SUN SEEBEYOND

eWAY™ ADAPTER FOR INFORMIX USER'S GUIDE

Release 5.1.1

Sun microsystems

Version 20060627112931

Contents

Chapter 1

Introducing the Informix eWay

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

This chapter provides a brief overview of operations, components, general features, and system requirements of the Informix eWay.

What's in This Chapter

1.1 About Informix

Informix Dynamic Server (IDS) database is a multithreaded object-relational database server that manages data that is stored in rows and columns. It employs a single processor or symmetric multiprocessing (SMP) systems and dynamic scalable architecture to deliver database scalability, manageability and performance.

1.2 About the Informix eWay

The Informix eWay is a component that connects eGate and the Informix Dynamic Server (IDS) database. The Informix eWay is designed to handle all the communication details necessary to send and receive data between these components.

In addition to handling communications, the Informix eWay can also apply business logic within Collaboration Rules to perform any of eGate's range of data identification, manipulation, and transformation operations.

1.3 What's New in This Release

The Informix eWay includes the following changes and new features:

- Version Control: An enhanced version control system allows you to effectively manage changes to the eWay components.
- Multiple Drag-and-Drop Component Mapping from the Deployment Editor: The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- Support to read configuration parameters from LDAP at runtime.
- Connection Retry Support: Allows you to specify the number of attempts to reconnect, and the interval between retry attempts, in the event of a connection failure.
- 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 eInsight Business Process.
- Connectivity Map Generator: Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.
- Support for Informix: Allows added support for Informix Version 10.

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

1.4 About This Document

This document includes the following chapters:

- **Chapter 1 "Introducing the Informix eWay"**: Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 "Installing the Informix eWay"**: Describes the system requirements and provides instructions for installing the Informix eWay.
- **Chapter 3 "Setting Properties of the Informix eWay"**: Provides instructions for configuring the eWay to communicate with Informix Dynamic Server (IDS) database.
- **Chapter 4 "Using the Informix eWay Database Wizard"**: Provides instructions for creating Object Type Definitions to be used with the Informix eWay.
- **Chapter 5 "Implementing the Informix eWay Sample Projects"**: Provides instructions for installing and running the sample Projects.

Informix eWay Javadoc

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

1.4.1 Scope

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

1.4.2 Intended Audience

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

1.4.3 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

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

1.5 Related Documents

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

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

1.6 Sun Microsystems, Inc. Web Site

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

http://www.sun.com

1.7 Documentation Feedback

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

CAPS_docsfeedback@sun.com

Chapter 2

Installing the Informix eWay

This chapter explains how to install the Informix eWay.

What's in This Chapter

2.1 Installing the Informix eWay

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

Refer to the readme for the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

The Informix eWay Readme is uploaded with the eWay's documentation file (**InformixeWayDocs.sar**) and can be accessed from the **Documentation** tab of the Sun Java Composite Application Platform Suite Installer. Refer to the Informix eWay Readme for the latest requirements before installing the Informix eWay.

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

2.1.1 Installing the Informix eWay on an eGate supported system

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

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

   To upload the Informix eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, expand the **Documentation** option and select **InformixeWayDocs**.
4. Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.
5. From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.
6. Once your product's installation is finished, continue installing the Sun Java Composite Application Platform Suite as instructed in the *Sun Java Composite Application Platform Suite Installation Guide*.

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

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

1. Complete steps 1 through 4 above.
2. Once your product's installation is complete, open the Enterprise Designer and select **Update Center** from the Tools menu. The **Update Center Wizard** appears.
3. For Step 1 of the wizard, simply click **Next**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**

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

After Installation

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

2.1.2 Extracting the Sample Projects and Javadocs

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

Steps to extract the Javadoc include:

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

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

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

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

Steps to extract the Sample Projects include:

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

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

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

Refer to **"Importing a Sample Project" on page 52** for instructions on importing the sample Project into your repository via the Enterprise Designer.

2.2 ICAN 5.0 Project Migration Procedures

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

Export the Project

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

2 From the Project Explorer, right-click your Project and select **Export** from the shortcut menu. The Export Manager appears.

3 Select the Project that you want to export in the left pane of the Export Manager and move it to the Selected Projects field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.

4 In the same manner, select the Environment that you want to export in the left pane of the Export Manager and move it to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.

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

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

Install Java CAPS 5.1.1

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

2 Start the Java CAPS 5.1.1 Enterprise Designer.

Import the Project

1 From the Java CAPS 5.1.1 Enterprise Designer's Project Explorer tree, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.

2 Browse to and select your exported Project file.

3 Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository when the Project was created that are not installed on the Java CAPS 5.1.1 Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.

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

Deploy the Project

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

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

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

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

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

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

To reconfigure your File eWays, do the following:

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

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

3 Deploy your Projects.

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

2.3 Installing Enterprise Manager eWay Plug-Ins

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

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

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

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

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

To add plug-ins from the Enterprise Manager

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

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

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

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

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

2.3.1 Viewing Alert Codes

You can view 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. If your eWay alert codes are not displayed under **Results**, do the following:
   A. From the **Install New Alert Codes** section, browse to and select the eWay alert properties file for the application plug-in that you added. The alert properties files are located in the **alertcodes** folder of your Sun Java Composite Application Platform Suite installation directory.
   B. Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of the eWay's available alert codes is displayed in Table 2.

Table 2 Alert Codes for the Informix 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

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

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

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

Chapter 3

Setting Properties of the Informix eWay

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

What's in This Chapter

3.1 Creating and Configuring an Informix eWay

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

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

3.2 Configuring the eWay Connectivity Map Properties

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

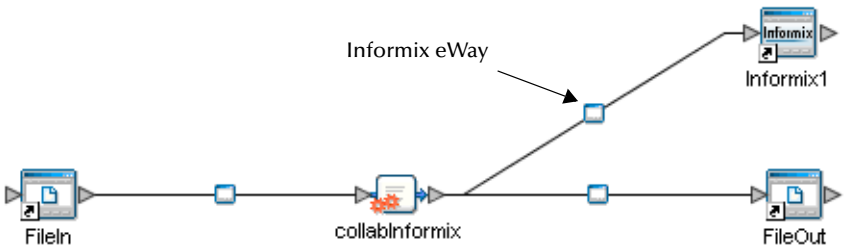
Transaction support levels provided by the Informix eWay include:

- Outbound Informix eWay
- Outbound Informix XA eWay
- Outbound Informix non-Transactional eWay

To configure the eWay properties:

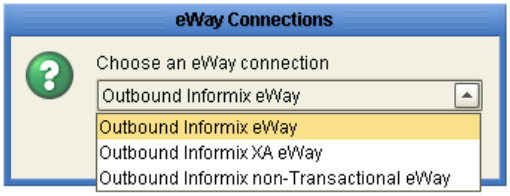
1 On the Enterprise Designer's Connectivity Map (see Figure 1), double-click the outbound Informix eWay icon. The Templates window appears.

Figure 1 Connectivity Map with Components



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

Figure 2 Template window



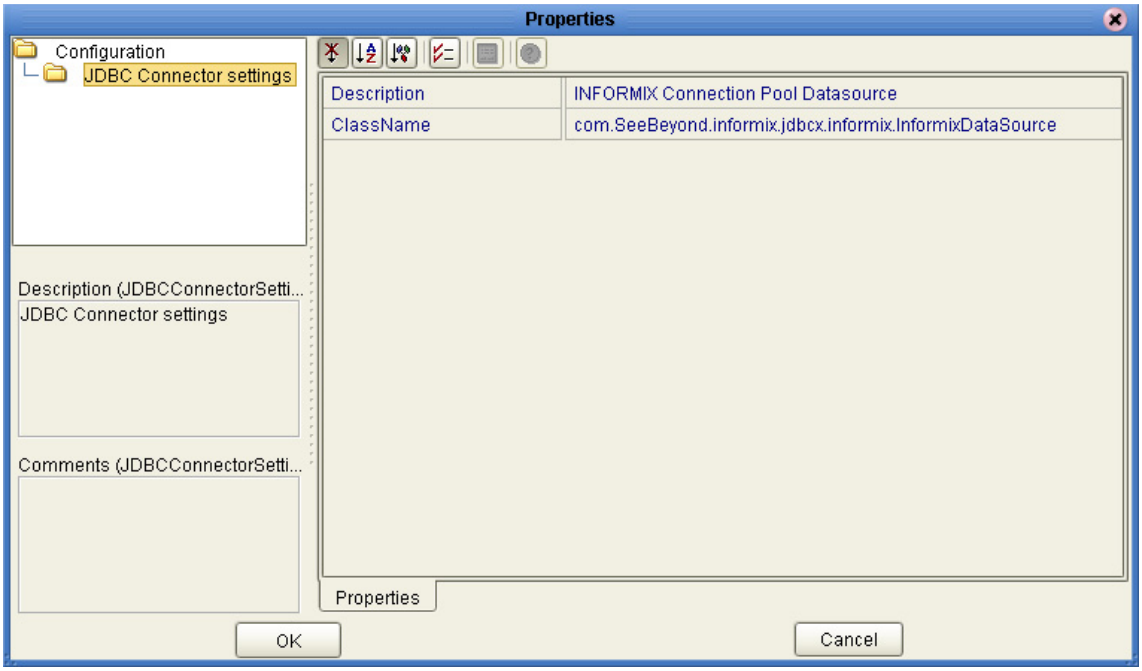
The choices to make are as follows:

- **Outbound Informix 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 Informix XA-eWay:** Also referred to as XATransaction, this support level allows two-phase commit. This means that the transaction, when aborted, will roll back all changes when one of the updates fails. The update could occur in the database eWay or other eWays that support XA. Additionally, the Collaboration can contain only the database eWay, or a combination of database eWay and other eWays that support XA.

- **Outbound Informix 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.

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

Figure 3 Outbound eWay Properties



3.2.1 Transaction Support Levels Between Different Versions

The types of transaction support levels used in Java CAPS 5.1.0 may be different from the support levels used in Java CAPS 5.1.1. Projects that are imported from a Java CAPS 5.1.0 version can potentially display different results, depending on whether the 5.1.0 Java Collaboration Definition (JCD) included multiple (insert/update/delete) operations. This only affects non-XA transactions. If you are using an XA transaction, then you can skip this section.

Example:

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

In 5.1.1, five new records are to be inserted into a table. If one of the records fails to insert (such as when a duplicate key exists), the other four records will not be inserted. This is the behavior of the LocalTransaction.

In order to achieve the same result as in 5.1.0 versions, you can choose the method below:

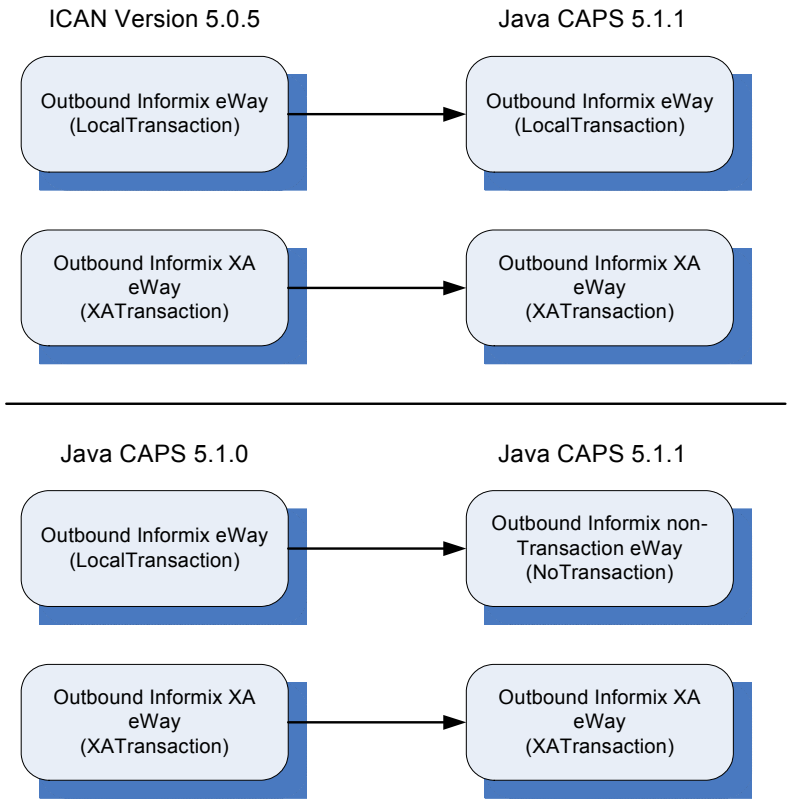
A In the Connectivity Map, delete the link to the database external application, then reconnect the link and select NoTransaction.

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

C Rebuild the Project.

The following charts identifies what transaction support levels changed between 5.0.5 and 5.1.1, and 5.1.0 and 5.1.1, respectively. **Note that there are no changes when migrating from ICAN version 5.0.5 and Java CAPS 5.1.1.**

Figure 4 Transaction Support Levels



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

3.3 Configuring the eWay Environment Properties

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

Available External System properties include:

- Inbound Informix eWay
- Outbound Informix eWay
- Outbound Informix XA eWay
- Outbound Informix non-Transactional eWay

To Configure the Environment Properties:

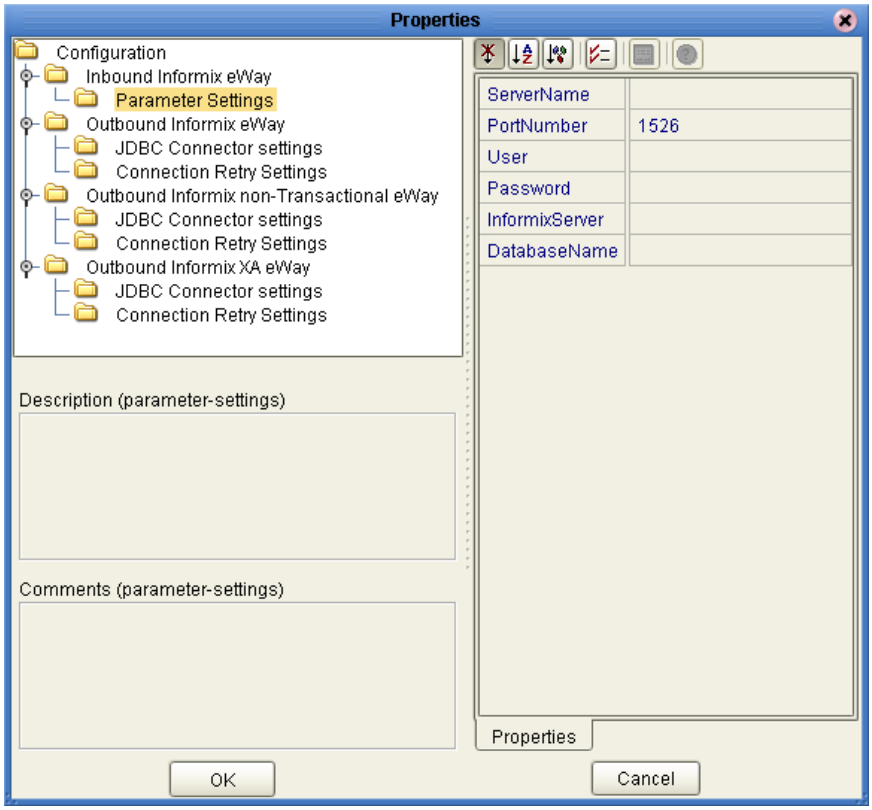
1 In Enterprise Explorer, click the Environment Explorer tab.

2 Expand the Environment created for the Informix Project and locate the Informix External System.

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

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

Figure 5 Informix 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.

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

3.4 eWay Connectivity Map Properties

The eWay Connectivity Map consists of the following properties categories:

- **Outbound eWay**
  - **Outbound Informix eWay Connectivity Map Properties** on page 24
- **Outbound XA eWay**
  - **Outbound Informix XA e Way Connectivity Map Properties** on page 24
- **Outbound non-Transactional eWay**
  - **Outbound Informix non-Transactional eWay Connectivity Map Properties** on page 25

3.4.1 Configuring the Outbound eWay Properties

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

Table 3 Outbound Informix eWay Connectivity Map Properties

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is INFORMIX Connection Pool Datasource.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.	A valid class name. The default is **com.SeeBeyond.informix.jdbcx.informix.InformixDataSource**.

3.4.2 Configuring the Outbound XA eWay Properties

The Outbound XA eWay Properties include inbound parameters used by the external database. Informix supports XA, a data source that provides connections that can participate in a distributed transaction. XA is a two-phase commit protocol that forms part of the JDBC 2.0 Standard Extension.

Table 4 Outbound Informix XA e Way Connectivity Map Properties

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is INFORMIX XA Datasource.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the XADataSource interface.	A valid class name. The default is **com.SeeBeyond.informix.jdbcx.informix.InformixDataSource**.

3.4.3 Configuring the Outbound non-Transactional eWay Properties

You can create Informix databases with or without logging enabled. If logging is disabled, then Non-Transactional mode must be used. Because data logs are not retained during Non-Transactional execution of SQL calls, data recovery is not possible during accidental or unscheduled shut-down of the database server.

Disabled logging also prevents transactions—enclosed in BEGIN-Tran and END-Tran statements—from occurring. This means that Non-Transactional mode cannot be used in XA (two-phase commit) transactions.

The Outbound non-Transactional eWay Properties listed in Table 5 include inbound parameters used by the external database.

Table 5 Outbound Informix non-Transactional eWay Connectivity Map Properties

Name	Description	Required Value
Description	Enter a description for the database.	A valid string. The default is INFORMIX non-Transactional Connection Pool Datasource.
ClassName	Specifies the Java class in the JDBC driver that is used to implement the non-Transactional ConnectionPoolDataSource interface.	A valid class name. The default is **com.SeeBeyond.informix.jdbcx.informix.InformixDataSource**.

3.5 eWay External System Properties

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

- **Inbound eWay**
  - **Inbound eWay External System Parameter Settings** on page 26
- **Outbound eWay**
  - **Outbound eWay External System JDBC Connector Settings** on page 26
  - **Outbound eWay External System Connection Retry Settings** on page 28
- **Outbound non-Transactional eWay**
  - **Outbound non-Transactional eWay External System JDBC Connector Settings** on page 28
  - **Outbound non-Transactional eWay External System Connection Retry Settings** on page 30
- **Outbound XA eWay**

- **Outbound XA eWay External System JDBC Connector Settings** on page 30
- **Outbound XA eWay External System Connection Retry Settings** on page 32

3.5.1 Inbound eWay External System Properties

Before deploying your eWay, you will need to set the Environment properties. This section describes the External System properties used by the Inbound Informix eWay.

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

Table 6 Inbound eWay External System Parameter 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.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1526**.
InformixServer	Specifies the name of the Informix server being used.	Any valid string.
DatabaseName	Specifies the name of the database being used by the server.	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.

3.5.2 Outbound eWay External System Properties

Before deploying your eWay, you will need to set the Environment properties. This section describes the External System properties used by the Outbound Informix eWay.

Table 7 Outbound eWay External System JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database. The default is INFORMIX Connection Pool Datasource.	Any valid string.
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1526**.
InformixServer	Specifies the name of the Informix server being used.	Any valid string.

Table 7 Outbound eWay External System JDBC Connector Settings (Continued)

Name	Description	Required Value
DatabaseName	Specifies the name of the database being used by the server.	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	Specifies the driver properties used for this eWay. If you choose 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 properties to assure a connection. The additional methods will need to be identified in the Driver Properties.	Any valid delimiter. 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 setURL, give the method a String for the URL **"setURL#<url>##"**. ***Note:*** The setSpyAttributes, 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. **"setURL#jdbc:SeeBeyond:informix://<host>1526;DatabaseName=<database>##setInformixServer#<informixserver>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##"**.
Delimiter	Specifies the delimiter character used in the DriverProperties prompt.	The default is **#**.

Table 7 Outbound eWay External System JDBC Connector Settings (Continued)

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.	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 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**.

Table 8 Outbound eWay External System Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection upon initial failure.	A valid numeric value. The default is **0**.
ConnectionRetryInterval	Specifies the number of milliseconds of pause before each reattempt to acces the destination file. This property is used in conjunction with the Connection Retries setting.	A valid numeric value. The default is **1000**.

3.5.3 Outbound non-Transactional eWay External System Properties

Before deploying your eWay, you will need to set the Environment properties. This section describes the External System properties used by the Outbound Informix non-Transactional eWay.

Table 9 Outbound non-Transactional eWay External System JDBC Connector Settings

Name	Description	Required Values
Description	Enter a description for the database. The default is INFORMIX non-Transactional Connection Pool Datasource.	Any valid string.
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1526**.

Table 9 Outbound non-Transactional eWay External System JDBC Connector Settings

Name	Description	Required Values
InformixServer	Specifies the name of the Informix server being used.	Any valid string.
DatabaseName	Specifies the name of the database being used by the server.	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	Specifies the driver properties used for this eWay. If you choose 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 properties to assure a connection. The additional methods will need to be identified in the Driver Properties.	Any valid delimiter. 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 setURL, give the method a String for the URL **"setURL#<url>##"**. ***Note:*** The setSpyAttributes, 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. **"setURL#jdbc:SeeBeyond:informix://<host>1526;DatabaseName=<database>##setInformixServer#<informixserver>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##"**.
Delimiter	Specifies the delimiter character used in the DriverProperties prompt.	The default is **#**.

Table 9 Outbound non-Transactional eWay External System JDBC Connector Settings

Name	Description	Required Values
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 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**.

Table 10 Outbound non-Transactional eWay External System Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection upon initial failure.	A valid numeric value. The default is **0**.
ConnectionRetryInterval	Specifies the number of milliseconds of pause before each reattempt to acces the destination file. This property is used in conjunction with the Connection Retries setting.	A valid numeric value. The default is **1000**.

3.5.4 Outbound XA eWay External System Properties

Before deploying your eWay, you will need to set the Environment properties. This section describes the External System properties used by the Outbound Informix XA eWay.

Table 11 Outbound XA eWay External System JDBC Connector Settings

Name	Description	Required Value
Description	Enter a description for the database. The default is INFORMIX XA Connection Pool Datasource.	Any valid string.
ServerName	Specifies the host name of the external database server.	Any valid string.
PortNumber	Specifies the I/O port number on which the server is listening for connection requests.	A valid port number. The default is **1526**.

Table 11 Outbound XA eWay External System JDBC Connector Settings (Continued)

Name	Description	Required Value
InformixServer	Specifies the name of the Informix server being used.	Any valid string.
DatabaseName	Specifies the name of the database being used by the server.	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	Specifies the driver properties used for this eWay. If you choose 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 properties to assure a connection. The additional methods will need to be identified in the Driver Properties.	Any valid delimiter. 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 setURL, give the method a String for the URL **"setURL#<url>##"**. ***Note:*** The setSpyAttributes, 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. **"setURL#jdbc:SeeBeyond:informix://<host>1526;DatabaseName=<database>##setInformixServer#<informixserver>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##"**.
Delimiter	Specifies the delimiter character used in the DriverProperties prompt.	The default is **#**.

Table 11 Outbound XA eWay External System JDBC Connector Settings (Continued)

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.	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 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**.

Table 12 Outbound XA eWay External System Connection Retry Settings

Name	Description	Required Value
ConnectionRetries	Specifies the number of retries to establish a connection upon initial failure.	A valid numeric value. The default is **0**.
ConnectionRetryInterval	Specifies the number of milliseconds of pause before each reattempt to acces the destination file. This property is used in conjunction with the Connection Retries setting.	A valid numeric value. The default is **1000**.

Chapter 4

Using the Informix eWay Database Wizard

This chapter describes how to build and use Object Type Definitions (OTDs) using the Informix eWay Database Wizard.

What's in This Chapter

4.1 About the Database OTD Wizard

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

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

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

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

4.2 Creating a New Informix OTD

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

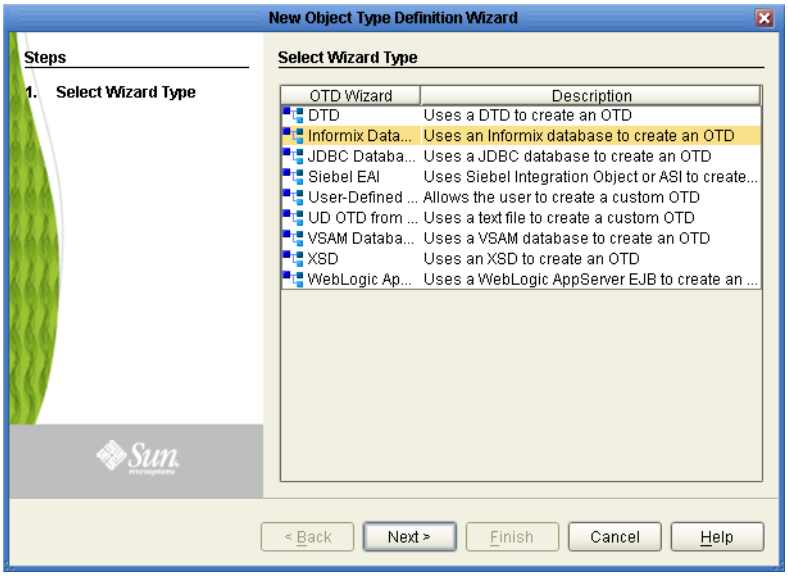
4.2.1 Select Wizard Type

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

Steps Required to Select the Informix Database OTD Wizard Include:

On the Project Explorer tree, right click the Project and select **New > Object Type Definition** from the shortcut menu. The **Select Wizard Type** page appears, displaying the available **OTD** wizards. See Figure 6.

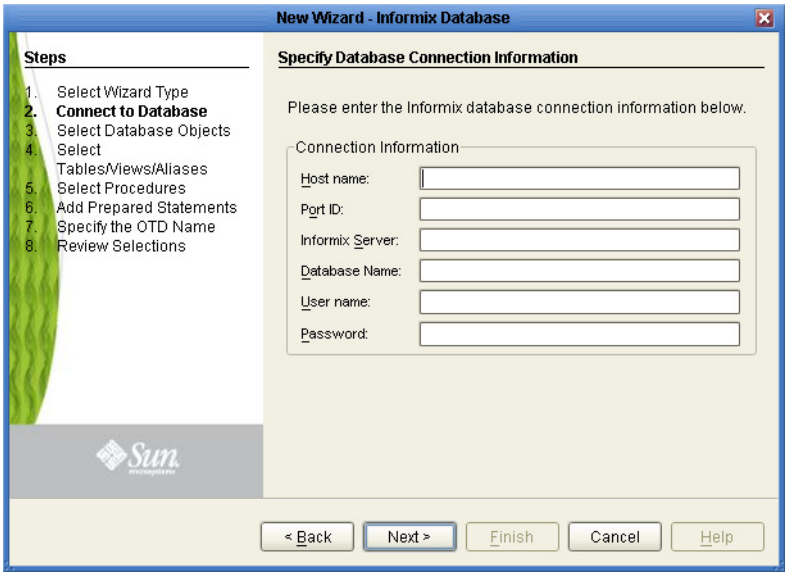
Figure 6 OTD Wizard Selection



4.2.2 Connect To Database

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

Figure 7 Database Connection Information



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

Required Database Connection Fields include:

- **Host name** – the name of the host to which you are connecting.
- **Port ID** – the host port number (1526 is the default).
- **Informix Server** – the name of the Informix server.
- **Database name** – the name of the database to which you are connecting.
- **User name** – a valid Informix database username.
- **Password** – a password for the user name noted above.

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

4.2.3 Select Database Objects

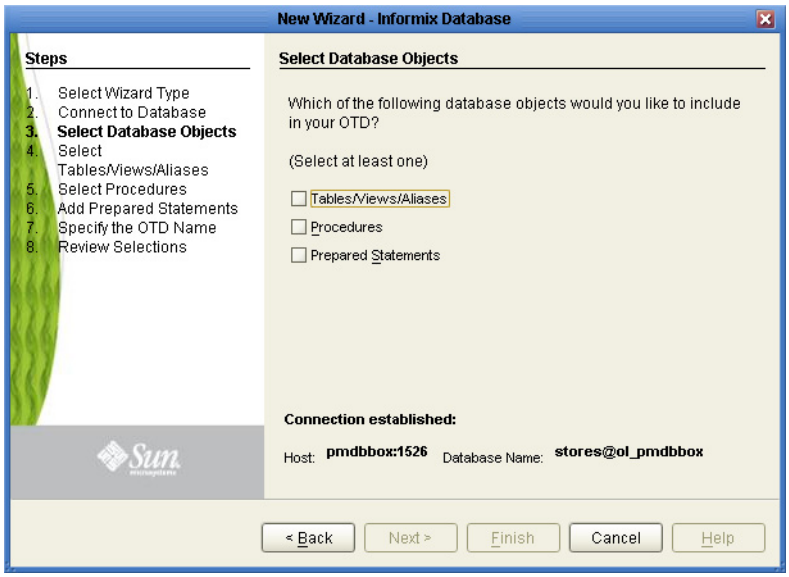
Select the type of Informix database objects you want included in the OTD.

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



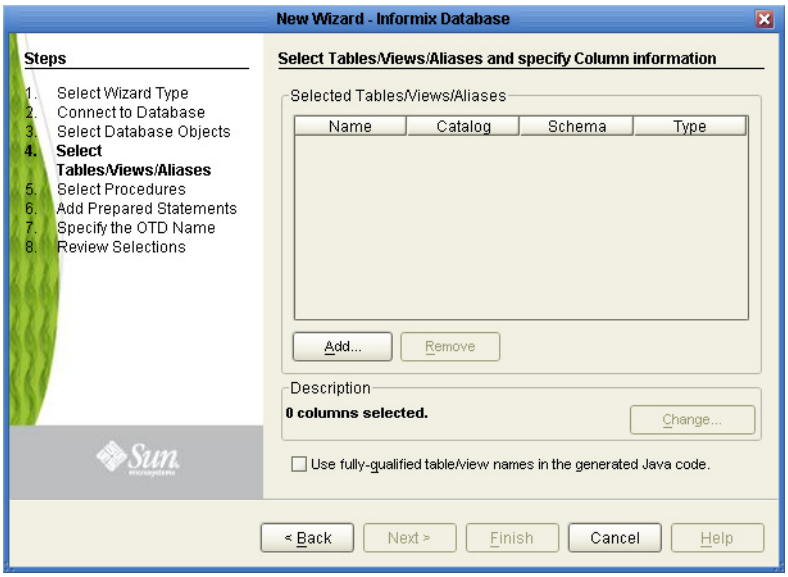
4.2.4 Select Tables/Views/Aliases

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

Steps Required to Select Table/Views/Aliases Include:

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

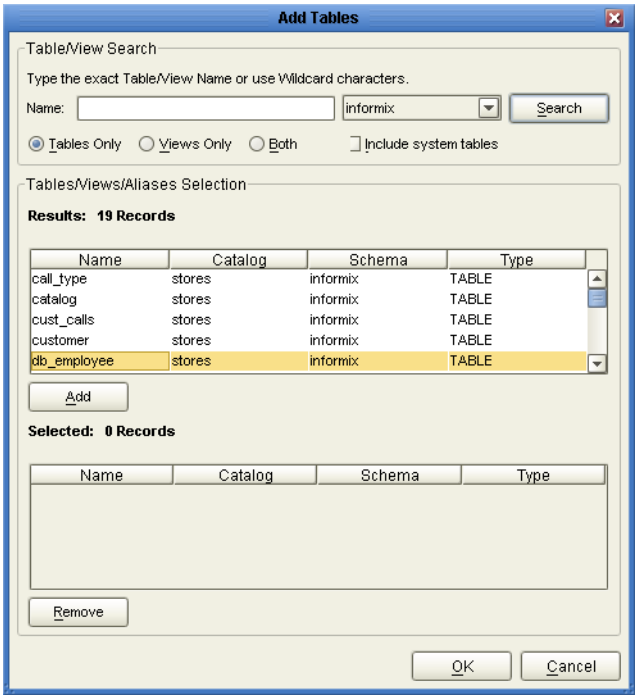
Figure 9 Select Tables/Views



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.

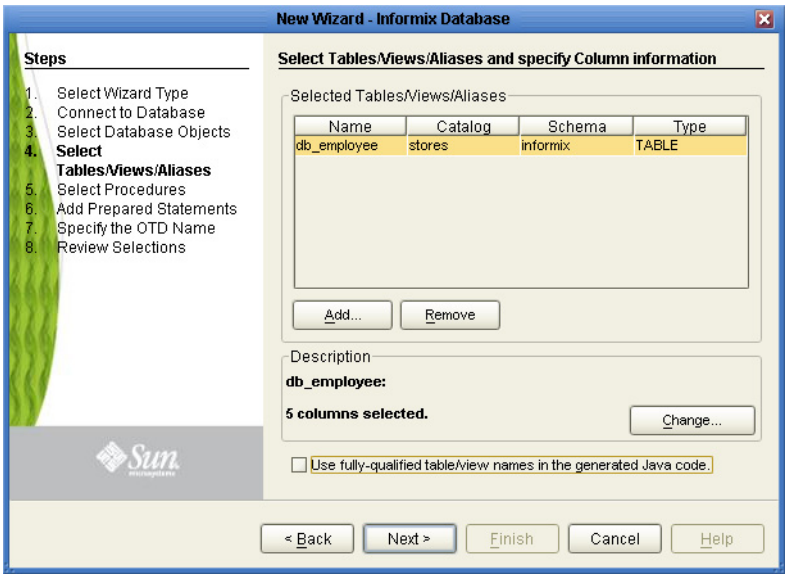
Figure 10 Database Wizard - All Schemes



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

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

Figure 11 Selected table and column window



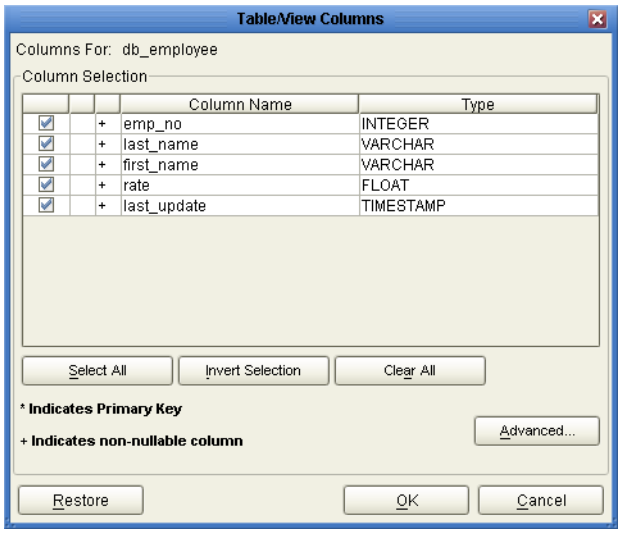
5 In the **Selected Tables/Views/Aliases** section, review the table(s) you have selected. To make changes to the selected Table or View, click Change. If you do not wish to make any additional changes, click **Next** to continue.

6 In the **Table/View Columns** window, you can select or deselect your table columns. You can also change the data type for each table by highlighting the data type and

selecting a different one from the drop down list. If you would like to change any of the tables columns, click **Change**. See Figure 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 window



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.

Important: *The Informix database driver currently deployed with the Informix eWay displays non-nullable columns in the OTD wizard dialogue box regardless of whether the columns in the database accept null values or not.*

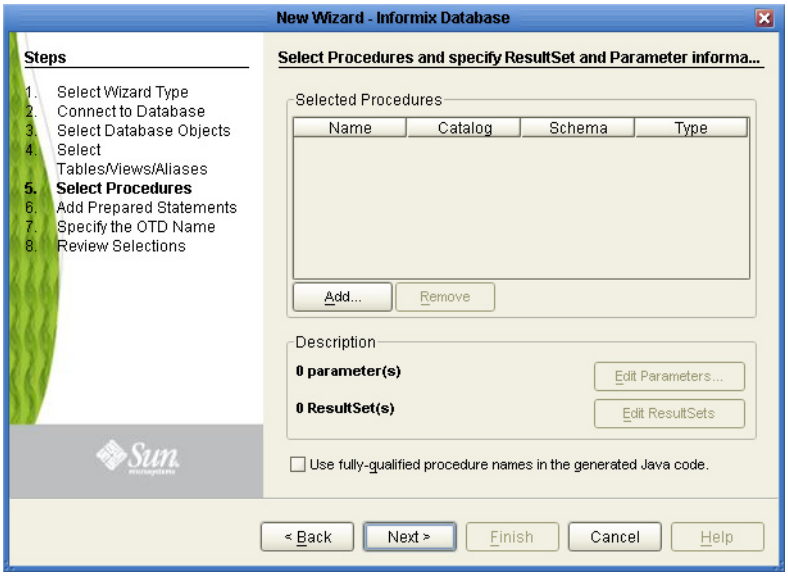
4.2.5 Select Procedures

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 13 Select Procedures window

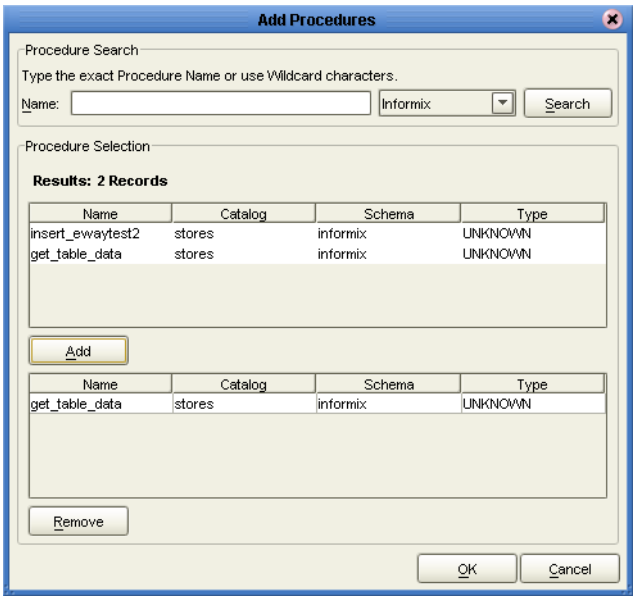


2 On the **Select Procedures** window, enter the name of a Procedure or select a table from the drop down list. Click **Search**. Wildcard characters can also be used.

Note: *You must use lower case schema names when calling stored procedures.*

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

Figure 14 Add Procedures

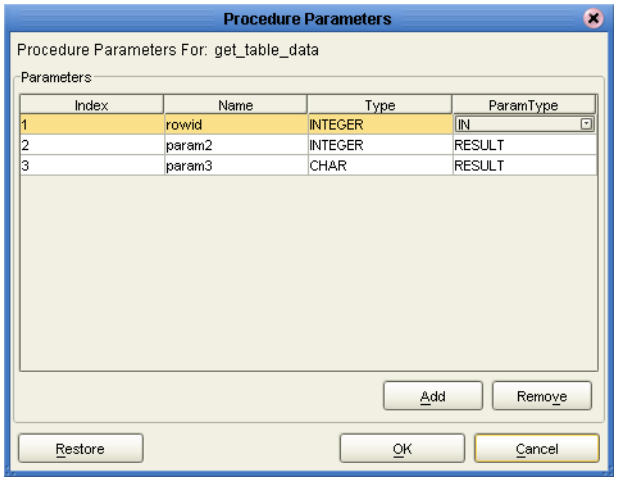


Important: *The Informix database driver currently deployed with the Informix eWay does not return metadata to provide fully the type of stored procedure available when a search*

*is executed. Thus the **Type** field for all procedures will be populated with "Unknown."*

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

Figure 15 Procedure Parameters

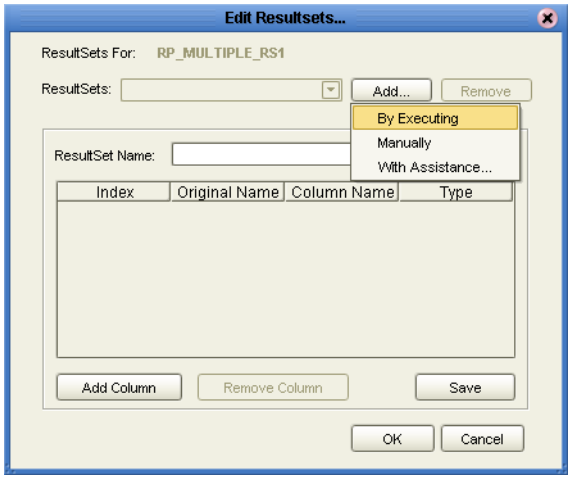


5 To restore the data type, click **Restore**. When finished, click **OK**.

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 16 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 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 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 is the most flexible way to generate the result set nodes. It allows users to specify the node name, original column name and data type manually. One drawback of this method is that users need to know the original column names and data types. This is not always possible. For example, the column name of 3*C in this query.

```
SELECT A, B, 3*C FROM table T
```

is generated by the database. In this case, "**With Assistance**" mode is a better choice.

If you modify the ResultSet generated by the "**Execute**" mode of the Database Wizard you need to make sure the indexes match the Stored Procedure. This assures your ResultSet indexes are preserved.

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

4.2.6 Add Prepared Statements

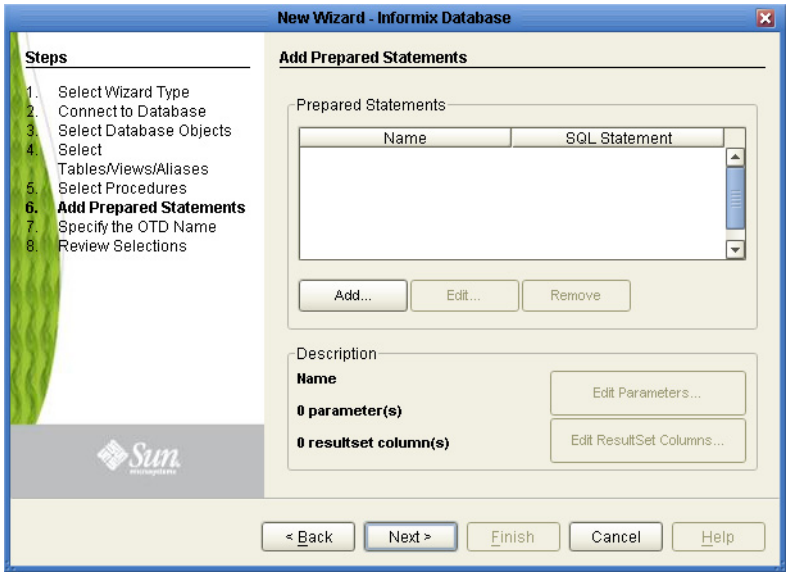
Add a Prepared Statement object to your OTD.

Steps Required to Add Prepared Statements Include:

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

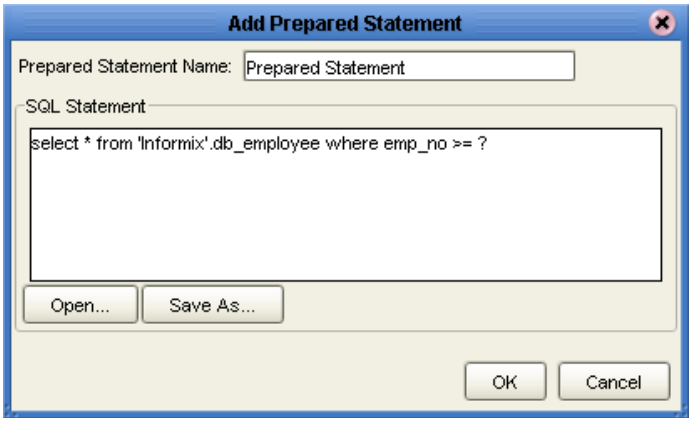
1 On the **Add Prepared Statements** window, click **Add**. The **Add Prepared Statement** window appears.

Figure 17 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 18.

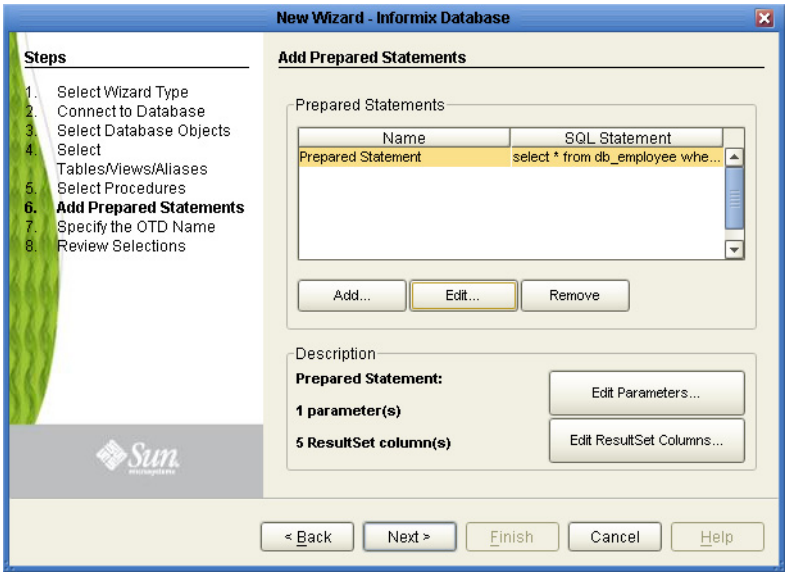
Figure 18 Prepared SQL Statement



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.

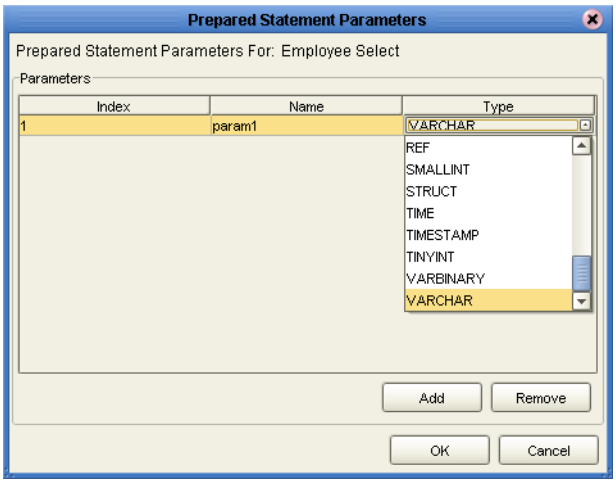
4 Click **Add** if you want to add additional parameters to the Statement or highlight a row and click **Remove** to remove it. Click **OK**. Figure 19.

Figure 19 Add Prepared Statement window



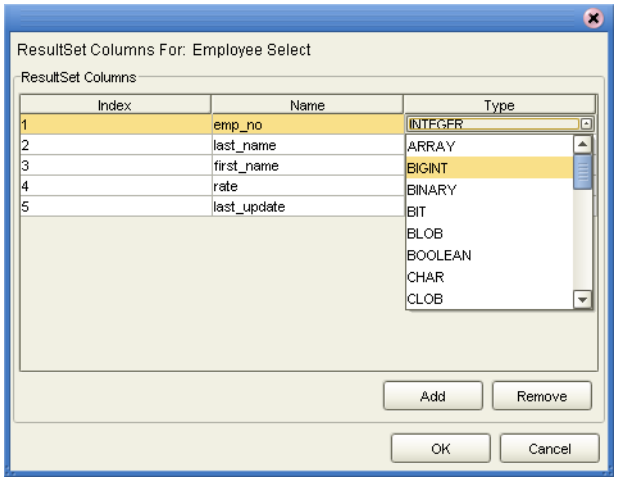
5 To edit the parameters, click **Edit Parameters**. You can change the datatype by clicking in the **Type** field and selecting a different type from the list.

Figure 20 Edit the Prepared Statement Parameters



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

Figure 21 ResultSet Columns



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

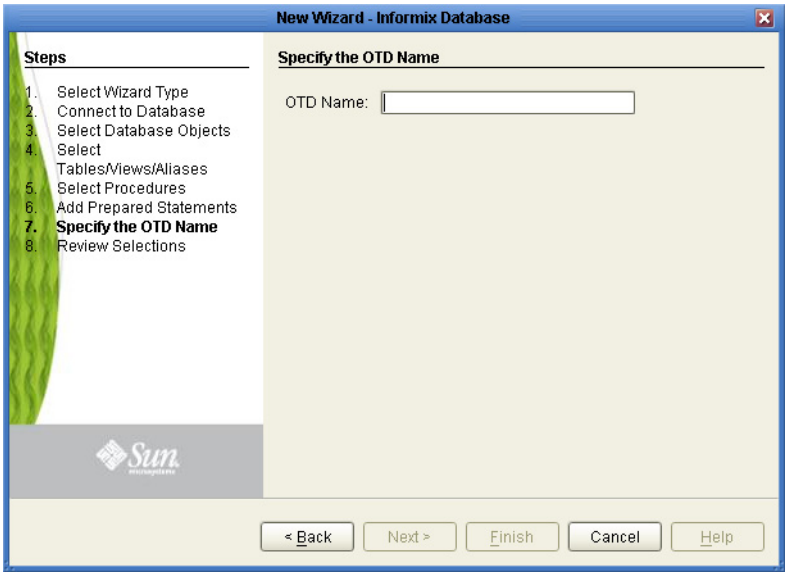
4.2.7 Specify the OTD Name

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

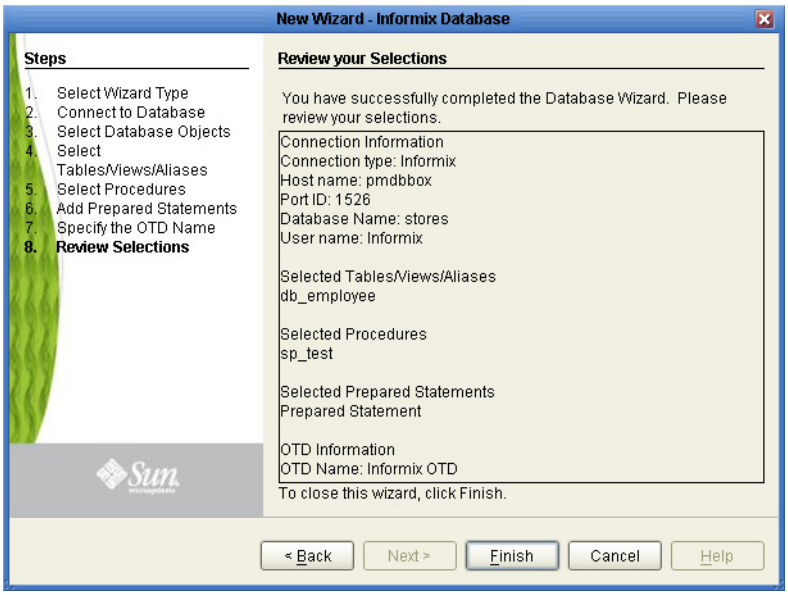
Review the selections made for the new OTD.

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 Informix 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 Informix Database Connection Information wizard appears.

4 Enter the connection information as described in **"Connect To Database" on page 34**, 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

Implementing the Informix eWay Sample Projects

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

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

What's in This Chapter

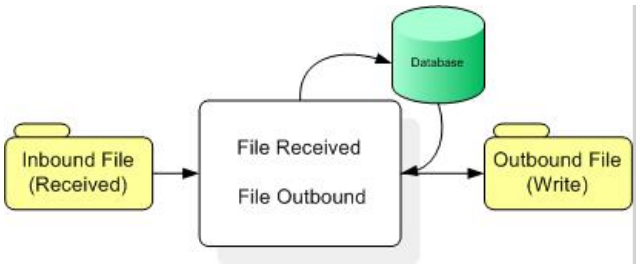
5.1 About the Informix eWay Sample Projects

The Informix eWay **Informix_eWay_Sample.zip** file contains two sample Projects that provide basic instruction on using Informix operations in the Java Collaboration Definition (JCD), or the eInsight Business Process Projects.

The **prjInformix_JCD** sample Project uses input files to pass data into Collaborations. There are four Collaborations that demonstrate the Insert, Update, Delete, and Table Select operations, and one Collaboration to demonstrate a Prepared Statement. Results are written out to an output file.

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

Figure 24 Database project flow



Both the **prjInformix_JCD** and **prjInformix_BPEL** sample Projects demonstrate how to:

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

In addition to the sample Projects, the **Informix_eWay_Sample.zip** file also includes six sample input trigger files and ten sample output files, as follows:

Sample input files

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

Sample output JCD files

- JCD_Insert_output().dat
- JCD_Delete_output().dat
- JCD_Update_output().dat
- JCD_PsSelect_output().dat
- JCD_TableSelect_output().dat

Sample output BPEL files

- BPEL_Insert_output().dat
- BPEL_Delete_output().dat
- BPEL_Update_output().dat
- BPEL_TableSelect_output().dat
- BPEL_PsSelect_output().dat

5.1.1 Sample Project Data

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

Table 13 Sample Project Data - db_employee Table

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

Note: *Informix databases do not accept table data with columns that include Boolean data types. For example, an Informix server expects a literal "T" or "F" for True or False boolean values, not Java primitive boolean True or False values.*

5.1.2 Operations Used in the Informix Sample Projects

The following database operations are used in both the JCD and eInsight Business Process sample projects:

- Insert
- Update
- Delete
- Select

Assigning Operations in JCD

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

1. Create a Collaboration that contains an OTD using Informix.
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 eInsight Business Processes

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

1. Select the desired **receive** or **write** operation under the eWay in the Enterprise Explorer.
2. Drag the operation onto the eInsight Business Process canvas.

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

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

Note: *Inbound database eWays are only supported within eInsight Business Processes Collaborations.*

5.1.3 About the eInsight Engine and eGate Components

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

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

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

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

5.2 Running the Sample Projects

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

1. Run the SQL script.

   The script creates the tables and records required by the sample Project.

2. Import the sample Projects.
3. Build, deploy, and run the sample Projects.

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

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

4. Check the output.

5.3 Running the SQL Script

The data used for both the **JCD** and **eInsight Business Processes** sample Projects are contained within a table called **db_employee**. You create this table by using the SQL statement **Informix_sample_script.sql**, that is included in the sample Project. Note that you must use a database tool to run the script.

Following is the SQL statement designed for the sample Projects.

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

The sample Projects provided with the Informix eWay use input files to pass predefined data or conditions into the Collaboration or eInsight Business Process, which then transforms the database contents, and delivers the ResultSet.

5.4 Importing a Sample Project

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

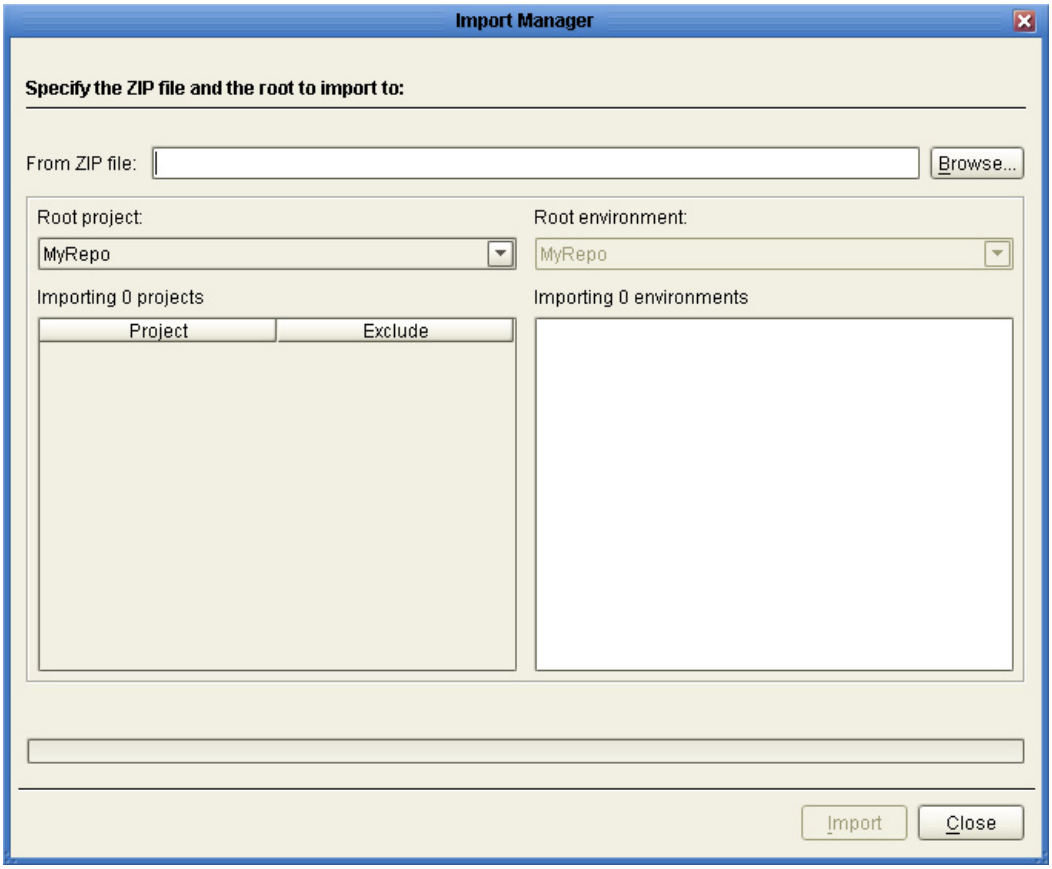
1 Extract the samples from the Sun Java Composite Application Platform Suite Installer to a local file.

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

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

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

Figure 25 Import Manager Dialog Box



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

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

5.5 Building and Deploying the prjInformix_JCD Sample Project

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

Steps required to create the sample project

5.5.1 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, **prjInformix_JCD**).

5.5.2 Creating the OTDs

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

- Informix Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a Informix **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 **Informix Database OTD Wizard** from the list of OTD Wizards and click **Next**.
3. Enter the connection information for the Informix database. Connection fields include:
   - Host Name
   - Port ID
   - Informix Server
   - Database Name
   - 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 Informix database. The **Add Tables** window appears.
6. Search for or Type in the name of the database. In this example we use the **DB_EMPLOYEE** table. Click **Select** when the database appears in the Results selection frame. Click **OK** to close the Add Tables window
7. Click **Next** the Add Prepared Statements Wizard appears.
8. Click **Add**, the Add Prepared Statement window appears. Enter the following:
   - Prepared Statement Name: Select_ps
   - SQL Statement:

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

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

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

Steps required to create inbound and outbound DTD OTDs include:

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

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

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

5.5.3 Creating a Connectivity Map

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

Steps required to create a new Connectivity Map:

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

- File External Application (2)

- Informix External Application
- Service

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

Steps required to select a Informix 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, **Informix** 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
   - Informix1 to esInformixOUT
4. Rename each Connectivity Map Service to match the intended operation, as for example:
   - jcdDelete
   - jcdInsert
   - jcdPsSelect
   - jcdTableSelect
   - jcdUpdate

5.5.4 Creating the Collaboration Definitions (Java)

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

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

jcdInsert Collaboration

Steps required to create the jcdInsert Collaboration:

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

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

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

jcdPsSelect Collaboration

Steps required to create the jcdPsSelect Collaboration:

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

   Note that the otdOutputDTD_DBemployee OTD is created from the otdOutputDTD.dtd that is included in the Sample Project.
6. Click the **Up One Level** button twice to return to the Repository. Double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **Selected OTDs** field now lists the **FileClient** OTD.
7. Click **Finish**. The Collaboration Editor with the new **jcdPsSelect** Collaboration appears in the right pane of the Enterprise Designer.

jcdTableSelect Collaboration

Steps required to create the jcdTableSelect Collaboration:

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

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

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

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

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 **prjInformix_JCD > otdALL > otdInformix**. The **otdInformix** OTD is added to the Selected OTDs field.

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

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

5.5.5 Create the Collaboration Business Rules

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

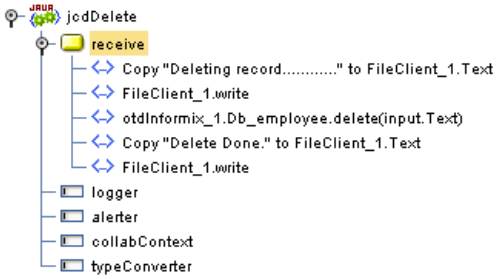
Creating the jcdDelete Business Rules

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

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

The **jcdDelete** Collaboration contains the Business Rules displayed in Figure 26.

Figure 26 jcdDelete Business Rules

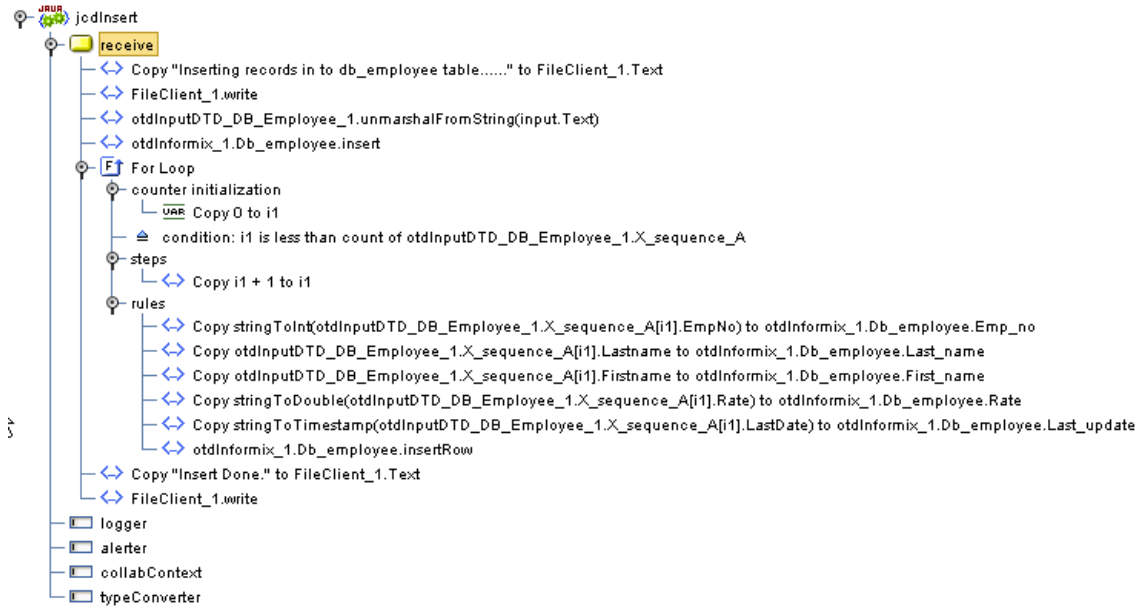


Creating the jcdInsert Business Rules

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 **otdInformix** 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 27.

Figure 27 jcdInsert Business Rules



Sample code from the jcdInsert Includes:

```
package prjInformix_JCDjcdALL;
public class jcdInsert
{

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

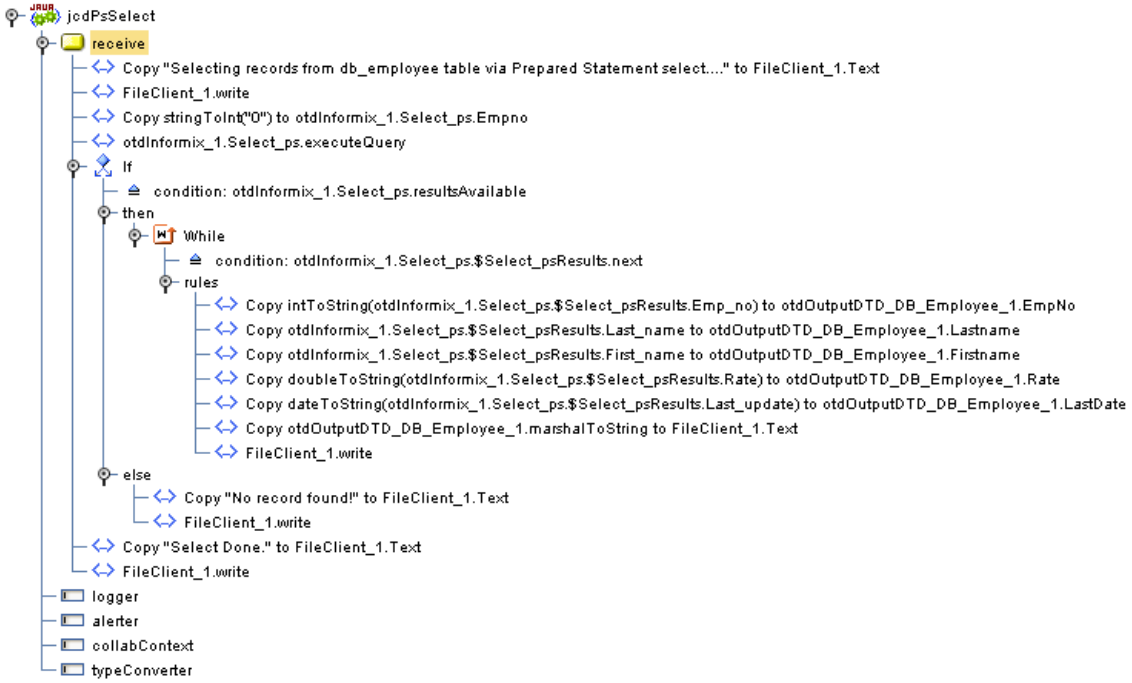
```
public void receive( com.stc.connector.appconn.file.FileTextMessage
input, com.stc.connector.appconn.file.FileApplication FileClient_1,
dtd.otdInputDTD1598082045.DB_Employee otdInputDTD_DB_Employee_1,
otdInformix.OtdInformixOTD otdInformix_1 )
        throws Throwable
    {

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

        otdInformix_1.getDb_employee().insert();
        for (int i1 = 0; i1 <
otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdInformix_1.getDb_employee().setEmp_no(
typeConverter.stringToInt( otdInputDTD_DB_Employee_1.getX_sequence_A(
i1 ).getEmpNo(), "#", false, 0 ) );
            otdInformix_1.getDb_employee().setLast_name(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdInformix_1.getDb_employee().setFirst_name(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdInformix_1.getDb_employee().setRate(
typeConverter.stringToDouble(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getRate(),
"#.000000;-#.000000", false, 0 ) );
            otdInformix_1.getDb_employee().setLast_update(
typeConverter.stringToTimestamp(
otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(),
"yyyy-MM-dd hh:mm:ss", false, "" ) );
            otdInformix_1.getDb_employee().insertRow();

\\ Writes a message to confirm the 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 **otdOutputDTD_DB_Employee** 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 28.

Figure 28 jcdPsSelect Business Rules



Sample code from the jcdPsSelect Includes:

```
package prjInformix_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, com.stc.connector.appconn.file.FileApplication FileClient_1,
dtd.otdOutputDTD_896924227.DB_Employee otdOutputDTD_DB_Employee_1,
otdInformix.OtdInformixOTD otdInformix_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 otdOutputDTD_DB_Employee
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();
```

```
        otdInformix_1.getSelect_ps().setEmpno(
typeConverter.stringToInt( "0", "#", false, 0 ) );
        otdInformix_1.getSelect_ps().executeQuery();
        if (otdInformix_1.getSelect_ps().resultsAvailable()) {
            while
(otdInformix_1.getSelect_ps().get$Select_psResults().next()) {
                otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.intToString(
otdInformix_1.getSelect_ps().get$Select_psResults().getEmp_no(), "#",
false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastname(
otdInformix_1.getSelect_ps().get$Select_psResults().getLast_name() );
                otdOutputDTD_DB_Employee_1.setFirstname(
otdInformix_1.getSelect_ps().get$Select_psResults().getFirst_name()
);
                otdOutputDTD_DB_Employee_1.setRate(
typeConverter.doubleToString(
otdInformix_1.getSelect_ps().get$Select_psResults().getRate(),
"#.000000;-#.000000", false, "" ) );
                otdOutputDTD_DB_Employee_1.setLastDate(
typeConverter.dateToString(
otdInformix_1.getSelect_ps().get$Select_psResults().getLast_update(),
"yyyy-MM-dd hh:mm:ss", false, "" ) );
                FileClient_1.setText(
otdOutputDTD_DB_Employee_1.marshalToString() );
                FileClient_1.write();
            }
        } else {
            FileClient_1.setText( "No record found!" );
            FileClient_1.write();
        }

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

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

}
```

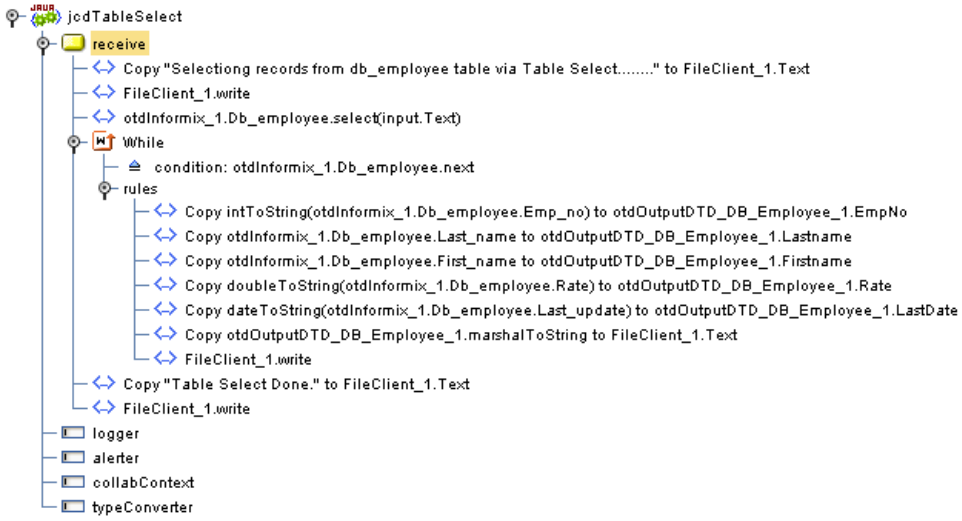
Creating the jcdTableSelect Business Rules

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

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

The **jcdTableSelect** Collaboration contains the Business Rules displayed in Figure 29.

Figure 29 jcdTableSelect Business Rulest



Sample code from the jcdTableSelect Includes:

```
package prjInformix_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, com.stc.connector.appconn.file.FileApplication FileClient_1,
dtd.otdOutputDTD_896924227.DB_Employee otdOutputDTD_DB_Employee_1,
otdInformix.OtdInformixOTD otdInformix_1 )
        throws Throwable
    {

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

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

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

        otdInformix_1.getDb_employee().select( input.getText() );
        while (otdInformix_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(
typeConverter.intToString(
otdInformix_1.getDb_employee().getEmp_no(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname(
otdInformix_1.getDb_employee().getLast_name() );
            otdOutputDTD_DB_Employee_1.setFirstname(
otdInformix_1.getDb_employee().getFirst_name() );
            otdOutputDTD_DB_Employee_1.setRate(
typeConverter.doubleToString(
```

```
otdInformix_1.getDb_employee().getRate(), "#.000000;-#.000000",
false, "" ) );
otdOutputDTD_DB_Employee_1.setLastDate( typeConverter.dateToString(
otdInformix_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();
    }

}
```

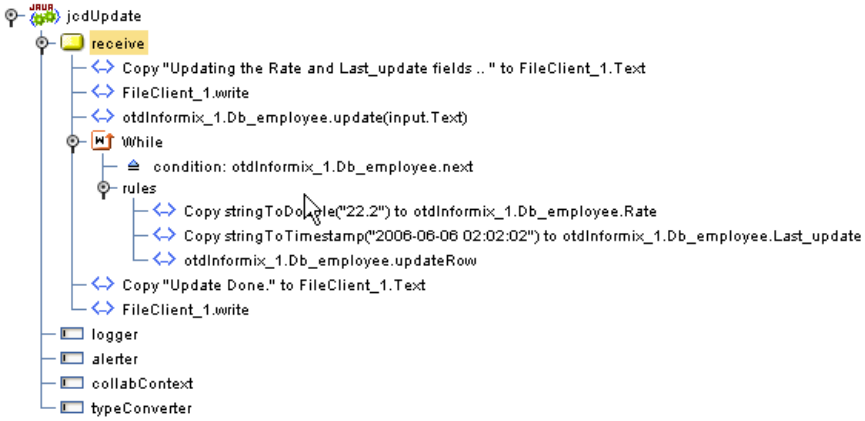
Creating the jcdUpdate Business Rules

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

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

The **jcdUpdate** Collaboration contains the Business Rules displayed in Figure 30.

Figure 30 jcdTableUpdate



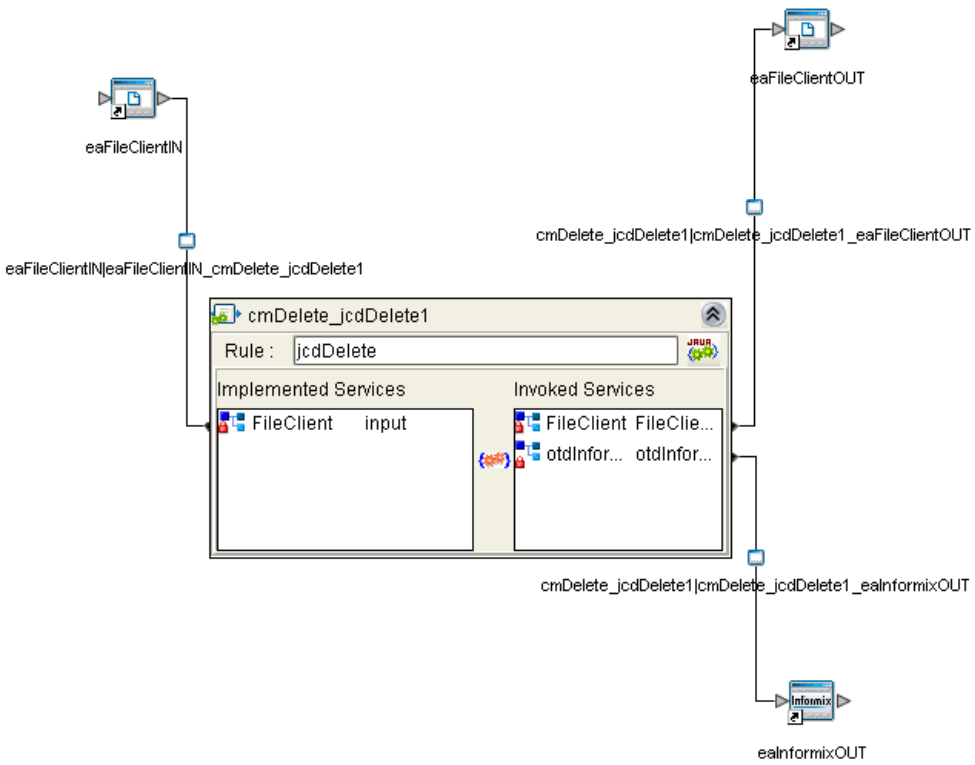
5.5.6 Binding the eWay Components

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

Steps required to bind eWay components together:

1. Double-click a Connectivity Map—in this example **cmDelete**—in the Project Explorer tree. The **cmDelete** Connectivity Map appears in the Enterprise Designers canvas.
2. Drag and drop the **jcdDelete** Collaboration from the Project Explorer to the **jcdDelete** Service. The Service icon "gears" change from red to green.
3. Double-click the **jcdDelete** Service. The **jcdDelete** Binding dialog box appears.
4. Map the input **FileClient** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **jcdDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **jcdDelete**.
5. From the **jcdDelete** Binding dialog box, map **otdInformix_1** (under Invoked Services) to the **esInformixOUT** External Application.
6. From the **jcdDelete** Binding dialog box, map **FileClient_1** to the **FileClientOUT** External Application, as seen in Figure 31.

Figure 31 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.

5.5.7 Creating an Environment

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

Steps required to create an Environment:

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

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

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

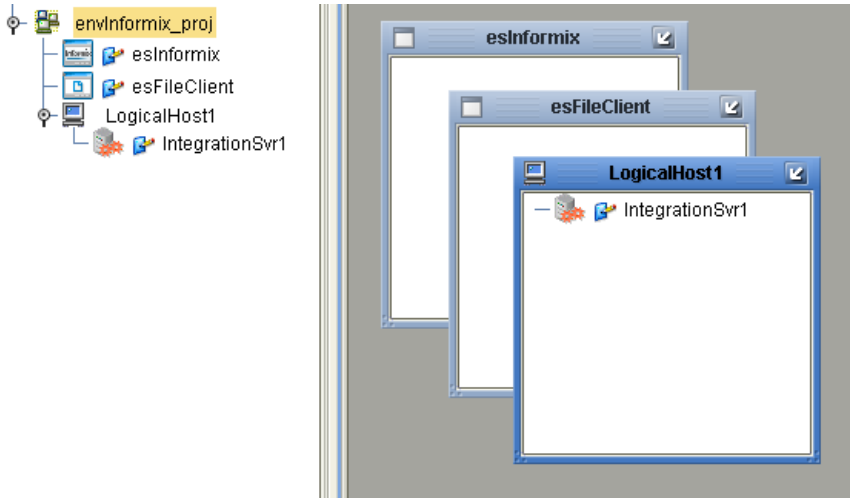
4 Right-click **envInformixProj** and select **New > Informix External System**. Name the External System **esInformix.** Click **OK**. **esInformix** is added to the Environment Editor.

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

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

7 Right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1**. See Figure 32.

Figure 32 Environment Editor - envInformixProj



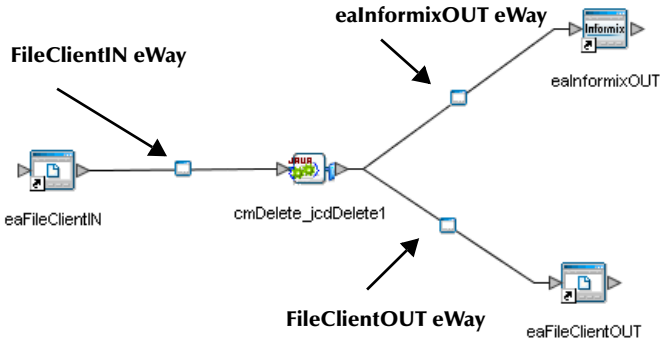
8 Save your current changes to the Repository.

5.5.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjInformix_JCD** sample Project uses three eWays that are represented as nodes between the External Applications and the Business Process. See Figure 33.

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

Figure 33 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 Values
cmDelete	Input file name	TriggerDelete.in
cmInsert	Input file name	TriggerBpInsert.in
cmPsSelect	Input file name	TriggerPsSelect.in
cmTableSelect	Input file name	TriggerTableSelect.in
cmUpdate	Input file name	TriggerUpdate.in

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

Table 15 FileClientOUT eWay Property Settings

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

Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

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

2 Modify the Informix eWay Environment configuration properties for your system (see **eWay External System Properties** on page 25) and click **OK**.

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

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

Table 16 File eWay Environment Properties

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

Configuring the Integration Server

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

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.
3. Click the ellipsis. The **Password Settings** dialog box appears.
4. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.
5. Click **OK** to accept the new property and close the Properties Editor.

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

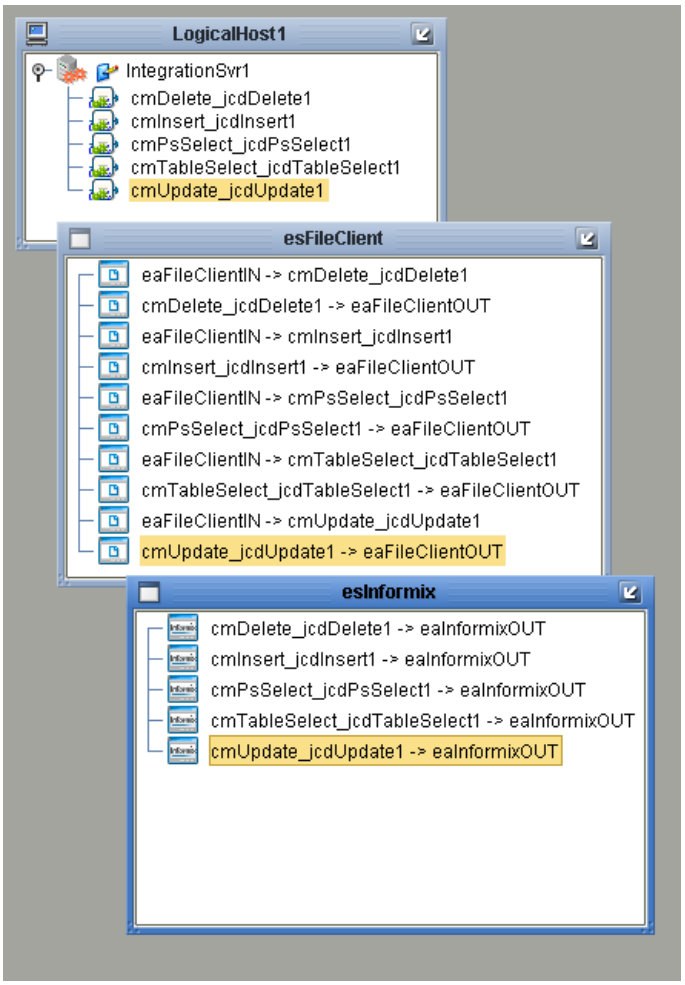
5.5.9 Creating the Deployment Profile

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

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer's Project Explorer, right-click the **prjInfomirx_JCD** Project and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpInformix_JCD**). Select **envInformixProj** as the Environment and click **OK**.
3. From the Deployment Editor toolbar, click the **Automap** icon. The Project's components are automatically mapped to their system windows. See Figure 34.

Figure 34 Deployment Profile



5.5.10 Creating and Starting the Domain

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

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

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your Sun Java Composite Application Platform Suite installation).
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running. Your domain will continue to run unless you shut it down.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.
5. Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.

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

5.5.11 Building and Deploying the Project

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

Build the Project

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

Deploy the Project

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

5.5.12 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.

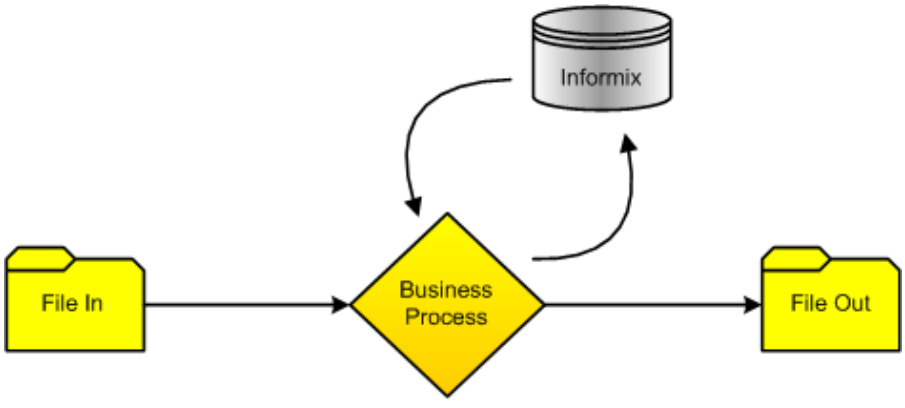
2 Verify the output data by viewing the sample output files. See **About the Informix eWay Sample Projects** on page 48 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.

5.6 Building and Deploying the prjInformix_BPEL Sample Project

The **Informix_BPEL_Sample** sample Project describes how to retrieve the last name and employee number of all employees in the Informix database, using eInsight's BPEL business process engine. In this sample, specific employee information is retrieved by passing specific conditions into an OTD Collaboration which then queries the database and extracts results in the form of an output file.

Figure 35 illustrates the business process used by the sample Project.

Figure 35 Sample Project Data Exchange



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

Steps required to create the sample project include:

5.6.1 Creating a Project

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

1. Start the Enterprise Designer.
2. From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.
3. Right-click **Project1** and select **Rename** from the shortcut menu. Rename the Project (for this sample, **prjInformix_BPEL**).

5.6.2 Creating the OTDs

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

- Informix Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a Informix 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 **Informix Database OTD Wizard** from the list of OTD Wizards and click **Next**.
3. Enter the connection information for the Informix database. Connection fields include:
   - Host Name
   - Port ID

- Informix Server
- Database Name
- 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 Informix database. The **Add Tables** window appears.

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

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

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

- Prepared Statement Name: Select_ps
- SQL Statement:

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

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

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

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

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.

5.6.3 Creating the Business Process

Steps required to create the Business Process include:

- Creating the business process flow
- Configuring the modeling elements

Creating the Business Process Flow

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

Steps to create a business process flow include:

1. Right-click your new Project in the Enterprise Designer's Project Explorer, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename **BusinessProcess1** to **bpInsert**.
2. Create four additional business processes and rename them as follows:
   - bpUpdate
   - bpDelete
   - bpTableSelect
   - bpPsSelect
3. Add the following Activities to the Business Process Designer canvas.

Table 17 Business Process Activities

Business Process	Activity
bpInsert	▪ FileClient.Receive ▪ FileClient.Write ▪ otdInformix.otdInformix.Db_employeeInsert (inside a Scope) ▪ otdInputDTD_Employee.unmarshal ▪ FileClient.Write
bpUpdate	▪ FileClient.receive ▪ FileClient.write ▪ otdInformix.DB_EMPLOYEEUpdate ▪ FileClient.write
bpDelete	▪ FileClient.receive ▪ FileClient.write ▪ otdInformix.DB_EMPLOYEEDelete ▪ FileClient.write

Table 17 Business Process Activities (Continued)

Business Process	Activity
bpTableSelect	▪ FileClient.receive ▪ FileClient.write ▪ otdInformix.DB_EMPLOYEESelectAll ▪ otdOutputDTD_DB_employees.marshal ▪ FileClient.write ▪ FileClient.write
bpPsSelect	▪ FileClient.receive ▪ FileClient.write ▪ otdInformix.Select_psPSSelectAll ▪ Decision ▪ FileClient.write (inside a Scope renamed “No records”) ▪ otdOutputDTD_DB_employees.marshal (inside a Scope renamed “Records found” ▪ FileClient.write (inside a Scope renamed “Records found”) ▪ FileClient.write

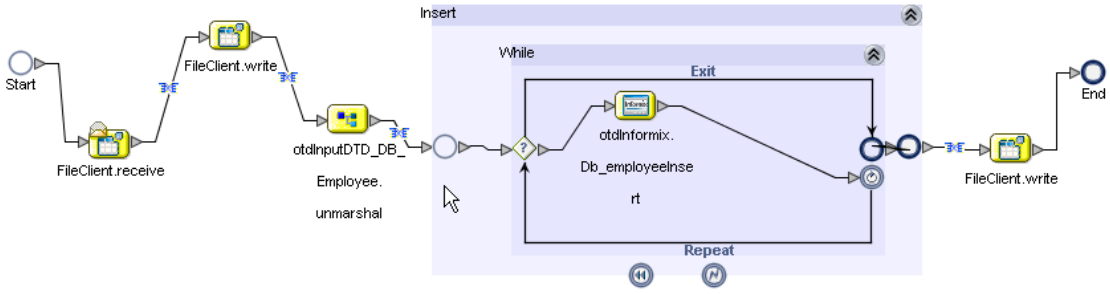
Configuring the bpInsert Modeling Elements

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

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

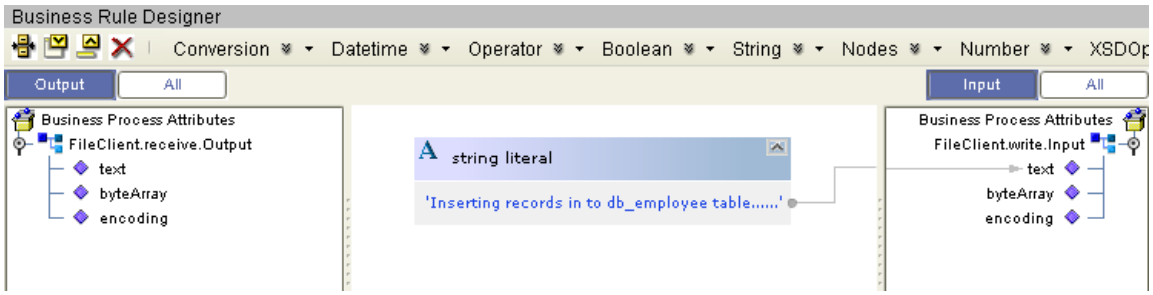
Figure 36 bpInsert Business Process



Steps required to configure the bpInsert business process:

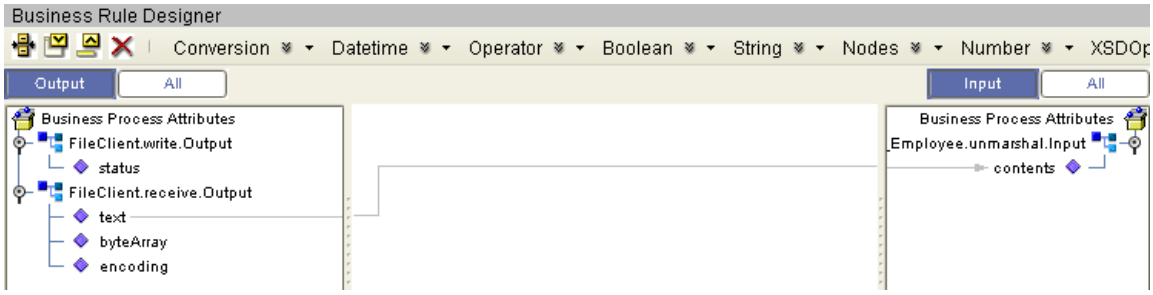
1 Configure the business rule between the **FileClient.receive** and **FileClient.write** Activities, as seen in Figure 37.

Figure 37 bpInsert Business Rule # 1



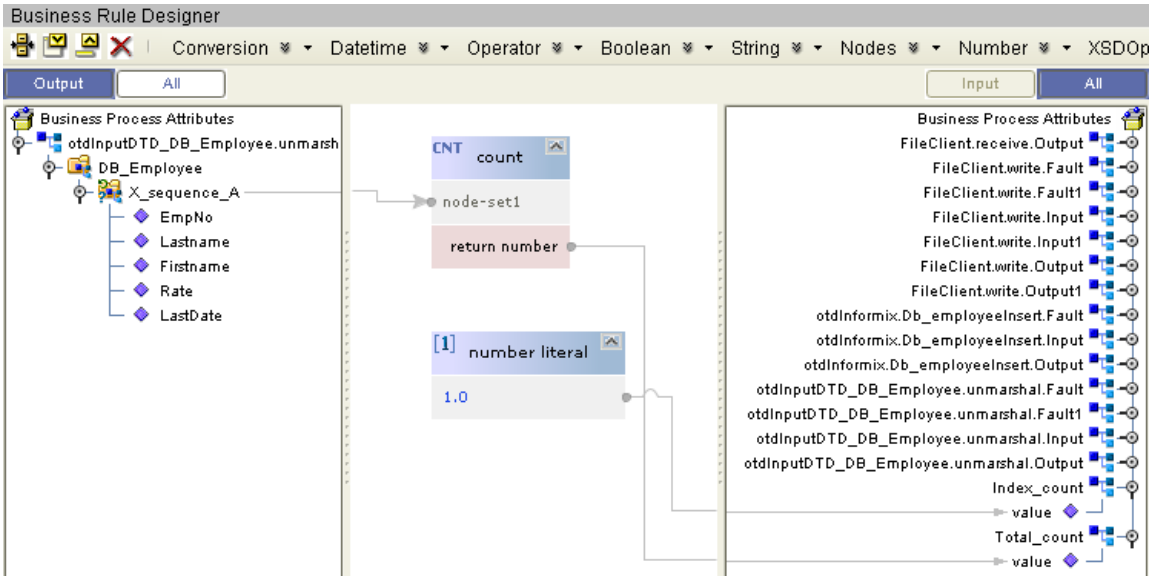
2 Configure the business rule between the **FileClient.write** Activity and **otdInputDTD_DB_Employee.unmarshal** Activity, as seen in Figure 38.

Figure 38 bpInsert Business Rule # 2



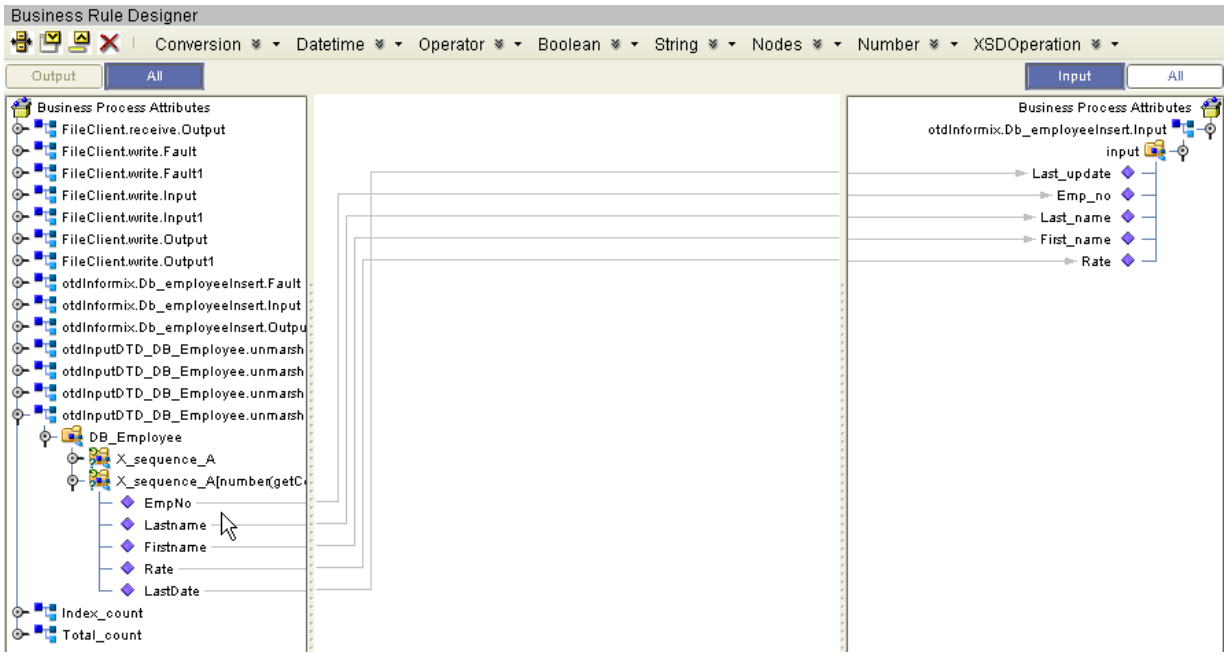
3 Configure the business rule between the **otdInputDTD_DB_Employee.unmarshal** Activity and the **Insert** (Scope element), as seen in Figure 39.

Figure 39 bpInsert Business Rule # 3



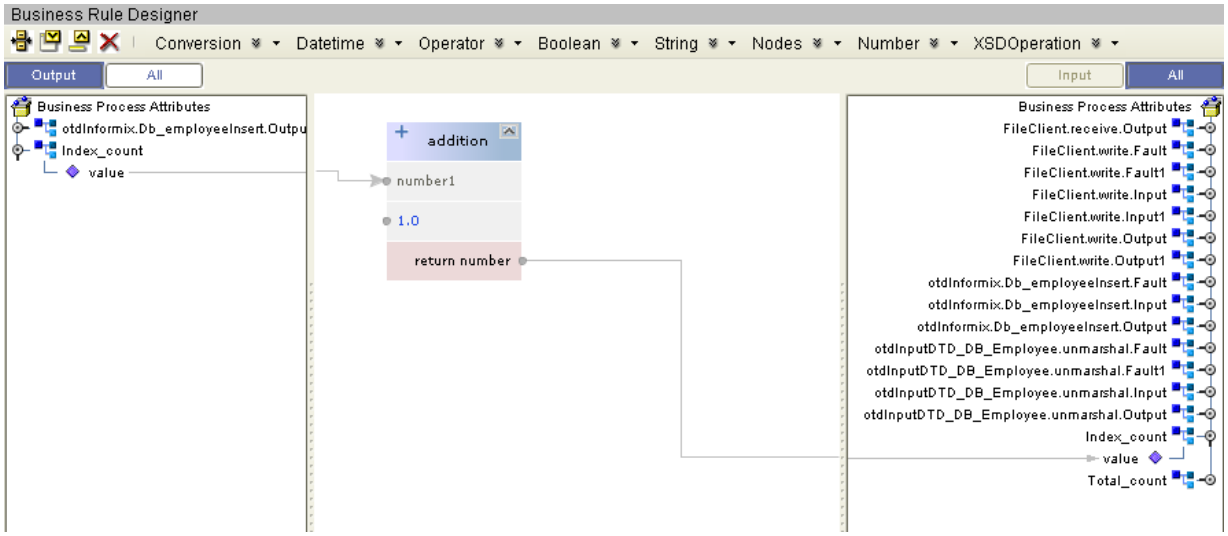
4 Configure the business rule in the **While** statement that connects to the **otdInformix.Db_employeeInsert** Activity, as seen in Figure 40.

Figure 40 bpInsert Business Rule # 4



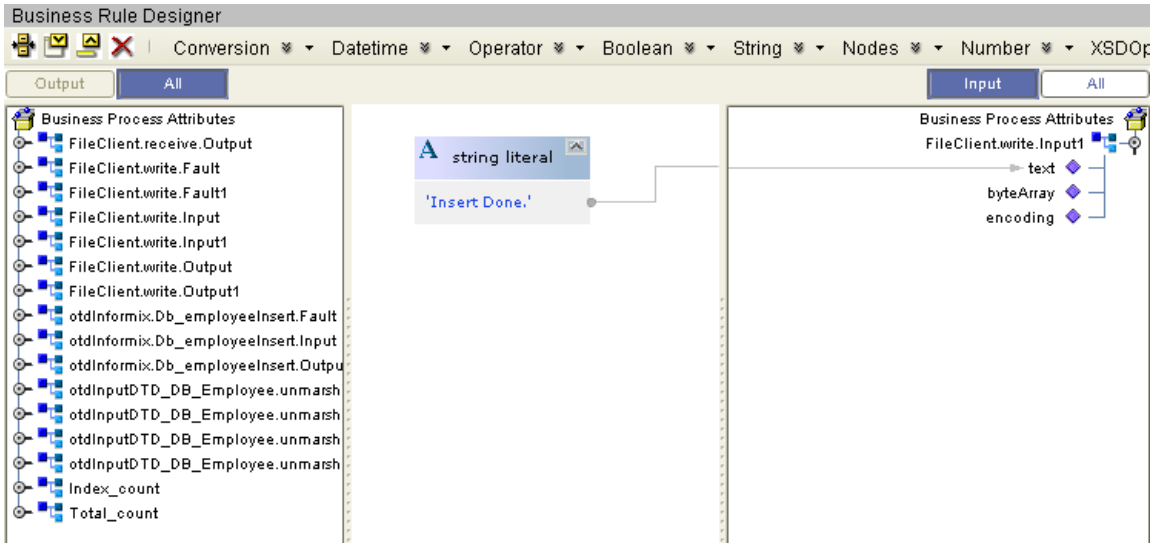
5 Configure the business rule in the **While** statement that connects from the **otdInformix.Db_employeeInsert** Activity, as seen in Figure 41.

Figure 41 bpInsert Business Rule # 5



6 Configure the business rule from the **Insert** (Scope element) to the **FileClient.write** Activity, as seen in Figure 42.

Figure 42 bpInsert Business Rule # 6



Configuring the bpUpdate Modeling Elements

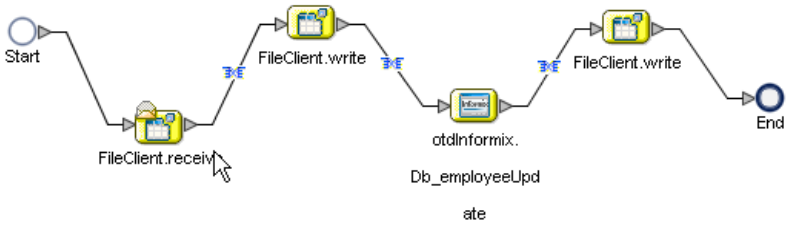
The bpUpdate business process describes how to update a record in the Informix database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Update operation. Figure 43 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 update a specific record. Also note that all records of the table in the database will be updated 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.*

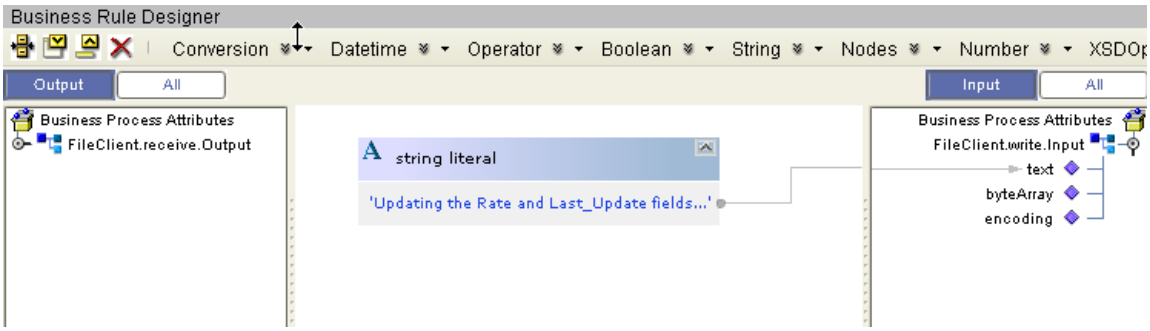
Figure 43 bpUpdate Business Process



Steps required to configure the bpUpdate business process:

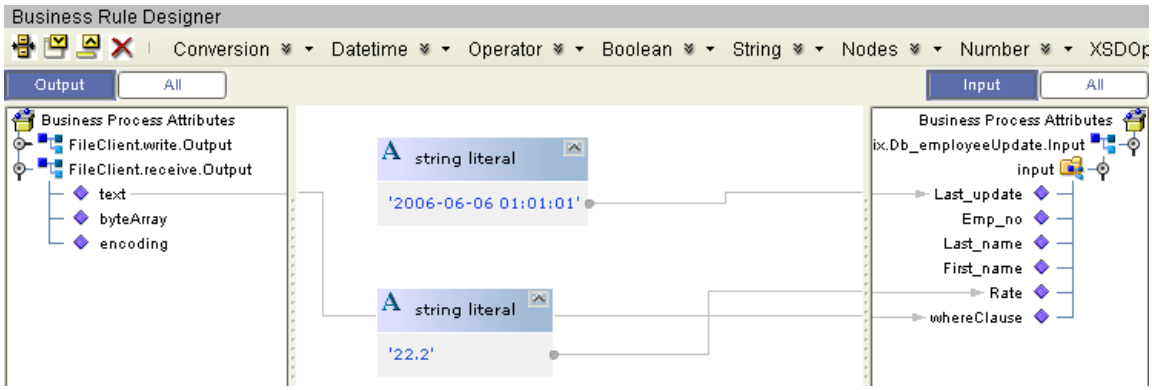
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** Activities, as seen in Figure 44.

Figure 44 bpUpdate Business Rule # 1



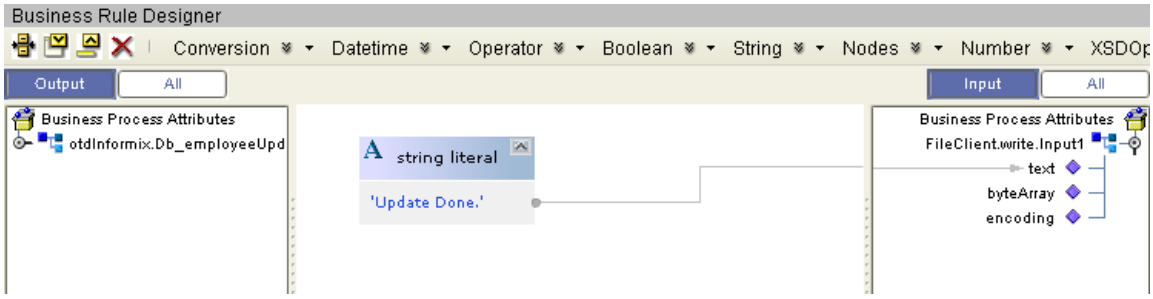
2 Configure the business rule between the **FileClient.write** Activity and **otdInformix.Db_employeeUpdate** Activity, as seen in Figure 45.

Figure 45 bpUpdate Business Rule # 2



3 Configure the business rule between **otdInformix.Db_employeeUpdate** Activities and the **FileClient.write** Activity, as seen in Figure 46.

Figure 46 bpUpdate Business Rule # 3



Configuring the bpDelete Modeling Elements

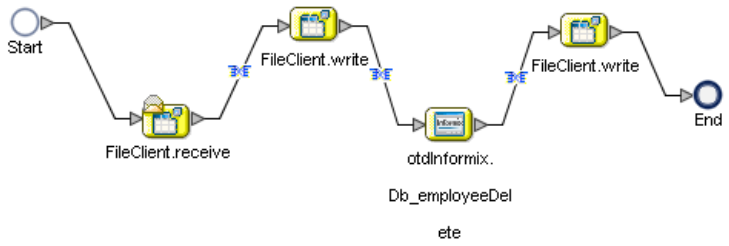
The bpDelete business process describes how to delete a record in the Informix database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Delete operation. See Figure 47 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 delete a specific record. Also note that all records of the table in the database will be deleted when the TriggerDelete.in file is empty.*

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

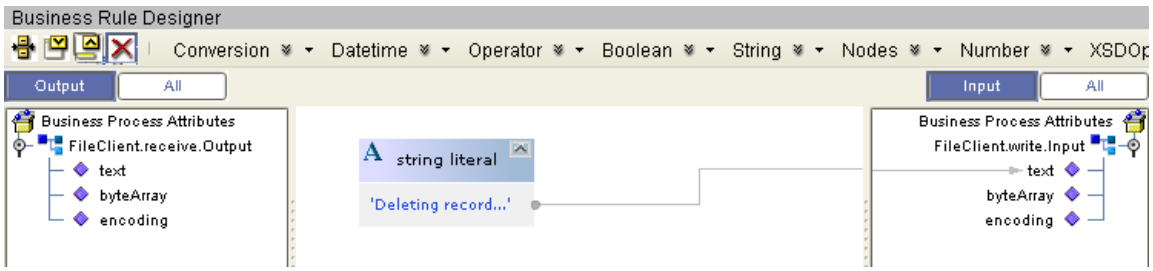
Figure 47 bpDelete Business Process



Steps required to configure the bpDelete business process:

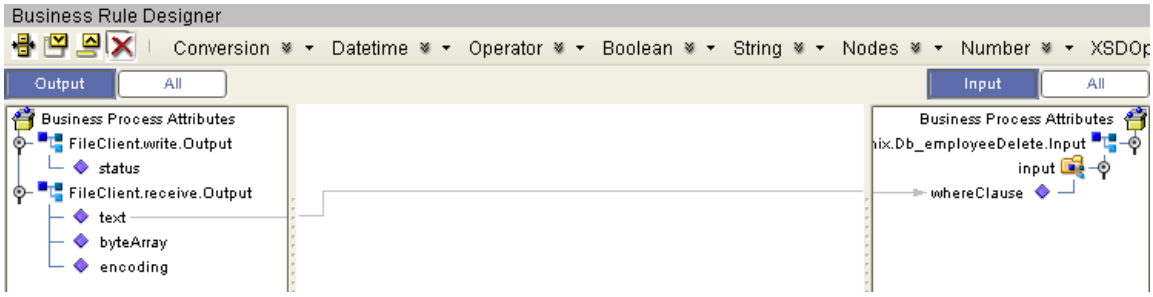
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** Activities, as seen in Figure 48.

Figure 48 bpDelete Business Rule # 1



2 Configure the business rule between the **FileClient.write** Activity and **otdInformix.Db_employeeDelete** Activity, as seen in Figure 49.

Figure 49 bpDelete Business Rule # 2



3 Configure the business rule between the **otdInformix.Db_employeeDelete** Activity and the **FileClient.write** Activity, as seen in Figure 50.

Figure 50 bpDelete Business Rule # 3

Configuring the bpTableSelect Modeling Elements

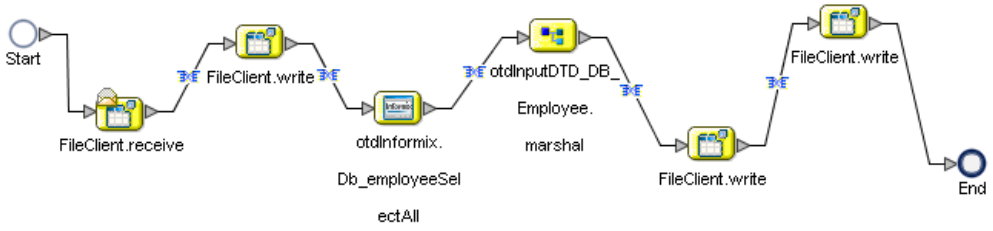
The bpTableSelect business process is describes how to select all records the Informix database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 51 for an illustration of how all the modeling elements appear when connected.

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

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

Figure 51 bpTableSelect Business Process



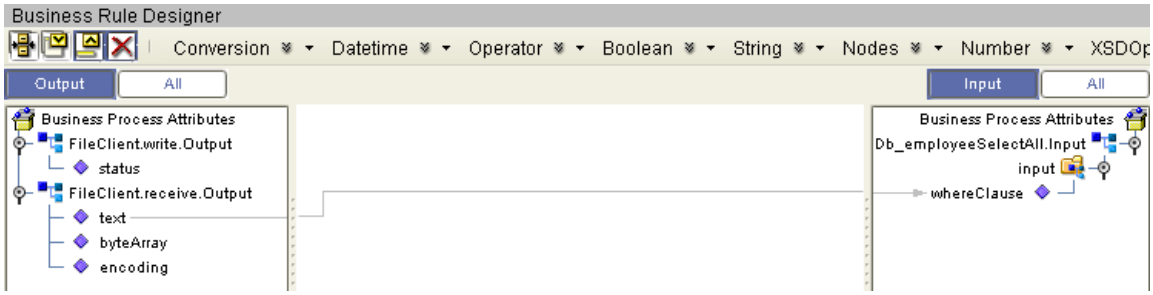
Steps required to configure the bpTableSelect business process:

1 Configure the business rule between **FileClient.receive** and **FileClient.write** Activities, as seen in Figure 52.

Figure 52 bpTableSelect Business Rule # 1

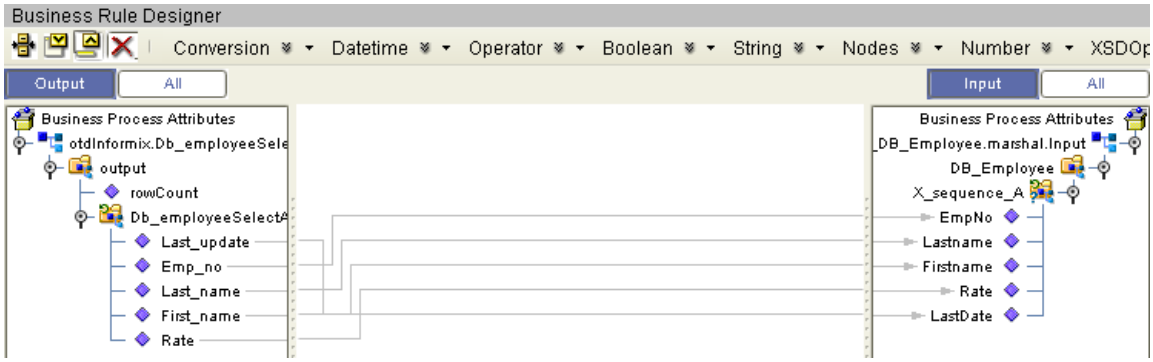
2 Configure the business rule between the **FileClient.write** Activity and **otdInformix.Db_employeeSelectAll** Activity as seen in Figure 53.

Figure 53 bpTableSelect Business Rule # 2



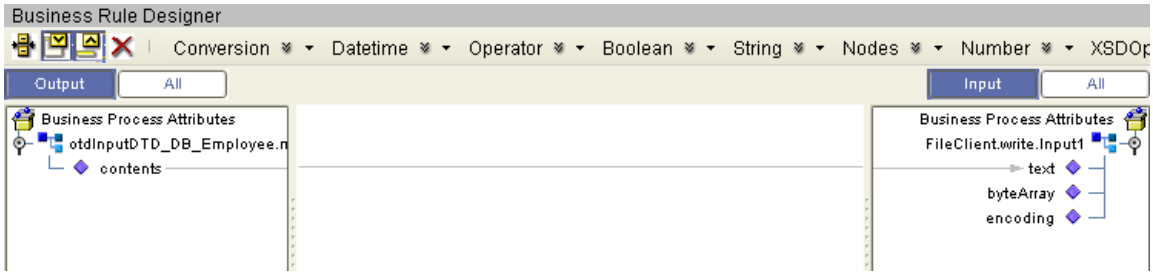
3 Configure the business rule between the **otdInformix.Db_employeeSelectAll** Activity and the **otdOutputDTD_DB_Employee.marshal** Activity as seen in Figure 54.

Figure 54 bpSelectTable Business Rule # 3



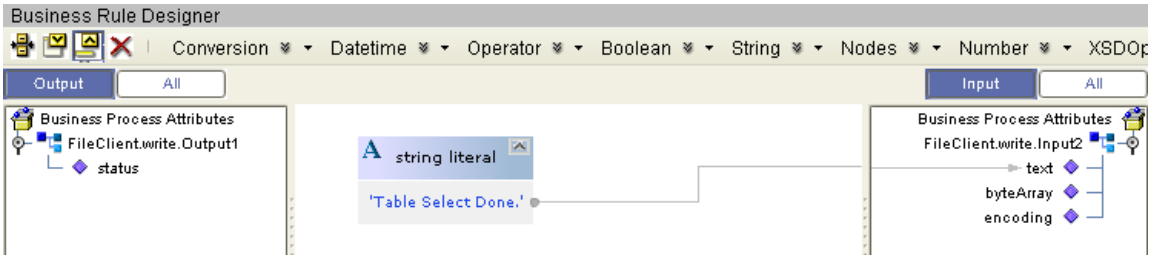
4 Configure the business rule between the **otdOutputDTD_DB_Employee.marshal** Activity and the **FileClient.write** Activity as seen in Figure 55.

Figure 55 bpTableSelect Business Rule # 4



5 Configure the business rule between the **FileClient.write** Activity and the **FileClient.write** Activity as seen in Figure 56.

Figure 56 bpTableSelect Business Rule # 5



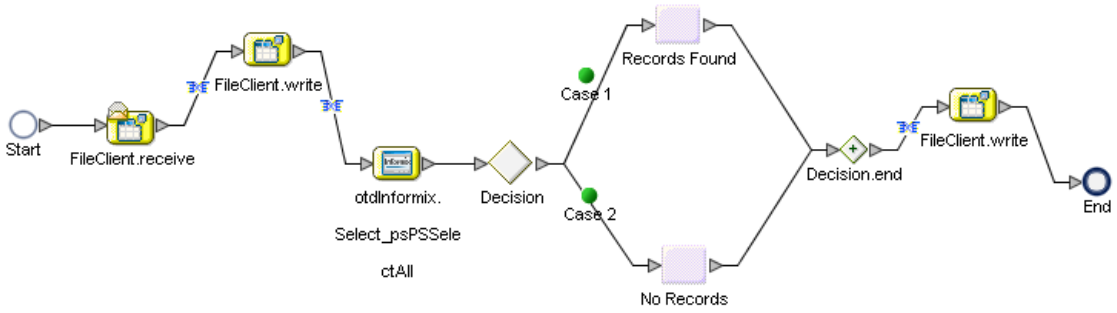
Configuring the bpPsSelect Modeling Elements

The bpPsSelect business process describes how to use a Prepared Statement query to select all records in the Informix 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 57 for an illustration of how all the modeling elements appear when connected.

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

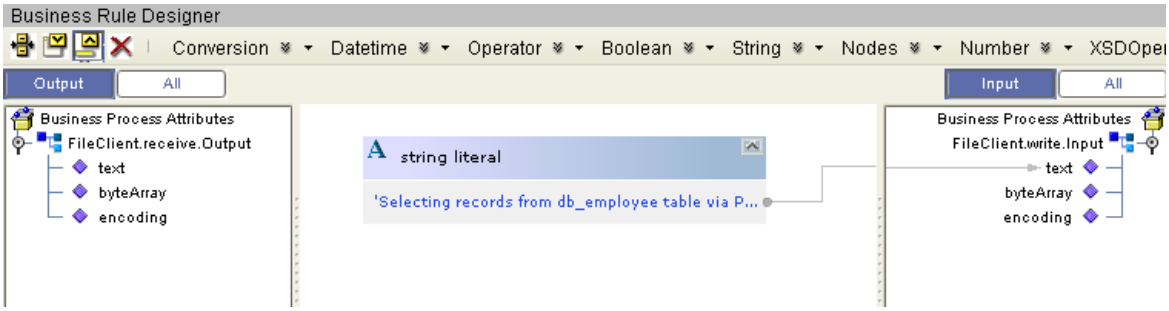
Figure 57 bpPsSelect Business Process



Steps required to configure the bpPsSelect business process:

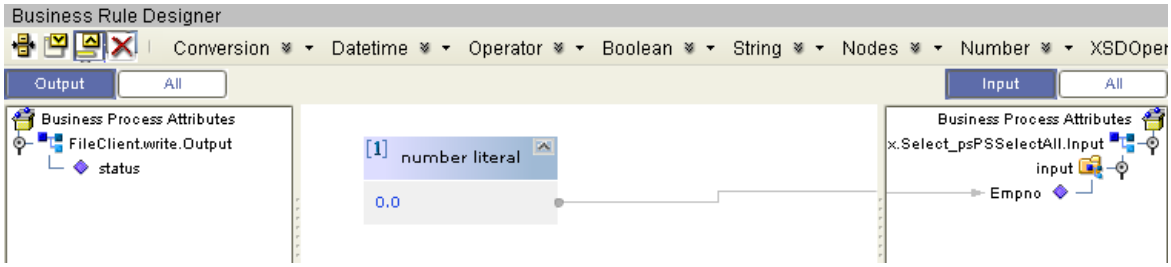
1 Configure the business rule between **FileClient.receive** and **FileCleint.write** Activities as seen in Figure 58.

Figure 58 bpPsSelect Business Rule # 1



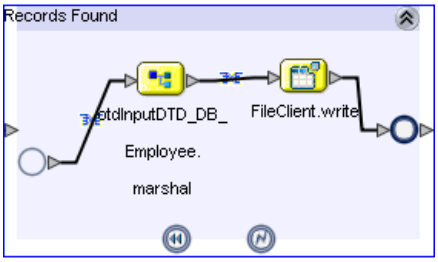
2 Configure the business rule between **FileClient.write** Activity and **otdInformix.SelectpsPSSelectAll** Activity as seen in Figure 59.

Figure 59 bpPsSelect Business Rule # 2



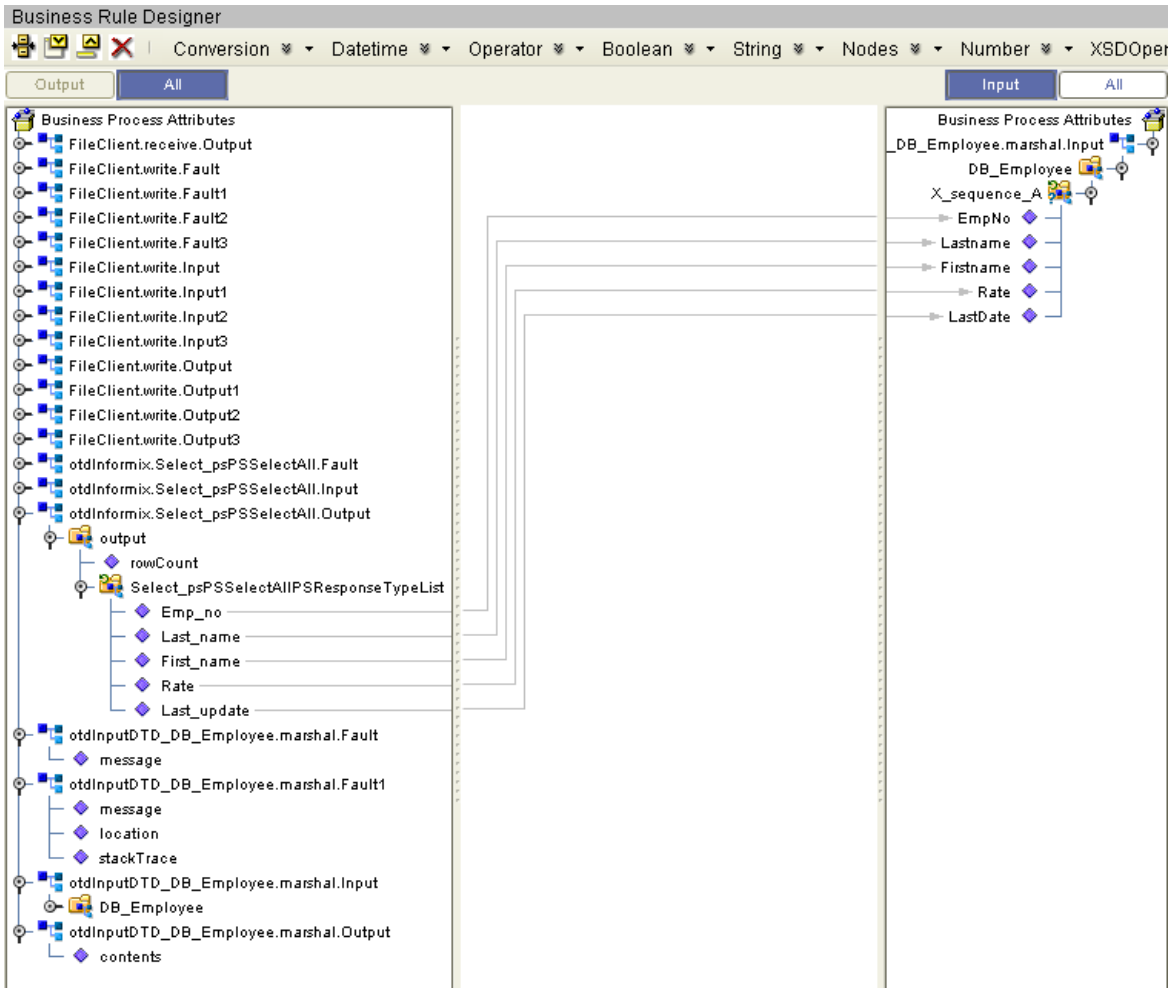
3 Configure **Case 1** of the Decision branching activity. This requires adding business rules between the **otdOutputDTD_DB_Employee.marshal** and the **FileClient.write** Activities within the Scope element.

Figure 60 Activities within Case 1 Scope



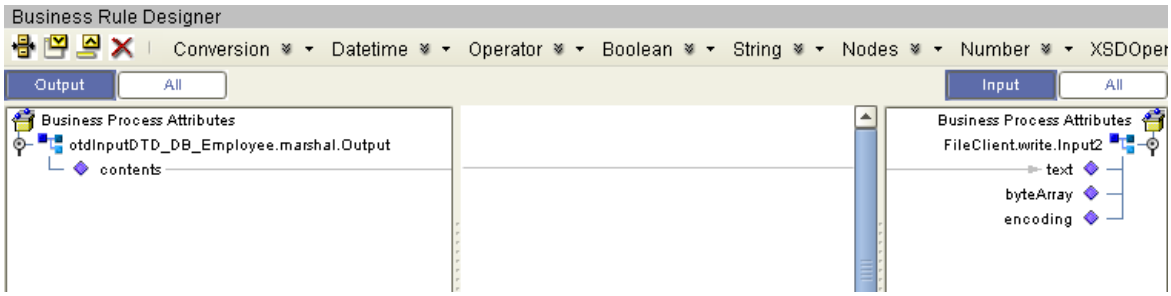
4 Configure the business rule between the start of the Scope element in **Case 1** and the **otdOutputDTD_DB_Employee.marshal** Activity, as seen in Figure 61.

Figure 61 Case 1 Scope Business Rule # 3



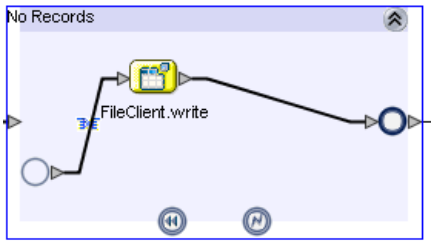
5 Configure the business rule between **otdOutputDTD_DB_Employee.marshal** Activity and **FileClient.write** Activity in the Scope element, as seen in Figure 62.

Figure 62 Case 1 Scope Business Rule # 4



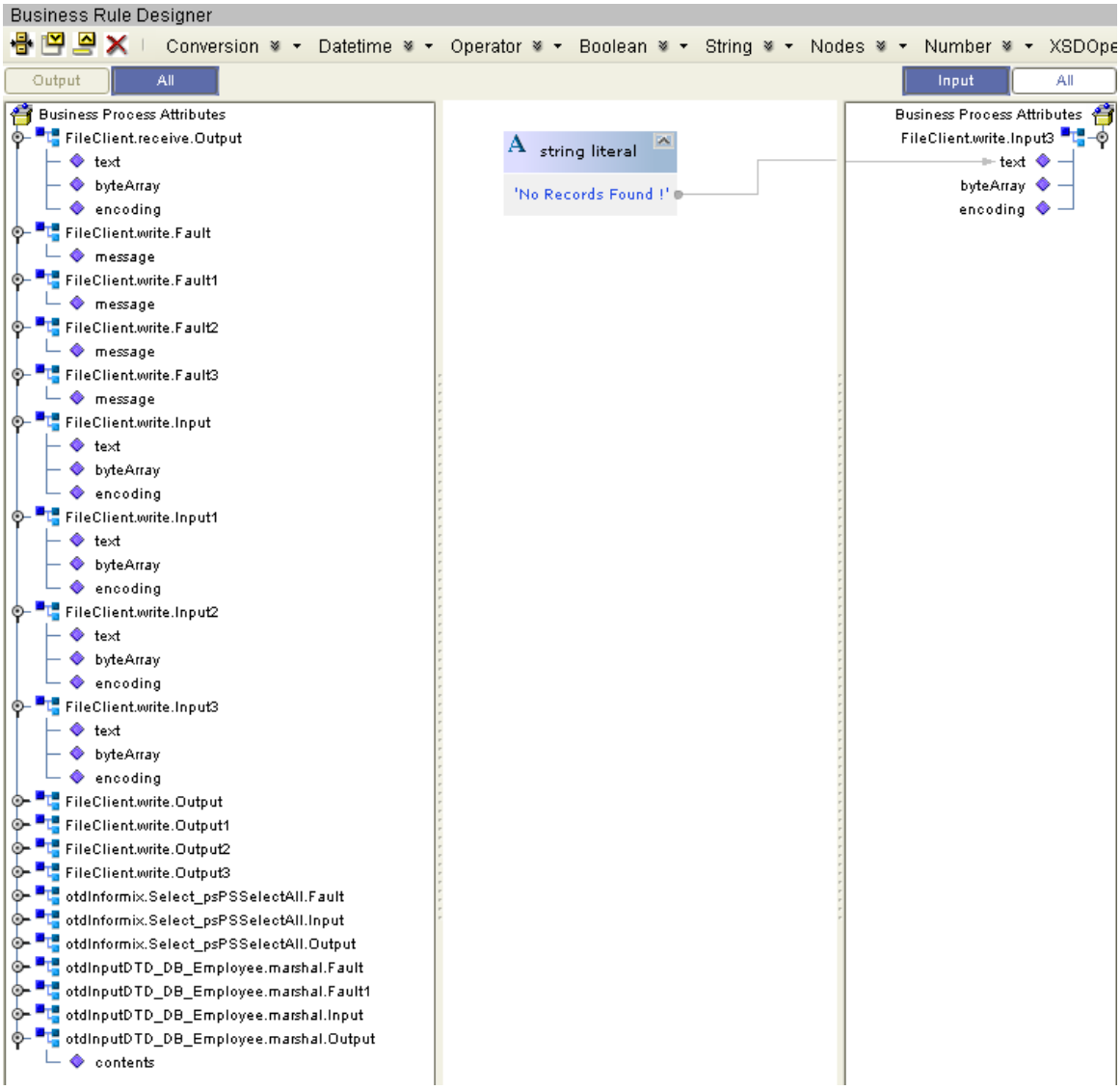
6 Configure Case 2 of the Decision branching activity. This requires adding business rules between the **otdInputDTD_Emp.marshal** and the **FileClient.write** Activities within the Scope element.

Figure 63 Activities within Case 2 Scope



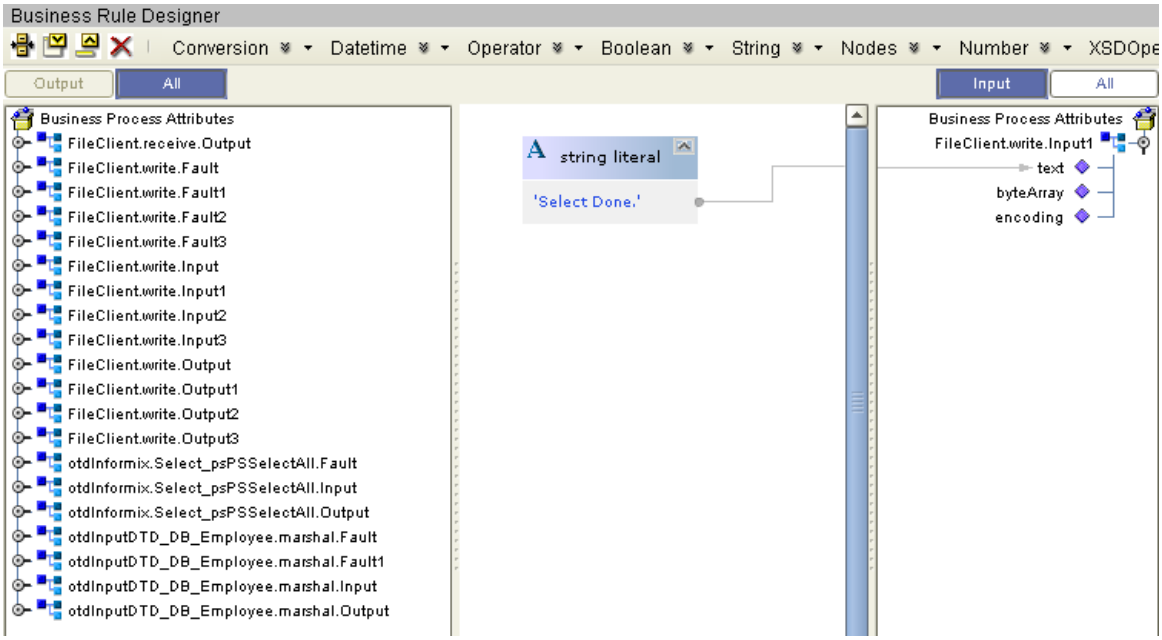
7 Configure the business rule between the start of the Scope element in **Case 2** and the **FileClient.Write** Activity, as seen in Figure 64.

Figure 64 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 65.

Figure 65 bpSelectTable Business Rule # 6



5.6.4 Creating the Connectivity Map

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

Steps required to create the Connectivity Map:

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

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

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

To Select an Informix 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, **Informix** 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
   - Informix1 to eaInformixOUT

To Select a Informix Business Process

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

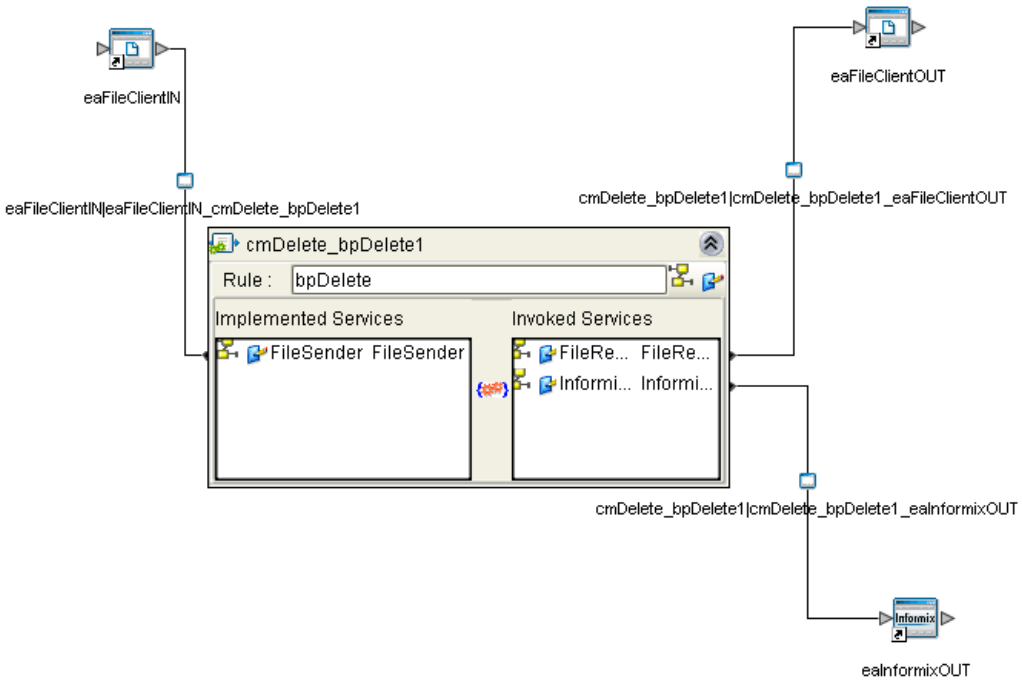
Binding the eWay Components

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 **Informix_otdInformix** (under Invoked Services) to the **eaInformixOUT** External Application.
4. From the **bpDelete** Binding dialog box, map **FileReceiver** to the **FileClientOUT** External Application, as seen in Figure 66.

Figure 66 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.

5.6.5 Creating an Environment

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

Steps required to create an Environment:

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

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

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

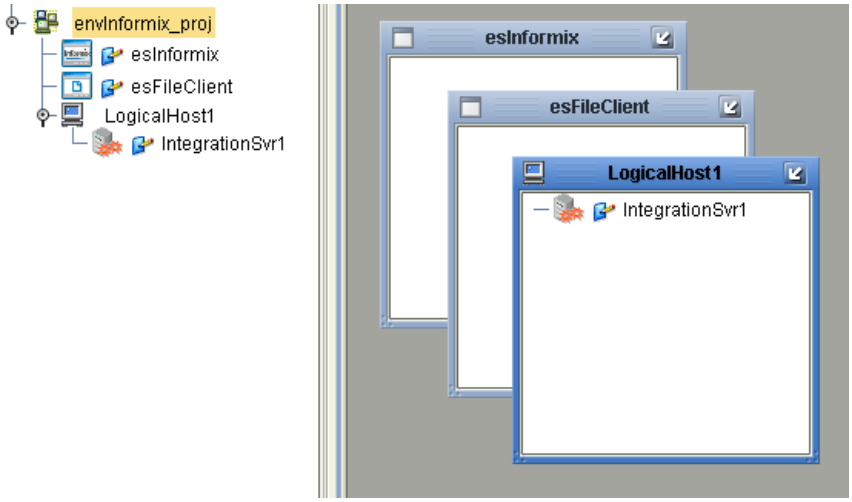
4 Right-click **envInformixProj** and select **New > Informix External System**. Name the External System **esInformix**. Click **OK**. **esInformix** is added to the Environment Editor.

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

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

7 Right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see Figure 67).

Figure 67 Environment Editor - envInformixProj



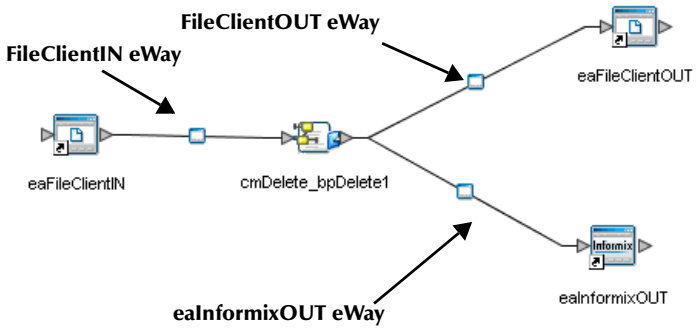
8 Save your current changes to the Repository.

5.6.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the The **prjInformix_BPEL** sample Project use three eWays that are represented as a nodes between the External Applications and the Business Process, as seen in Figure 68.

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

Figure 68 eWays in the cmDelete Connectivity Map



Configuring the eWay Properties

Steps required to configure the eWay properties:

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

Table 18 FileClientIN eWay Property Settings

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

2 Double-click the **FileClientOUT** eWay on each of the Connectivity Maps and modify the properties for your system, as seen in Table 19. Click **OK** to close the Properties Editor.

Table 19 FileClientOUT eWay Property Settings

Connectivity Map	Property Name	Required Values
cmDelete	Output file name	BPEL_Delete_output%d.dat
cmInsert	Output file name	BPEL_Insert_output%d.dat
cmPsSelect	Output file name	BPEL_PsSelect_output%d.dat
cmTableSelect	Output file name	BPEL_TableSelect_output%d.dat
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 Informix External System (**esInformix** in this sample), and select **Properties**. The Properties Editor opens to the Informix eWay Environment configuration.
2. Modify the Informix eWay Environment configuration properties for your system (see **eWay External System Properties** on page 25) and click **OK**.
3. From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the File eWay Environment configuration.
4. Modify the File eWay Environment configuration properties for your system, as seen in Table 20, and click **OK**.

Table 20 File eWay Environment Properties

Section	Property Name	Required Values
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

Configuring the Integration Server

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

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

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

3 Click the ellipsis. The **Password Settings** dialog box appears.

4 Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.

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

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

5.6.7 Creating the Deployment Profile

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

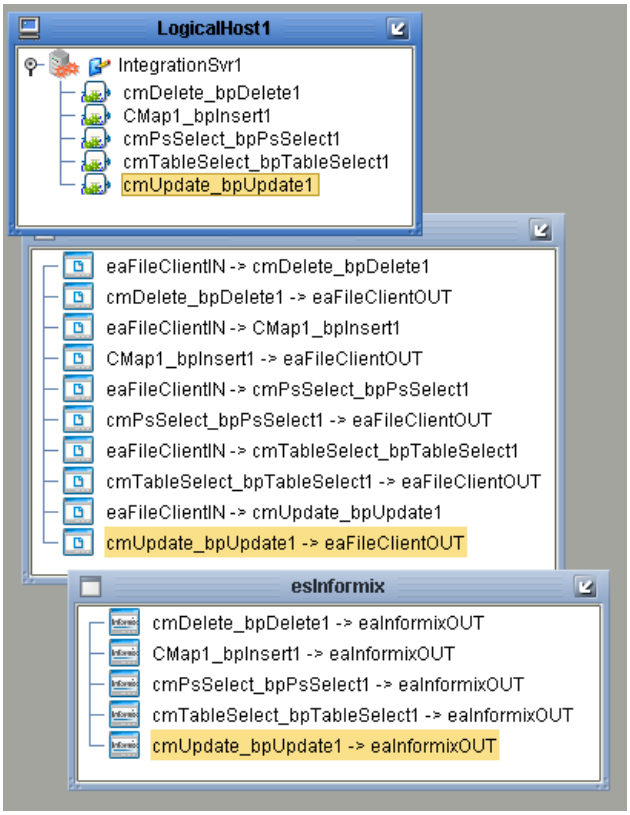
Steps required to create the Deployment Profile:

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

2 Enter a name for the Deployment Profile (for this sample **dpInformix_BPEL**). Select **envInformixProj** 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 69.

Figure 69 Deployment Profile



5.6.8 Creating and Starting the Domain

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

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

Steps required to create and start the domain:

1. Navigate to your **<JavaCAPS51>\logicalhost** directory (where <JavaCAPS51> is the location of your Sun Java Composite Application Platform Suite installation).
2. Double-click the **domainmgr.bat** file. The **Domain Manager** appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the **Start an Existing Domain** button. Once your domain is started, a green check mark indicates that the domain is running.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click **Yes**. The **Create Domain** dialog box appears.
5. Make any necessary changes to the **Create Domain** dialog box and click **Create**. The new domain is added to the Domain Manager. Select the domain and click the **Start**

an Existing Domain button. Once your domain is started, a green check mark indicates that the domain is running.

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

5.6.9 Building and Deploying the Project

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

Build the Project

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

Deploy the Project

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

5.6.10 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 EMPID = 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 Informix eWay Sample Projects** on page 48 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.

5.7 Supported Data Types

Informix eWay supports the following data types:

Table 21 Data Types Supported by Informix

Data Type	Description	OTD/Java Data Type
BOOLEAN	Stores Boolean values true and false.	boolean
BYTE	Stores any kind of binary data.	byte[]
CHAR(n)	Stores single-byte or multibyte sequences of characters, including letters, numbers, and symbols; collation is code-set dependent.	java.lang.String
CHARACTER VARYING(m,r)	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols of varying length (ANSI compliant); collation is code-set dependent.	java.lang.String
DATE	Stores calendar date.	java.sql.Date
DATETIME	Stores calendar date combined with time of day.	java.sql.Timestamp
DECIMAL	Stores numbers with definable scale and precision.	java.math.BigDecimal
DOUBLE PRECISION	Behaves the same way as FLOAT.	double
FLOAT(n)	Stores double-precision floating-point numbers corresponding to the double data type in C.	double
LVARCHAR	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols; collation is locale dependant.	java.lang.String
INTEGER	Stores whole numbers from -2,147,483,647 to +2,147,483,647.	int
INT8	Stores an 8-byte integer value. These whole numbers can be in the range -(263 -1) to 263 -1.	long
MONEY(p,s)	Stores currency amount.	java.math.BigDecimal

Data Type	Description	OTD/Java Data Type
NCHAR(n)	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols; collation is locale dependent.	java.lang.String
NVARCHAR(m,r)	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols of varying length; collation is locale dependent.	java.lang.String
SERIAL	Stores sequential integers.	int
SERIAL8	Stores large sequential integers; has same range as INT8.	long
SMALLFLOAT	Stores single-precision floating-point numbers corresponding to the float data type in C.	float
SMALLINT	Stores whole numbers from -32,767 to +32,767.	short
TEXT	Stores any kind of text data.	java.lang.String
VARCHAR(m,r)	Stores multibyte strings of letters, numbers, and symbols of varying length to a maximum of 255 bytes; collation is code-set dependent.	java.lang.String

5.8 Converting Data Types in Informix eWay

When working with data in the Informix eWay OTDs, use the following conversion chart (Table 22) as a guide for working with Insert and Update operations.

Table 22 Insert and Update Datatype Conversions

Data Type	Java Method or New Constructor to Use(Default: Java Method)	Sample Data
BOOLEAN	**Boolean:** valueOf(java.lang.String)	True/False
BYTE	**Byte:** valueOf(java.lang.String)	n/a
CHAR(n)	**String:** valueOf(java.lang.Object)	Any Character
CHARACTER VARYING(m,r)	**String:** valueOf(java.lang.Object)	Any Character
DATE	**Date:** valueOf(java.lang.String)	2004-12-31

Data Type	Java Method or New Constructor to Use(Default: Java Method)	Sample Data
DATETIME	**Timestamp:** valueOf(long)	2004-12-31 00:00:00
DECIMAL	**BigDecimal:** valueOf(long)	123.45
DOUBLE PRECISION	**Double:** ParseDouble(java.lang.String)	123.45
FLOAT(n)	**Double:** ParseDouble(java.lang.String)	123.45
INTEGER	**Integer:** ParseInt(java.lang.String)	123
INT8	**Long:** parseLong(java.lang.String)	123
LVARCHAR	**String:** java.lang.String	Any Character
MONEY(p,s)	**BigDecimal:** valueOf(long)	123.45
NCHAR(n)	**String:** valueOf(java.lang.Object)	Any Character
NVARCHAR(m,r)	**String:** valueOf(java.lang.Object)	Any Character
SERIAL	**Integer:** ParseInt(java.lang.String)	123.45
SERIAL8	**Long:** parseLong(java.lang.String)	123
SMALLFLOAT	**Float:** parseFloat(java.lang.String)	123.45
SMALLINT	**Short:** parseShort(java.lang.String)	123
TEXT	**String:** valueOf(java.lang.Object)	Any Character
VARCHAR(m,r)	**String:** valueOf(java.lang.Object)	Any Character

When working with data in the Informix eWay OTDs, use the following conversion chart (Table 23) as a guide for working with Select operations.

Table 23 Select Datatype Conversions

Data Type	Java Method or New Constructor to Use(Default: Java Method)	Sample Data
BOOLEAN	**String:** valueOf(boolean)	True/False
CHAR(n)	**Direct Assign**	Any Character
CHARACTER VARYING(m,r)	**Direct Assign**	Any Character
DATE	**String:** valueOf(java.lang.Object)	2004-12-31
DATETIME	**String:** valueOf(java.lang.Object)	2004-12-31 00:00:00
DECIMAL	**String:** valueOf(java.lang.Object)	123.45
DOUBLE PRECISION	**Double:** valueOf(double)	123.45
FLOAT(n)	**String:** valueOf(double)	123.45
INTEGER	**String:** valueOf(int)	123
INT8	**String:** valueOf(long)	123
LVARCHAR	**Direct Assign**	Any Character
MONEY(p,s)	**String:** valueOf(java.lang.Object)	123.45
NCHAR(n)	**Direct Assign**	Any Character
NVARCHAR(m,r)	**Direct Assign**	Any Character
SERIAL	**String:** valueOf(int)	123
SERIAL8	**String:** valueOf(long)	123
SMALLFLOAT	**String:** valueOf(float)	123.45
SMALLINT	**Short:** toString(short)	123
TEXT	**Direct Assign**)	Any Character
VARCHAR(m,r)	**Direct Assign**	Any Character

5.9 Using OTDs with Tables, Views, and Stored Procedures

Tables, Views, and Stored Procedures are manipulated through OTDs. Common operations include insert, delete, update, and query.

5.9.1 The Table

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

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

- SetConcurrencytoUpdatable
- SetConcurrencytoReadOnly
- SetScrollTypetoForwardOnly
- SetScrollTypetoScrollSensitive
- SetScrollTypetoInsensitive

The methods should be called before executing the select() method. For example,

```
getDBEmp().setConcurToUpdatable();
getDBEmp().setScroll_TypeToScrollSensitive();
getDBEmp().getDB_EMPLOYEE().select("");
```

The Query 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 SelectSales;

public class Select
{

    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,db_employee.Db_employeeOTD
db_employee_1,employeedb.Db_employee employeedb_db_employee_1 )
    throws Throwable
    {
        //@map:Db_employee.select(Text)
```

```
            db_employee_1.getDb_employee().select( input.getText() );

            //while
            while (db_employee_1.getDb_employee().next()) {
                //@map:Copy EMP_NO to Employee_no
                employeedb_db_employee_1.setEmployee_no(
    java.lang.Integer.toString(
    db_employee_1.getDb_employee().getEMP_NO() ) );

                //@map:Copy LAST_NAME to Employee_lname
                employeedb_db_employee_1.setEmployee_lname(
    db_employee_1.getDb_employee().getLAST_NAME() );

                //@map:Copy FIRST_NAME to Employee_fname
                employeedb_db_employee_1.setEmployee_fname(
    db_employee_1.getDb_employee().getFIRST_NAME() );

                //@map:Copy RATE to Rate
                employeedb_db_employee_1.setRate(
    java.lang.Double.toString(
    db_employee_1.getDb_employee().getRATE() ) );

                //@map:Copy LAST_UPDATE to Update_date
                employeedb_db_employee_1.setUpdate_date(
    db_employee_1.getDb_employee().getLAST_UPDATE().toString() );


            //@map:Copy employeedb_db_employee_1.marshalToString to
    Text
            FileClient_1.setText(
    employeedb_db_employee_1.marshalToString() );

            //@map:FileClient_1.write
            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.

```
    //DB_EMPLOYEE.insert
        Table_OTD_1.getDB_EMPLOYEE().insert();

//Copy EMP_NO to EMP_NO
   insert_DB_1.getInsert_new_employee().setEmployee_no(
   java.lang.Integer.parseInt(
     employeedb_with_top_db_employee_1.getEmployee_no() ) );

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

//@map:Copy Employee_fname to Employee_Fname
   insert_DB_1.getInsert_new_employee().setEmployee_Fname(
   employeedb_with_top_db_employee_1.getEmployee_fname() );
```

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

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

//@map:Insert Row
   Table_OTD_1.getDB_EMPLOYEE().insertRow();

}
```

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()**.

```
//SalesOrders_with_top_SalesOrders_1.unmarshalFromString(Text)
  SalesOrders_with_top_SalesOrders_1.unmarshalFromString(
  input.getText() );

//SALES_ORDERS.update("SO_num =99")
  DB_sales_orders_1.getSALES_ORDERS().update( "SO_num ='01'" );

//while
  while (DB_sales_orders_1.getSALES_ORDERS().next()) {

//Copy SalesOrderNum to SO_num
  DB_sales_orders_1.getSALES_ORDERS().setSO_num(
  SalesOrders_with_top_SalesOrders_1.getSalesOrderNum() );

//Copy CustomerName to Cust_name
  DB_sales_orders_1.getSALES_ORDERS().setCust_name(
  SalesOrders_with_top_SalesOrders_1.getCustomerName() );

//Copy CustomerPhone to Cust_phone
  DB_sales_orders_1.getSALES_ORDERS().setCust_phone(
  SalesOrders_with_top_SalesOrders_1.getCustomerPhone() );

//SALES_ORDERS.updateRow
  DB_sales_orders_1.getSALES_ORDERS().updateRow();
}

}
```

The Delete Operation

To perform a delete operation on a table

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

In this example DELETE an employee.

```
//@map:Db_employee.delete("first_name like 'John%'")
otdDB_EMPLOYEE_1.getDb_employee().delete("first_name like 'John%'");
```

Prepared Statement

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

Prepared statements can be used to perform insert, update, delete and query operations. A prepared statement uses a question mark (?) as a place holder for input. For example:

```
insert into EMP_TAB(Age, Name, Dept No) value(?, ?, ?)
```

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

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

Batch Operations

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

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

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

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

5.9.2 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" inbound parameter (INLOCALID). This inbound parameter is generated by the DataBase Wizard and is represented in the resulting OTD as nodes. Within the Transformation Designer, you can drag values from the input parameters, and execute the call.

Below are the steps for executing the Stored Procedure:

1. Specify the input values.
2. Execute the Stored Procedure.

For example:

```
package Storedprocedure;

public class sp_jce
{

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

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

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

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

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

        //@map:Copy Employee_fname to Employee_Fname
        insert_DB_1.getInsert_new_employee().setEmployee_Fname(
employeedb_with_top_db_employee_1.getEmployee_fname() );

        //@map:Copy java.lang.Float.parseFloat(Rate) to Rate
```

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

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

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

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

        //@map:FileClient_1.write
        FileClient_1.write();
    }

}
```

Manipulating the ResultSet and Update Count Returned by Stored Procedure

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

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

DataDirect driver for Infomix requires the following syntax when defining a Stored Procedure ResultSet:

```
CREATE PROCEDURE user_defined (arg0 datatype0, ..., argN datatypeN)
RETURNING returntype AS return_param_name0, ..., returntype AS
return_param_nameN;
...
END PROCEDURE;

CREATE FUNCTION user_defined (arg0 datatype0, ..., argN datatypeN)
RETURNING returntype AS return_param_name0, ..., returntype AS
return_param_nameN;
...
END FUNCTION;
```

The DataDirect driver must use the AS keyword in the returning clause of a Stored Procedure ResultSet to name a returned parameter. The AS keyword is only available in 9.4 version of Informix server, so the Stored Procedure ResultSet is only supported in Informix 9.4.

ResultSets generated in the OTD must match the return parameter name and type in the returning clause in the Stored Procedure definition. The Informix OTD Wizard automatically generates an OTD when using the By Executing mode, but when using the Manual or With Assistance mode, the Original Names must match the returning parameter names defined by the AS keyword. Using the Manual mode requires adding column items manually, but using the With Assistance mode might populate erroneous data in the Original Name column, which must be manually changed to match the return parameter names.

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

The **resultsAvailable()** method, added to the OTD, simplifies the whole process of determining whether any results, 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 OTD 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 OTD nodes to the Business Rules. Below is a code snippet that can be generated by the Collaboration Editor:

```
// resultsAvailable() will be true if there's an update count and/or a
result set available.
// note, it should not be called indiscriminantly because each time
the results pointer is
// advanced via getMoreResults() call.
while (getSPIn().getSpS_multi().resultsAvailable())
{
    // check if there's an update count
    if (getSPIn().getSpS_multi().getUpdateCount() > 0)
    {
        logger.info("Updated
"+getSPIn().getSpS_multi().getUpdateCount()+" rows");
    }
// each result set node has an available() method (similar to OTD's)
that tells the user
// whether this particular result set is available. note, JDBC does
support access to
// more than one result set at a time, i.e., cannot drag from 2
distinct result sets
// simultaneously
    if (getSPIn().getSpS_multi().getNormRS().available())
    {
    while (getSPIn().getSpS_multi().getNormRS().next())
    {
    logger.info("Customer Id =
"+getSPIn().getSpS_multi().getNormRS().getCustomerId());
    logger.info("Customer Name =
"+getSPIn().getSpS_multi().getNormRS().getCustomerName());
    }
    if (getSPIn().getSpS_multi().getDbEmployee().available())
    {
    while (getSPIn().getSpS_multi().getDbEmployee().next())
    {
    logger.info("EMPNO =
"+getSPIn().getSpS_multi().getDbEmployee().getEMPNO());
    logger.info("ENAME =
"+getSPIn().getSpS_multi().getDbEmployee().getENAME());
    logger.info("JOB =
"+getSPIn().getSpS_multi().getDbEmployee().getJOB());
    logger.info("MGR =
"+getSPIn().getSpS_multi().getDbEmployee().getMGR());
    logger.info("HIREDATE =
"+getSPIn().getSpS_multi().getDbEmployee().getHIREDATE());
    logger.info("SAL =
"+getSPIn().getSpS_multi().getDbEmployee().getSAL());
    logger.info("COMM =
"+getSPIn().getSpS_multi().getDbEmployee().getCOMM());
    logger.info("DEPTNO =
"+getSPIn().getSpS_multi().getDbEmployee().getDEPTNO());
    }
}
```

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

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.

Index

P

S

T