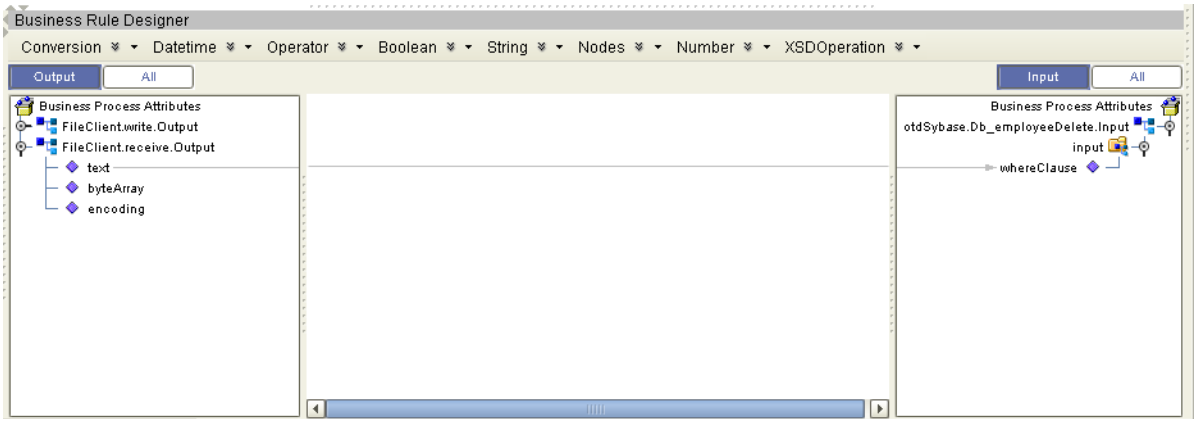
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 37.

Figure 37 bpDelete Business Rule # 1



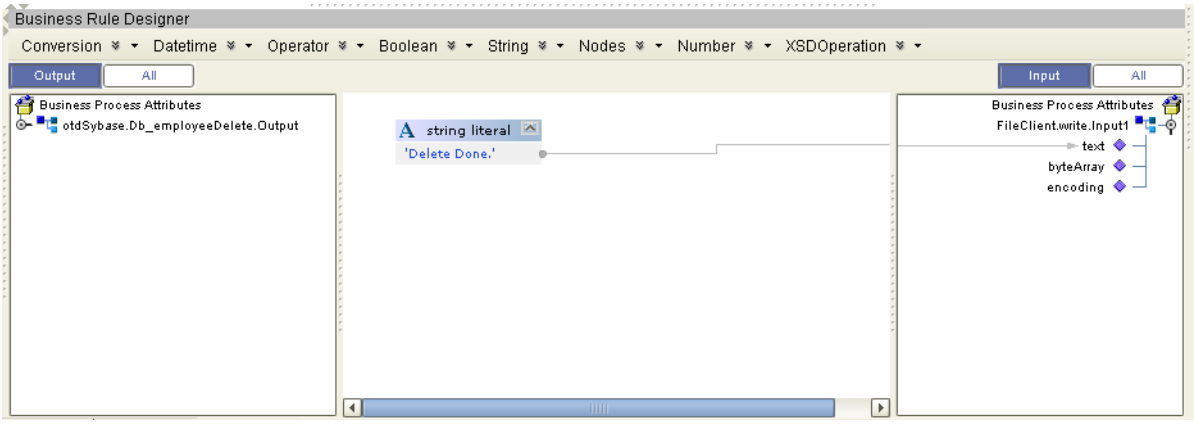
2 Configure the business rule between the **FileClient.write** Activity and **otdSybase.DB_EMPLOYEEDelete** Activity as seen in Figure 38.

Figure 38 bpDelete Business Rule # 2



3 Configure the business rule between the **otdSybase.DB_EMPLOYEEDelete** Activity and the **FileClient.write** Activity as seen in Figure 39.

Figure 39 bpDelete Business Rule # 3



Configuring the bpTableSelect Modeling Elements

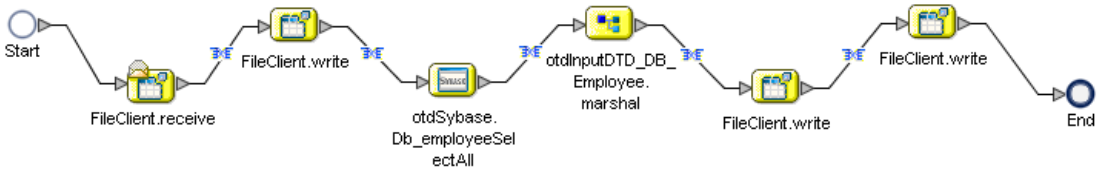
The bpTableSelect business process is describes how to select all records the Sybase database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 40 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 Sun SeeBeyond eInsight™ Business Process Manager User's Guide for a more detailed description of the steps required to connect and add business rules to a modeling elements in a business process.*

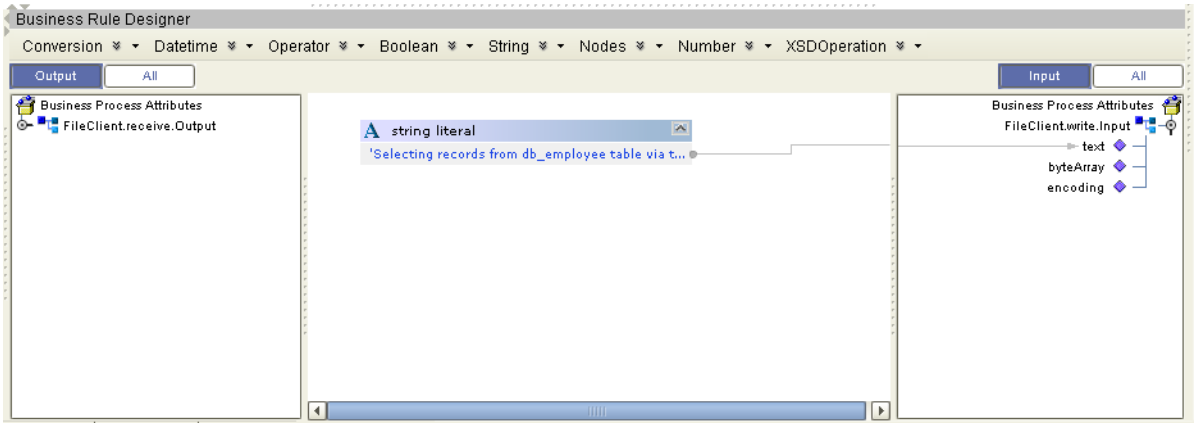
Figure 40 bpTableSelect Business Process



Steps required to configure the bpTableSelect business process:

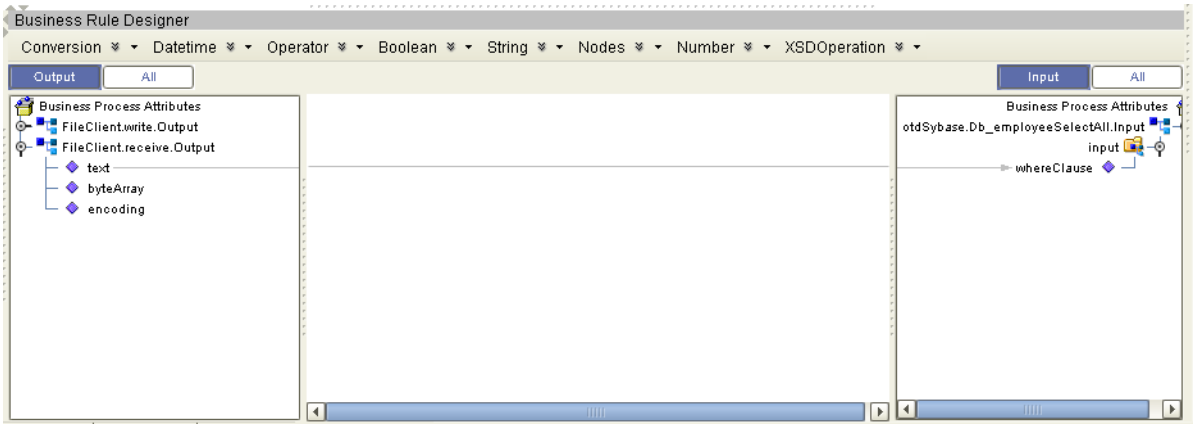
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 41.

Figure 41 bpTableSelect Business Rule # 1



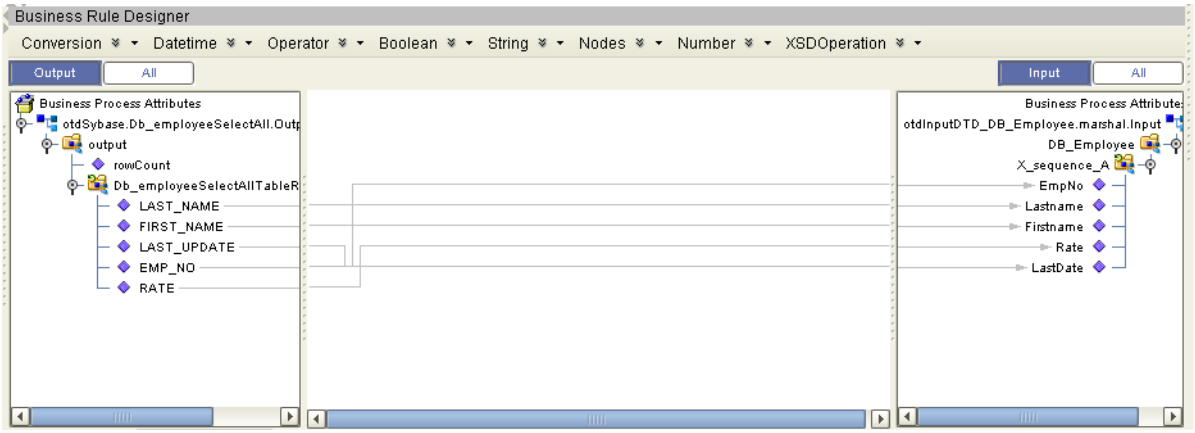
2 Configure the business rule between the **FileClient.write** Activity and **otdSybase.DB_EMPLOYEESelectAll** Activity as seen in Figure 42.

Figure 42 bpTableSelect Business Rule # 2



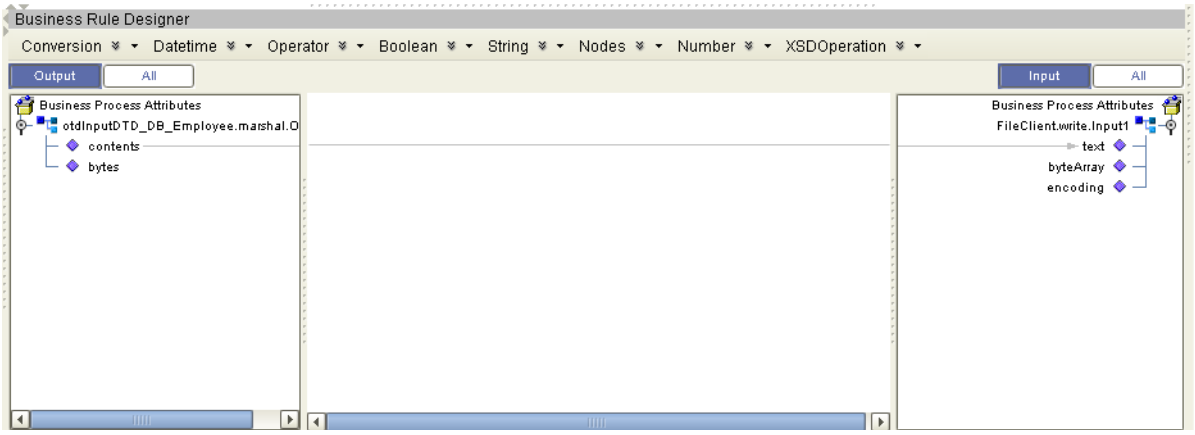
3 Configure the business rule between the **otdSybase.DB_EMPLOYEESelectAll** Activity and the **otdInputDTD_DBemployees.marshal** Activity as seen in Figure 43.

Figure 43 bpSelectTable Business Rule # 3



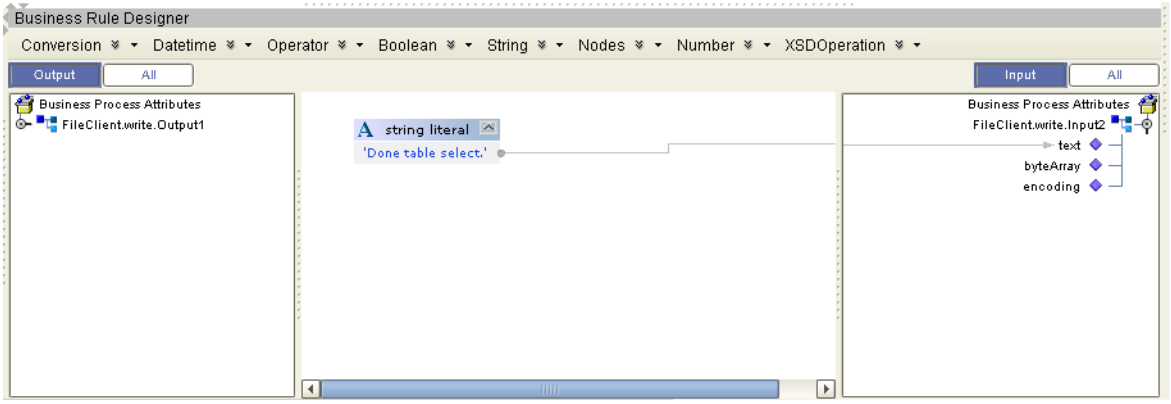
4 Configure the business rule between the **otdInputDTD_DBemployees.marshal** Activity and the **FileClient.write** Activity as seen in Figure 44.

Figure 44 bpTableSelect Business Rule # 3



5 Configure the business rule between the **FileClient.write** Activity and the **FileClient.write** Activity as seen in Figure 45.

Figure 45 bpTableSelect Business Rule # 3



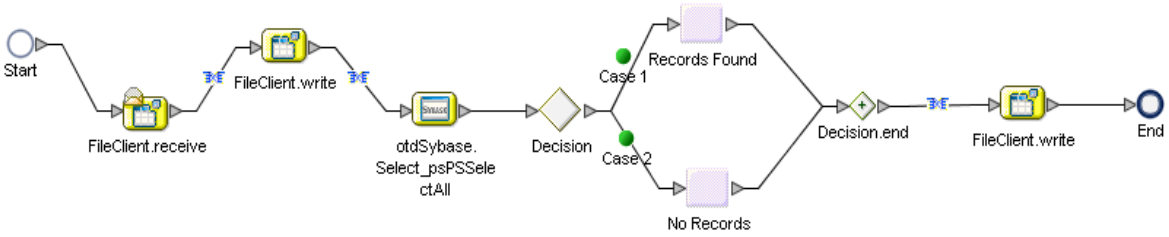
Configuring the bpPsSelect Modeling Elements

The bpPsSelect business process describes how to use a Prepared Statement query to select all records in the Sybase 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 46 for an illustration of how all the modeling elements appear when connected.

Note: *Review the Sun SeeBeyond 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.*

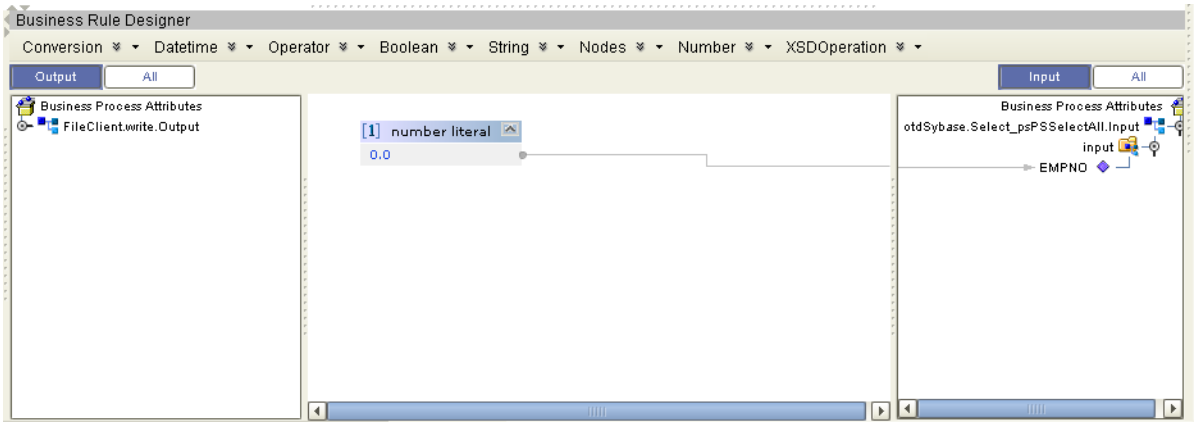
Figure 46 bpPsSelect Business Process



Steps required to configure the bpPsSelect business process:

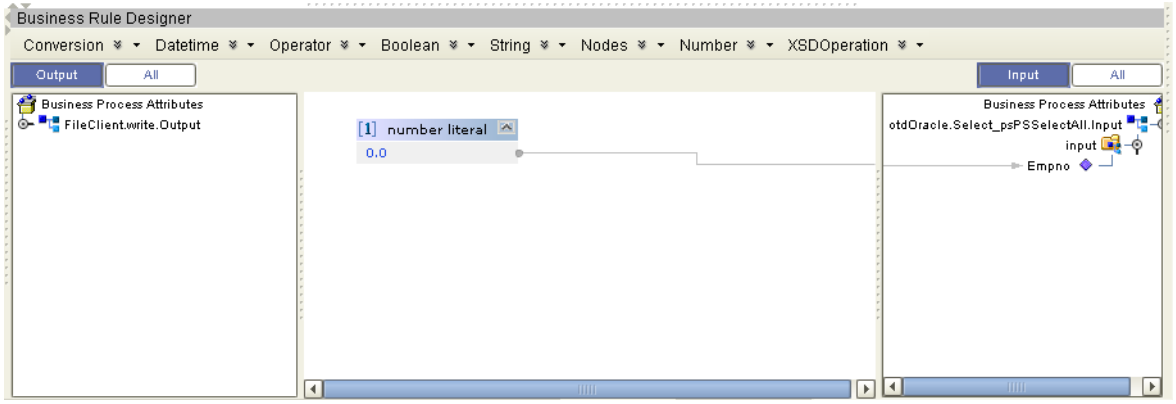
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** as seen in Figure 41.

Figure 47 bpSelectTable Business Rule # 1



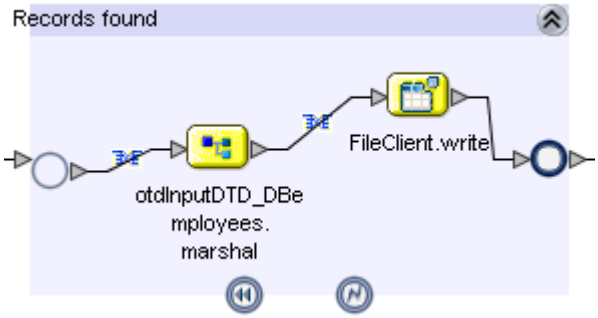
2 Configure the business rule between **FileClient.write** and **otdSybase.Select_psPSSelectAll** as seen in Figure 48.

Figure 48 bpSelectTable Business Rule # 2



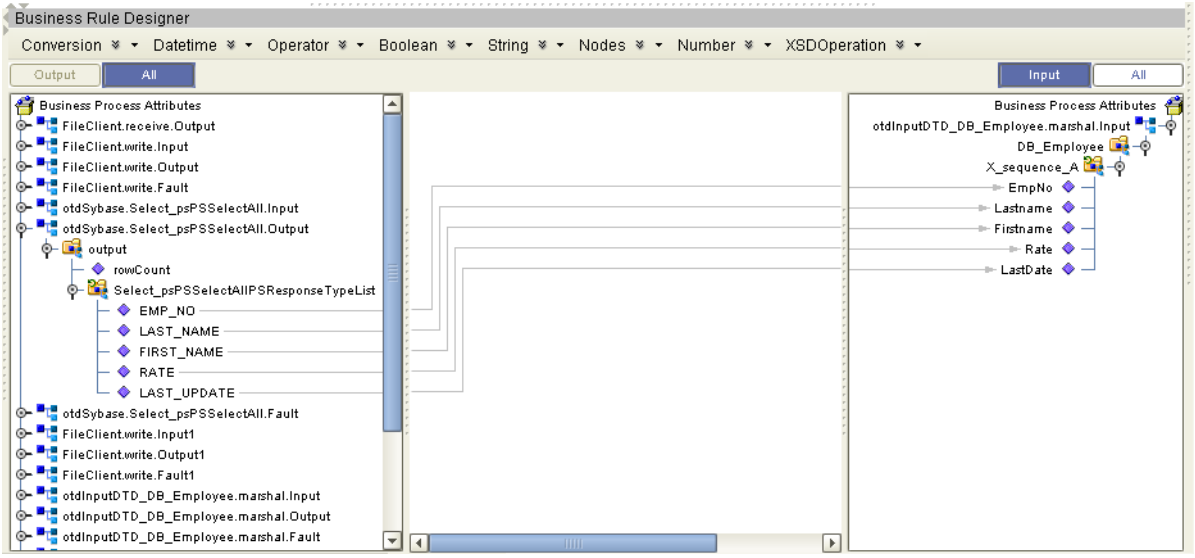
3 Configure **Case 1** of the Decision branching activity. This requires adding business rules between the **otdInputDTD_DBemployees.marshal** and the **FileClient.write** activities within the Scope element.

Figure 49 Activities within Case 1 Scope



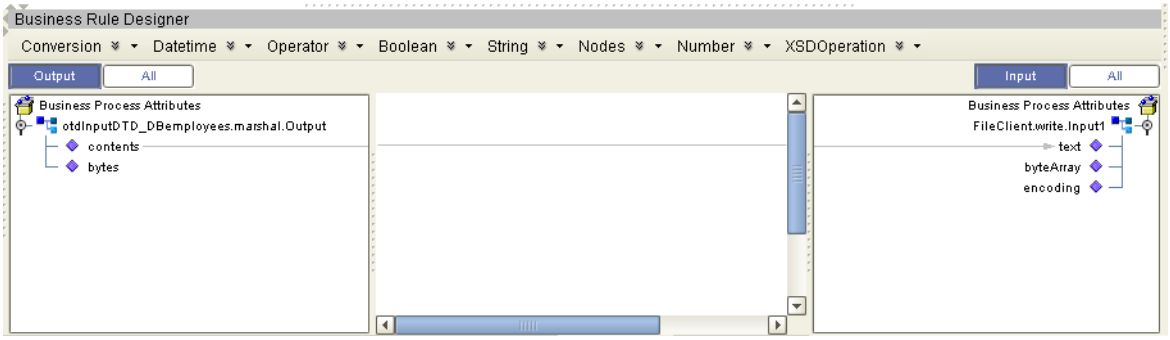
4 Configure the business rule between the start of the Scope element in **Case 1** and the **otdInputDTD_DBemployees.marshal** activity, as seen in Figure 50.

Figure 50 Case 1 Scope Business Rule # 3



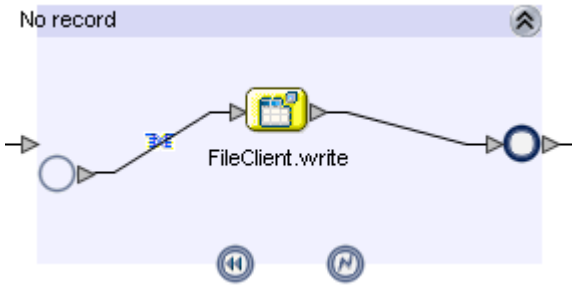
5 Configure the business rule between **otdInputDTD_DBemployees.marshal** and **FileCleint.write** in the Scope element, as seen in Figure 51.

Figure 51 Case 1 Scope Business Rule # 4



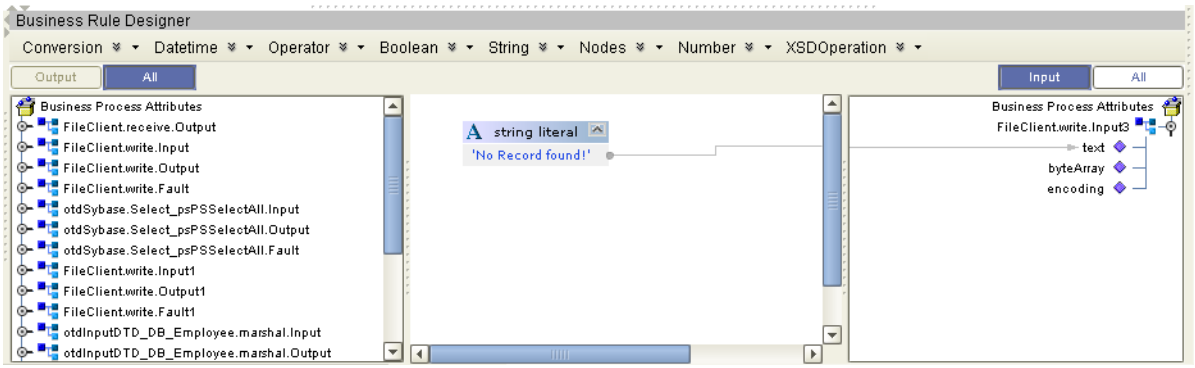
6 Configure Case 2 of the Decision branching activity. This requires adding business rules between the **otdInputDTD_DBemployees.marshal** and the **FileClient.write** activities within the Scope element.

Figure 52 Activities within Case 2 Scope



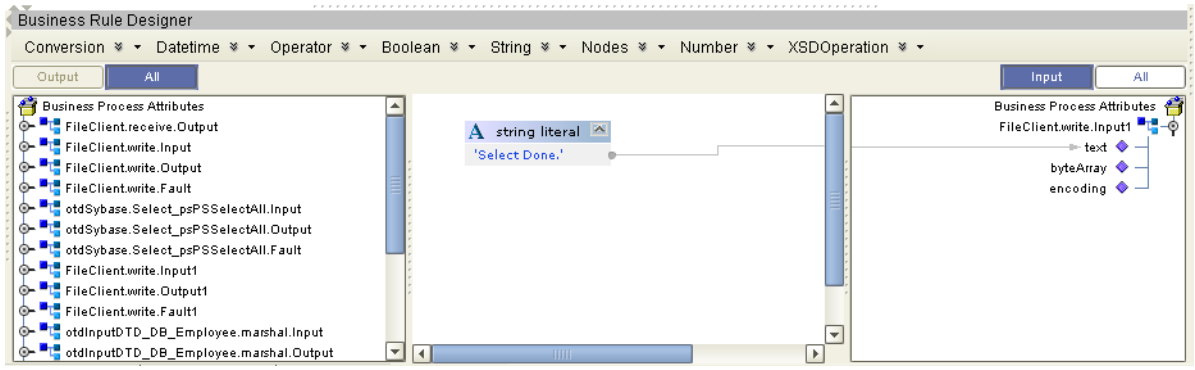
7 Configure the business rule between the start of the Scope element in **Case 2** and the **FileClient.Write** activity, as seen in Figure 53.

Figure 53 Case 2 Scope Business Rule # 5



8 Configure the business rule between the **Decision.end** Element and the **FileClient.write** Activity, as seen in Figure 54.

Figure 54 bpSelectTable Business Rule # 6



6.5.4 Creating the Connectivity Map

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

Steps required to create the Connectivity Map:

1. From the Project Explorer tree, right-click the new **prjSybase_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 **prjSybase_BPEL** sample Project requires the following components:

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

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

To Select a Sybase 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, **Sybase** 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

- Sybase1 to eaSybaseOUT

To Select a Sybase 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

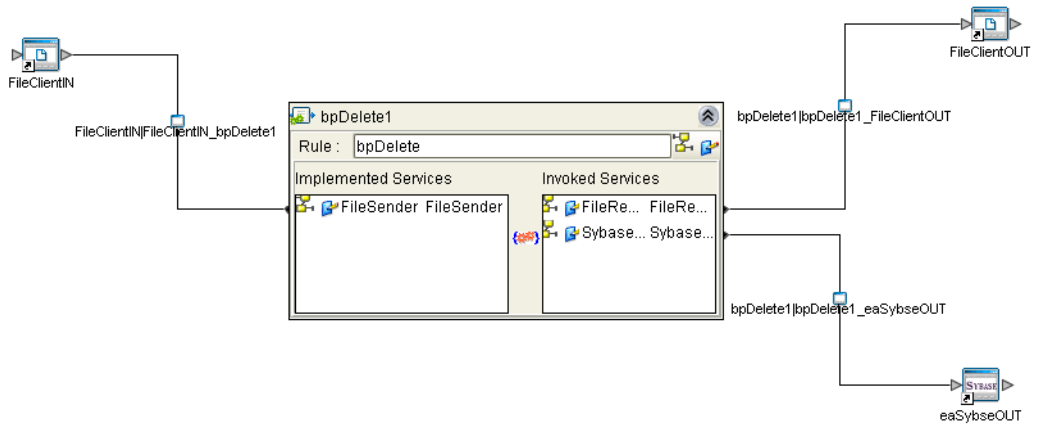
Binding the eWay Components

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 **Sybase_otdSybase** (under Invoked Services) to the **eaSybaseOUT** External Application.
4. From the **bpDelete** Binding dialog box, map **FileReceiver** to the **FileClientOUT** External Application, as seen in Figure 55.

Figure 55 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.

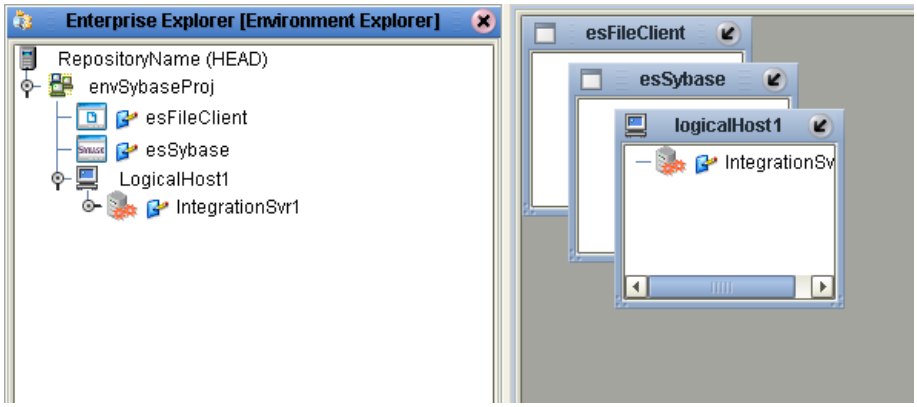
6.5.5 Creating an Environment

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.

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 **envSybaseProj**.
4. Right-click **envSybaseProj** and select **New Sybase External System**. Name the External System **esSybase**. Click **OK**. **esSybase** is added to the Environment Editor.
5. Right-click **envSybaseProj** and select **New File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.
6. Right-click **envSybaseProj** 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 56).

Figure 56 Environment Editor - envSybaseProj



8. Save your current changes to the Repository.

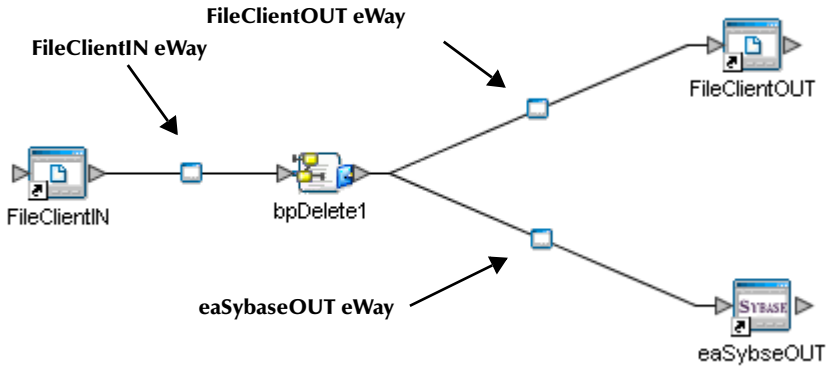
6.5.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the The **prjSybase_BPEL**

sample Project use three eWays that are represented as a nodes between the External Applications and the Business Process, as seen in Figure 57.

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

Figure 57 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 14. Click **OK** to close the Properties Editor.

Table 14 FileClientIN eWay Property Settings

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

2 Double-click the **FileClientIN eWay on each of the Connectivity Maps and** modify the properties for your system, as seen in Table 15. Click **OK** to close the Properties Editor.

Table 15 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

Connectivity Map	Property Name	Required Value
cmTableSelect	Output file name	BPEL_TableSelect_output%d.dat.in
cmUpdate	Output file name	BPEL_Update_output%d.dat

Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

1. From the **Environment Explorer** tree, right-click the File External System (**esSybase** in this sample), and select **Properties**. The Properties Editor opens to the Sybase eWay Environment configuration.
2. Modify the Sybase eWay Environment configuration properties for your system, as seen in Table 16, and click **OK**.

Table 16 Sybase eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound Sybase eWay > JDBC Connector settings	ServerName	Enter the host name of the database server being used. ***Note:*** The Sybase eWay does not support using "localhost" as the server name.
	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 Sybase eWay Environment configuration.
4. Modify the File eWay Environment configuration properties for your system, as seen in Table 17, and click **OK**.

Table 17 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

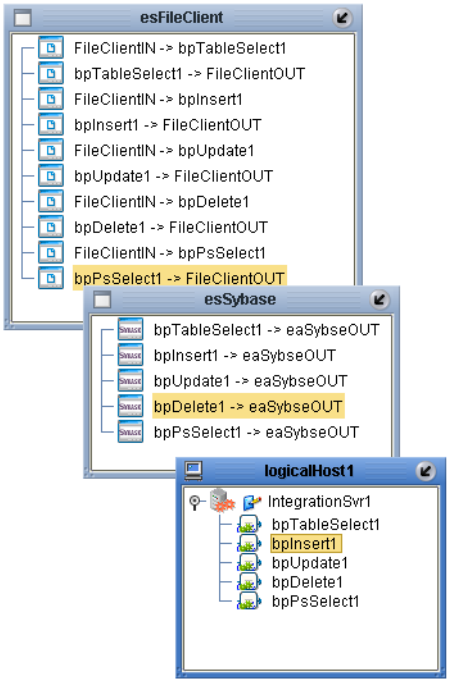
6.5.7 Creating the Deployment Profile

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 **prjSybase_BPEL** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpSybase_BPEL**). Select **envSybaseProj** 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 58.

Figure 58 Deployment Profile



6.5.8 Creating and Starting the Domain

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

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

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your 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 Sun SeeBeyond eGate™ Integrator System Administration Guide.*

6.5.9 Building and Deploying the Project

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

Build the Project

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

Deploy the Project

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

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*".

6.5.10 Running the Sample Project

Additional steps are required to run the deployed sample Project.

Steps required to run the sample Project:

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

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

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

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

   - TriggerTableSelect.in
   - TriggerDelete.in
   - 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 Sybase 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 prjSybase_JCD Sample Project

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

Steps required to create the sample project include:

- **Creating a Project** on page 88
- **Creating the OTDs** on page 88
- **Creating a Connectivity Map** on page 90
- **Creating the Collaboration Definitions (Java)** on page 91
- **Binding the eWay Components** on page 101
- **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, **prjSybase_BPEL**).

6.6.2 Creating the OTDs

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

- Sybase Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a Sybase 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 **Sybase Database OTD Wizard** from the list of OTD Wizards and click **Next**.
3. Enter the connection information for the Sybase database. Connection fields include:
   - Host name:
   - Port ID:
   - SID:
   - User name:
   - Password:
4. Click **Next**, and select the types of database object you want to include in the sample Project. For our example, select the following:
   - Tables/Views/Aliases
   - Prepared Statements
5. Click **Add** to select tables from the Sybase 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 icanuser1.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 **otdSybase**.
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 **prjSybase_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 **prjSybase_JCD** sample Project requires the following components:

- File External Application (2)

- Sybase External Application
- Service

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

Steps required to select a Sybase 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, **Sybase** 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
   - Sybase1 to eaSybaseOUT
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 **prjSybase_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 **prjSybase_JCD** > **otdALL** > **otdSybase**. The **otdSybase** 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 **prjSybase_JCD** > **otdALL** > **otdSybase**. The **otdSybase** 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 **prjSybase_JCD** > **otdALL** > **otdSybase**. The **otdSybase** 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 **prjSybase_JCD** > **otdALL** > **otdSybase**. The **otdSybase** 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 **prjSybase_JCD > otdALL > otdSybase**. The **otdSybase** 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.

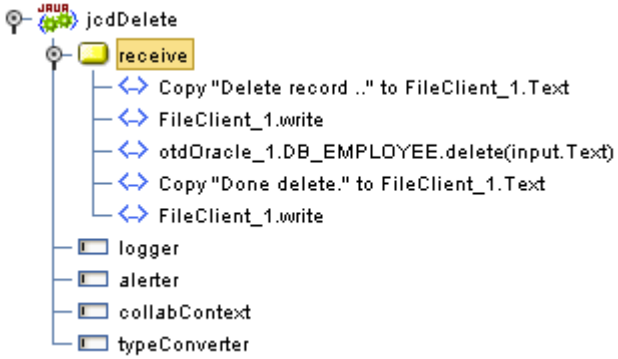
Creating the jcdDelete Business Rules

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 59.

Figure 59 jcdDelete Business Rules

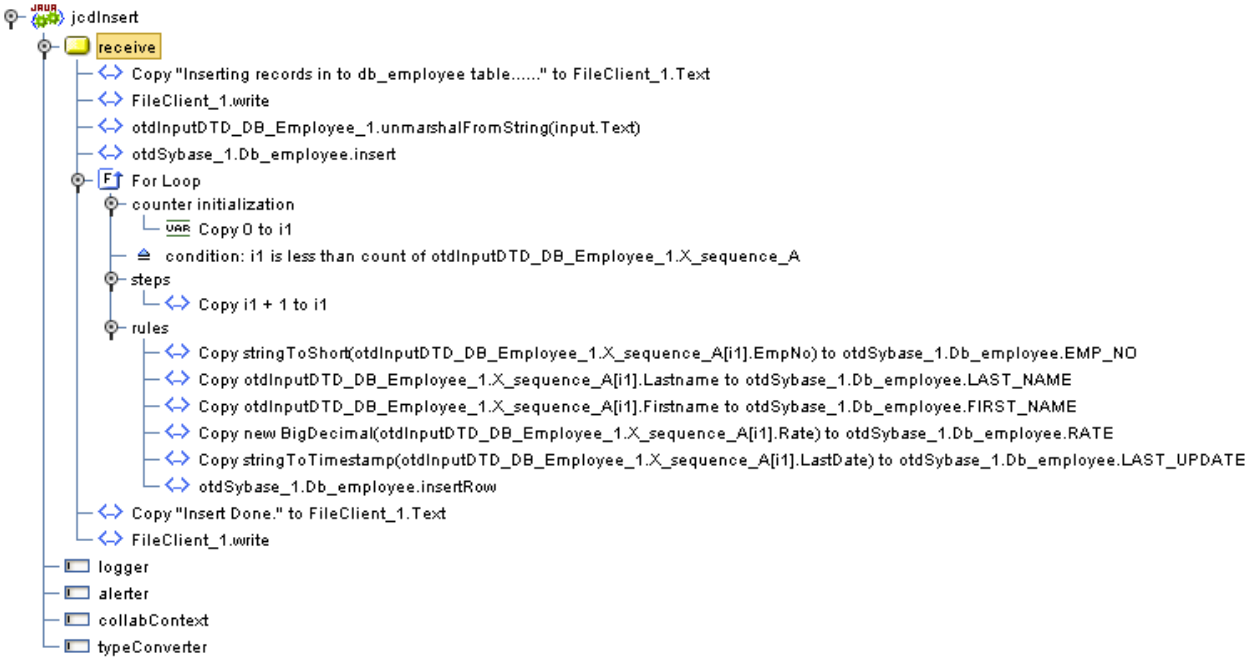


Creating the jcdInsert Business Rules

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 **otdSybase** 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 60.

Figure 60 jcdInsert Business Rules



Sample code from the jcdInsert Includes:

```
package prjSybase_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, otdSybase.OtdSybaseOTD otdSybase_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.

        otdSybase_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdSybase_1.getDb_employee().setEMP_NO(
typeConverter.stringToShort(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
false, 0 ) );
            otdSybase_1.getDb_employee().setLAST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdSybase_1.getDb_employee().setFIRST_NAME(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdSybase_1.getDb_employee().setRATE( new
java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1
).getRate() ) );
            otdSybase_1.getDb_employee().setLAST_UPDATE(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-
MM-dd hh:mm:ss", false, "" ) );
            otdSybase_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 61.

Figure 61 jcdPsSelect

Sample code from the jcdPsSelect Includes:

```
package prjSybase_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,
otdSybase.OtdSybaseOTD otdSybase_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();
        otdSybase_1.getSelect_ps().setEMPNO(
typeConverter.stringToShort( "0", "#", false, 0 ) );
        otdSybase_1.getSelect_ps().executeQuery();
        if (otdSybase_1.getSelect_ps().resultsAvailable()) {
            while
(otdSybase_1.getSelect_ps().get$Select_psResults().next()) {
                otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.intToString(
otdSybase_1.getSelect_ps().get$Select_psResults().getEMP_NO(), "#",
false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastname(
otdSybase_1.getSelect_ps().get$Select_psResults().getLAST_NAME() );
                otdOutputDTD_DB_Employee_1.setFirstname(
otdSybase_1.getSelect_ps().get$Select_psResults().getFIRST_NAME() );
                otdOutputDTD_DB_Employee_1.setRate(
typeConverter.doubleToString(
otdSybase_1.getSelect_ps().get$Select_psResults().getRATE(),
"#.000000;-#.000000", false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdSybase_1.getSelect_ps().get$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 62.

Figure 62 jcdTableSelect

Sample code from the jcdTableSelect Includes:

```
package prjSybase_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, otdSybase.OtdSybaseOTD otdSybase_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.

        otdSybase_1.getDb_employee().select( input.getText() );
        while (otdSybase_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.shortToString(
otdSybase_1.getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdSybase_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdSybase_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate(
otdSybase_1.getDb_employee().getRATE().toString() );
```

```
            otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdSybase_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();
    }
}
```

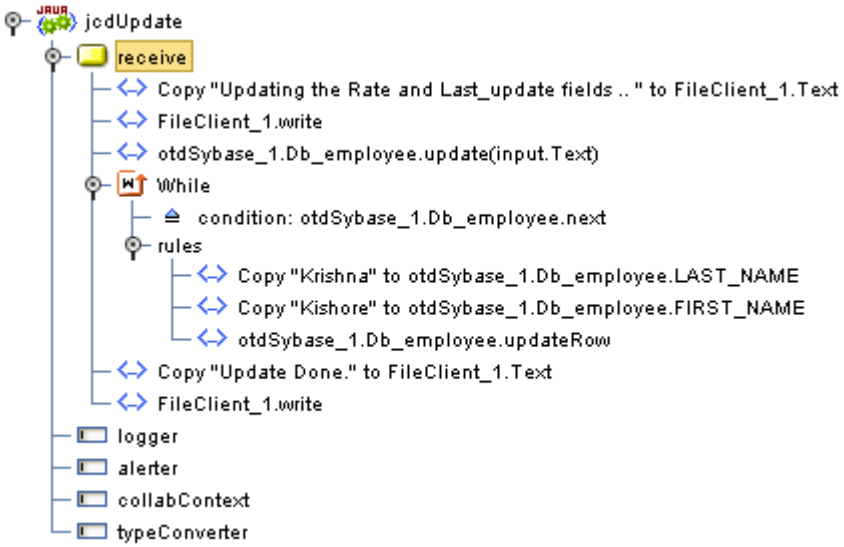
Creating the jcdUpdate Business Rules

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 63.

Figure 63 jcdTableUpdate



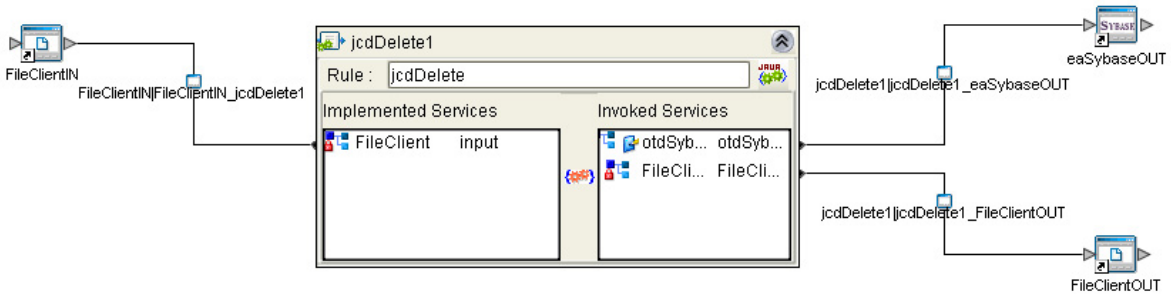
6.6.6 Binding the eWay Components

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 Designer's 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 **otdSybase_1** (under Invoked Services) to the **eaSybaseOUT** External Application.
6. From the **bpDelete** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 55.

Figure 64 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.

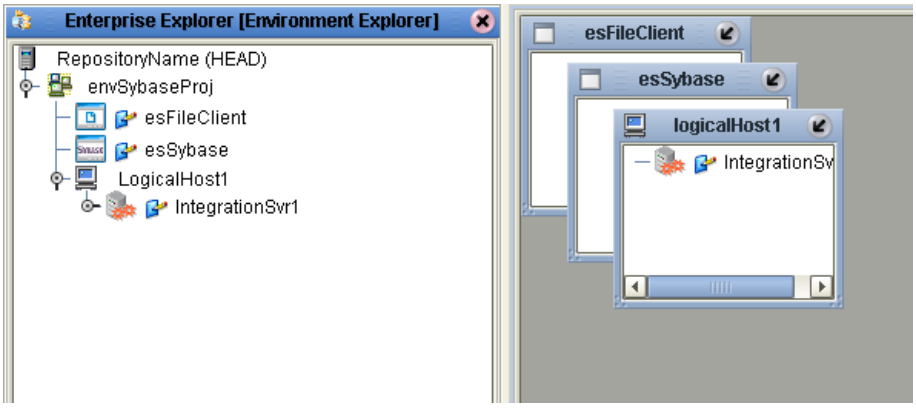
6.6.7 Creating an Environment

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 **envSybaseProj**.
4. Right-click **envSybaseProj** and select **New Sybase External System**. Name the External System **esSybase.** Click **OK**. **esSybase** is added to the Environment Editor.
5. Right-click **envSybaseProj** and select **New File External System**. Name the External System **esFileClient**. Click **OK**. **esFileClient** is added to the Environment Editor.
6. Right-click **envSybaseProj** 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 56).

Figure 65 Environment Editor - envSybaseProj



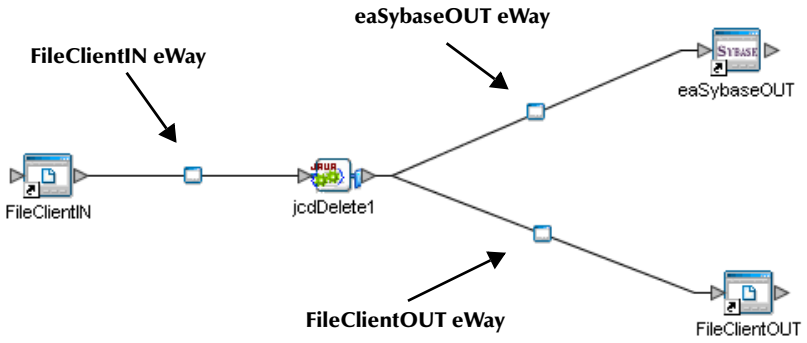
8. Save your current changes to the Repository.

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 **prjSybase_BPEL** sample Project uses three eWays that are represented as nodes between the External Applications and the Business Process, as seen in Figure 57.

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

Figure 66 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 18. Click **OK** to close the Properties Editor.

Table 18 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 19. Click **OK** to close the Properties Editor.

Table 19 FileClientOUT eWay Property Settings

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

Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

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

Table 20 Sybase eWay Environment Properties

Section	Property Name	Required Value
Configuration > Inbound Sybase eWay > JDBC Connector settings	ServerName	Enter the name of the database server being used. ***Note:*** The Sybase eWay does not support using "localhost" as the server name.
	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 Sybase eWay Environment configuration.
4. Modify the File eWay Environment configuration properties for your system, as seen in Table 21, and click **OK**.

Table 21 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
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

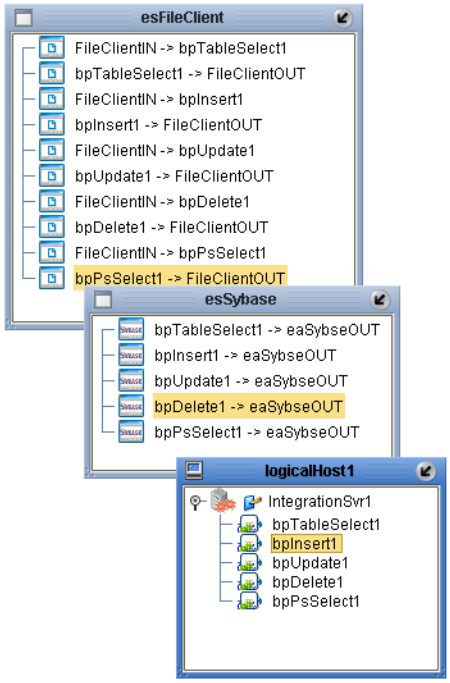
6.6.9 Creating the Deployment Profile

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 **prjSybase_BPEL** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpSybase_BPEL**). Select **envSybaseProj** 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 58.

Figure 67 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.

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

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your 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.

2 Verify the output data by viewing the sample output files. See **About the Sybase 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 Sybase 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 22 lists common datatype conversions used for the Sybase Server.

Table 22 The Sybase eWay Datatype Conversions

Sybase 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

Sybase 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	BigDecimal: **java.math.BigDecimal(String)**	2456.95
Smallmoney	BigDecimal	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 **44**
Automap **85**, **105**

B

binding
  dialog box **81**, **101**
book **109**
BPEL operations **49**
  Delete **49**
  Insert **49**
  ReceiveOne **49**
  SelectAll **49**
  SelectMultiple **49**
  SelectOne **49**
  Update **49**

C

Collaboration
  editor **91**
configuring Sybase eWay **18**
connection retry settings, outbound **28**
connection retry settings, outbound XA **31**, **33**
Connection Retry Support **8**
Connectivity Map Generator **8**
conventions, text **9**

D

database operations (BPEL) **49**
  activity input and output **49**
  Delete **49**
  Insert **49**
  ReceiveOne **49**
  SelectAll **49**
  SelectMultiple **49**
  SelectOne **49**
  Update **49**
database operations (JCD) **51**
  delete(String sWhere) **51**
  deleteRow() **51**
  insert() **51**
  insertRow() **51**
  select(String where) **51**
  update(String sWhere) **51**
  updateRow() **51**
database OTD wizard
  about **34**
  add prepared statement **44**
  connect to database **35**
  editing existing OTDs **48**
  review selections **47**
  select database objects **36**
  select procedures **40**
  select tables/views **37**
  select wizard type **35**
  specify the OTD name **46**
  steps to create **34**
DataDirect JDBC 3.5 Driver **8**
DBCS support **34**
Deployment Profile
  Automap **85**, **105**
document
  scope **9**
Double-Byte Character Set (DBCS) **34**
driver class, JDBC **24**, **25**

E

Editable OTD Support **8**
eWay Connectivity Map **19**, **23**
eWay environment properties **22**
external properties, eWay **25**

I

Importing sample Projects **64**
index
  book **109**
installation **11–16**
Installing
  migration procedures **13**
  Repository on UNIX **11**
  sample Projects and Javadocs **13**
  Sybase **11**

J

Javadocs, installing **13**
JCD operations **51**
  delete(String sWhere) **51**
  deleteRow() **51**
  insert() **51**
  insertRow() **51**

select(String where) **51**
update(String sWhere) **51**
updateRow() **51**
JDBC
driver class **24**, **25**
JDBC connector settings, outbound **26**
JDBC connector settings, outbound XA **29**, **31**

L

LDAP Configuration **8**

M

migration procedures **13**
Multiple Drag-and-Drop **8**

O

operations **49**
OTD, editing existing **48**
outbound eWay properties **23**
outbound non-Transactional properties **24**
outbound XA properties **24**

P

prepared statement
batch operations **59**
executing **59**
Project
importing **64**
properties
ClassName **24**, **25**
ConnectionRetries **28**, **31**, **33**
ConnectionRetryInterval **29**, **31**, **33**
DatabaseName **25**, **26**, **29**, **32**
Delimiter **27**, **30**, **33**
Description **24**, **25**, **26**, **29**, **31**
DriverProperties **27**, **30**, **32**
MaxIdleTime **28**, **31**, **33**
MaxPoolSize **28**, **30**, **33**
MinPoolSize **28**, **30**, **33**
Password **26**, **29**, **32**
PortNumber **25**, **26**, **29**, **32**
ServerName **25**, **26**, **29**, **32**
User **26**, **29**, **32**

R

Relaunchable OTD Support **8**
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 **64**
sample projects, installing **13**
scope **9**
Select Database Objects **36**
Select Procedures **40**
Select Table/Views **37**
Setting Properties
configuring Sybase eWay **18**
connection retry settings **28**, **31**, **33**
eWay Connectivity Map **19**, **23**
eWay environment properties **22**
eWay external **25**
inbound Sybase eWay **25**
JDBC connector settings **26**, **29**, **31**
outbound eWay **23**
outbound non-Transactional properties **24**
outbound non-Transactional Sybase eWay **31**
outbound Sybase eWay **26**
outbound XA properties **24**
outbound XA Sybase eWay **29**
SQL **51**
SQL operations, table **51**
SQL operations, table Delete **54**
SQL operations, table Insert **53**
SQL operations, table Select **52**
SQL operations, table Update **54**
SQL Server eWay Database Wizard **34**
Stored Procedure
executing **55**
manipulating ResultSet and update count **56**
Sybase ASE 15 **8**
Sybase eWay Project
assigning operations in BPEL **62**
assigning operations in JCD **62**
building and deploying (BPEL) **64**
eInsight and eGate components **62**
implementing **60**
Importing **64**
input files **61**
Insert, Update, Delete, Select **61**
output BPEL files **61**
output JCD files **61**
prjSybase_BPEL **60**

prjSybase_JCD **60**
running sample projects **87**, **107**
running the SQL script **63**
Steps to run sample projects **63**

T

Table
SQL operations **51**
SQL operations, Delete **54**
SQL operations, Insert **53**
SQL operations, Select **52**
SQL operations, Update **54**
text conventions **9**

U

update count **56**

V

Version control **8**