

SeeBeyond ICAN Suite

SQL Server eWay Intelligent Adapter User's Guide

Release 5.0.3



The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e*Gate, and e*Way are the registered trademarks of SeeBeyond Technology Corporation in the United States and select foreign countries; the SeeBeyond logo, e*Insight, and e*Xchange are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2003-2004 by SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence.

Version 20040422105858.

Contents

Chapter 1

Introduction	6
Overview	6
Supported Operating Systems	6
System Requirements	7
External System Requirements	7

Chapter 2

Installation	8
Before Installing the eWay	8
Installing the SQL Server eWay	8
After Installation	9

Chapter 4

Properties of the SQL Server eWay	10
Setting the eWay Properties in the Connectivity Map	10
Setting the Properties in the Outbound eWay	10
ClassName	11
Description	11
InitialPoolSize	12
LoginTimeout	12
MaxIdleTime	12
MaxPoolSize	12
MaxStatements	12
MinPoolSize	13
NetworkProtocol	13
PropertyCycle	13
RoleName	13
Setting the Properties in the Inbound eWay	14
Pollmilliseconds	14
PreparedStatement	14
Setting the Properties in the Outbound eWay Environment	15
DatabaseName	15

DataSourceName	15
Delimiter	16
Description	16
DriverProperties	16
Password	16
PortNumber	16
ServerName	17
User	17
Setting the Properties in the Inbound eWay Environment	18
DatabaseName	18
Password	18
PortNumber	19
ServerName	19
User	19
Setting the Properties in the Outbound eWay with XA Support	20
ClassName	20
Description	20
InitialPoolSize	21
LoginTimeOut	21
MaxIdleTime	21
MaxPoolSize	21
MaxStatements	21
MinPoolSize	22
NetworkProtocol	22
PropertyCycle	22
RoleName	22
Setting the Properties in the Outbound eWay XA Environment	23
DatabaseName	23
DataSourceName	23
Delimiter	24
Description	24
DriverProperties	24
Password	24
PortNumber	25
ServerName	25
User	25

Chapter 5

Using the SQL Server eWay Database Wizard	26
Using the Database OTD Wizard	26
To create a new OTD using the Database Wizard	27

Chapter 6

Working with the Sample Project(s)	37
eInsight Engine and eGate Components	37
Using the Sample Project in eInsight	37
The Business Process	38

Contents

whereClause()	40
SelectAll	40
SelectMultiple	41
SelectOne	43
Insert	44
Update	46
Delete	47
Using the Sample Project in eGate	49
Working with the Sample Project in eGate	49
Configuring the eWays	50
Creating the External Environment	51
Deploying a Project	51
Running the Sample	51
Common DataType Conversions	51
Using OTDs with Tables, Views, and Stored Procedures	53
The Table	53
The Query Operation	53
The Insert Operation	54
The Update Operation	55
The Delete Operation	55
The Stored Procedure	56
Executing Stored Procedures	56
Manipulating the ResultSet and Update Count Returned by Stored Procedure	57
Alerting and Logging	59
Index	60

Introduction

This document describes how to install and configure the eWay Intelligent Adapter for SQL Server.

This Chapter Includes:

- [Overview](#) on page 6
- [Supported Operating Systems](#) on page 6
- [System Requirements](#) on page 7
- [External System Requirements](#) on page 7

1.1 Overview

The SQL Server eWay enables the eGate system to exchange data with external SQL Server databases. Using the java library, SQL statements are issued to interact with the SQL Server databases.

1.2 Supported Operating Systems

The SQL Server eWay is available on the following operating systems:

- Windows Server 2003, Windows XP, and Windows 2000
- HP Tru64 V5.1A
- HP-UX 11.0 and 11i
- HP-UX 11i V2 (11.23)
- IBM AIX 5.1 and 5.2
- Red Hat Linux 8.0
- Red Hat Enterprise Linux AS 2.1
- Sun Solaris 8 and 9

1.3 System Requirements

The system requirements for the SQL Server eWay are the same as for eGate Integrator. For information, refer to the *eGate Integrator Installation Guide*. It is also helpful to review the **Readme.txt** for any additional requirements prior to installation. The **Readme.txt** is located on the installation CD-ROM.

***Note:** To enable Web Services, you must install and configure the SeeBeyond ICAN Suite eInsight Business Process Manager.*

1.4 External System Requirements

The SQL Server eWay supports the following software for external systems running eGate Projects.

- SQL Server 7 and SQL Server 2000
- Driver support for Data Direct Drivers JDBC 3.3

Installation

This chapter describes how to install the SQL Server eWay.

This Chapter Includes:

- **Before Installing the eWay** on page 8
- **Installing the SQL Server eWay** on page 8
- **After Installation** on page 9

2.1 Before Installing the eWay

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

2.2 Installing the SQL Server eWay

During the eGate Integrator installation process, the Enterprise Manager, a web-based application, is used to select and upload eWays (eWay.sar files) from the eGate installation CD-ROM to the Repository.

The installation process includes installing the following components:

- Installing the Repository
- Uploading products to the Repository
- Downloading components (such as Enterprise Designer and Logical Host)
- Viewing product information home pages

Follow the instructions for installing the eGate Integrator in the *ICAN Installation Guide*, and include the following steps:

- On the Enterprise Manager, select the **SQLServereWay.sar** (to install the SQL Server eWay) file to upload.
- On the Enterprise Manager, select the **FileeWay.sar** (to install the File eWay, used in the sample Project) file to upload.

- On the Enterprise Manager, install the **SQLServereWayDocs.sar** (to install the documentation and the sample) file to upload.
- On the Enterprise Manager under the Documentation tab, click on the document link or the sample file link. For the sample project, it is recommended that you extract the file to another file location prior to importing it using the Enterprise Explorer's Import Project tool.

For additional information on how to use eGate, please see the *eGate Tutorial*.

Continue installing the eGate Enterprise Designer as instructed.

2.3 After Installation

Once the eWay is installed and configured 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.

Properties of the SQL Server eWay

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

This Chapter Includes:

- [Setting the Properties in the Outbound eWay](#) on page 10
- [Setting the Properties in the Inbound eWay](#) on page 14
- [Setting the Properties in the Outbound eWay Environment](#) on page 15
- [Setting the Properties in the Inbound eWay Environment](#) on page 18
- [Setting the Properties in the Outbound eWay with XA Support](#) on page 20
- [Setting the Properties in the Outbound eWay XA Environment](#) on page 23

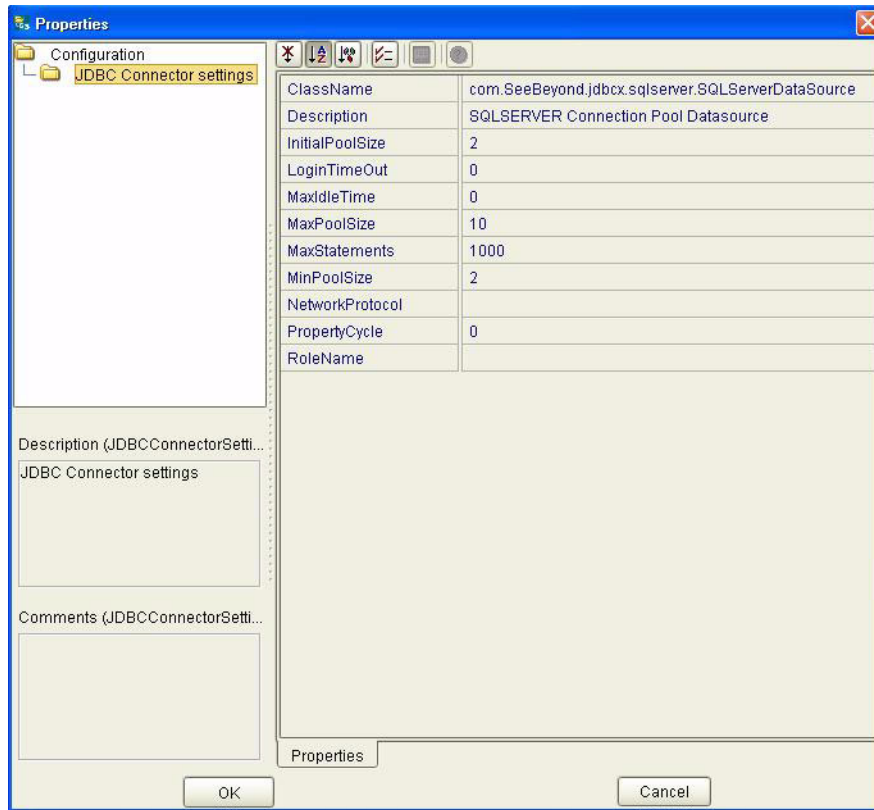
4.1 Setting the eWay Properties in the Connectivity Map

On the Properties sheet window and using the descriptions below, enter the information necessary for the eWay to establish a connection to the external application.

4.1.1. Setting the Properties in the Outbound eWay

The DataSource settings define the properties used to interact with the external database.

Figure 1 The eWay Properties



The DataSource settings define the properties used to interact with the external database.

ClassName

Description

Specifies the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.

Required Values

A valid class name.

The default is **com.SeeBeyond.jdbcx.sqlserver.SQLServerDataSource**.

Description

Description

Enter a description for the database.

Required Value

A valid string.

InitialPoolSize

Description

Enter a number for the physical connections the pool should contain when it is created.

Required Value

A valid numeric value. The default is 2.

LoginTimeout

Description

The number of seconds driver will wait before attempting to log in to the database before timing out.

Required Value

A valid numeric value.

MaxIdleTime

Description

The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.

Required Value

A valid numeric value.

MaxPoolSize

Description

The maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum.

Required Value

A valid numeric value. The default is 10.

MaxStatements

Description

The maximum total number of statements that the pool should keep open. 0 (zero) indicates that the caching of statements is disabled.

Required Value

A valid numeric value. The default is 1000.

MinPoolSize

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.

Required Value

A valid numeric value. The default is 2.

NetworkProtocol

Description

The network protocol used to communicate with the server.

Required Values

Any valid string.

PropertyCycle

Description

The interval, in seconds, that the pool should wait before enforcing the current policy defined by the values of the other connection pool properties in this deployment descriptor.

Required Values

A valid numeric value. The default is 0.

RoleName

Description

An initial SQL role name.

Required Values

Any valid string.

4.1.2. Setting the Properties in the Inbound eWay

Figure 2 Properties of the Inbound SQL Server eWay



Pollmilliseconds

Description

Polling interval in milliseconds.

Required Value

A valid numeric value. The default is 5000.

PreparedStatement

Description

Prepared Statement used for polling against the database.

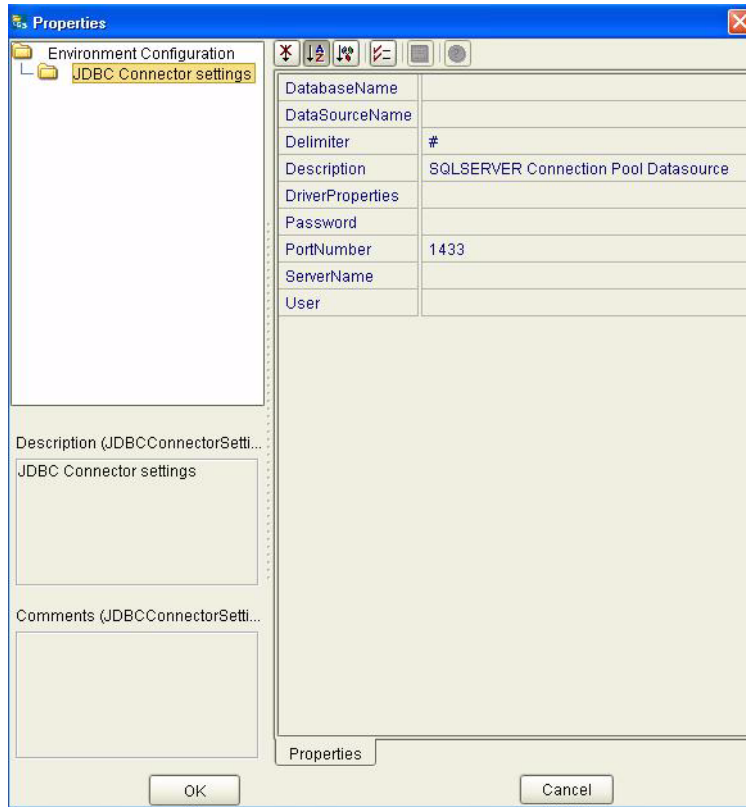
Required Value

The Prepared Statement must be the same Prepared Statement you created using the Database OTD Wizard. Only SELECT Statement is allowed. Additionally, no place holders should be specified. There should not be any “?” in the Prepared Query.

4.1.3. Setting the Properties in the Outbound eWay Environment

Before deploying your eWay, you will need to set the properties of the eWay environment using the following descriptions.

Figure 3 eWay Environment Configuration



DatabaseName

Description

Specifies the name of the database instance.

Required Values

Any valid string.

DataSourceName

Description

Provide the name of the ConnectionPoolDataSource object that the DataSource object delegates behind the scenes when connection pooling or distributed transaction management is being done.

Required Value

Optional. In most cases, leave this box empty.

Delimiter

Description

This is the delimiter character to be used in the DriverProperties prompt.

Required Value

The default is #

Description

Description

Enter a description for the database.

Required Value

A valid string.

DriverProperties

Description

If you choose to not to use the JDBC driver that is shipped with this eWay, you will need to add the drivers properties to the eWay. Often times the DataSource implementation will need to execute additional properties to assure a connection. The additional methods will need to be identified in the Driver Properties.

Required Value

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>##`".

If you are using Spy Log. Optional:

`setURL#jdbc:Seebeyond:sqlserver://<server>:50000;DatabaseName=<database>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##`".

Password

Description

Specifies the password used to access the database.

Required Values

Any valid string.

PortNumber

Description

Specifies the I/O port number on which the server is listening for connection requests.

Required Values

A valid port number. The default is 1521.

ServerName

Description

Specifies the host name of the external database server.

Required Values

Any valid string.

User

Description

Specifies the user name the eWay uses to connect to the database.

Required Values

Any valid string.

4.1.4. Setting the Properties in the Inbound eWay Environment

Figure 4 Inbound SQL Server eWay Environment



DatabaseName

Description

Specifies the name of the database instance.

Required Values

Any valid string.

Password

Description

Specifies the password used to access the database.

Required Values

Any valid string.

PortNumber

Description

Specifies the I/O port number on which the server is listening for connection requests.

Required Values

A valid port number. The default is 1433.

ServerName

Description

Specifies the host name of the external database server.

Required Values

Any valid string.

User

Description

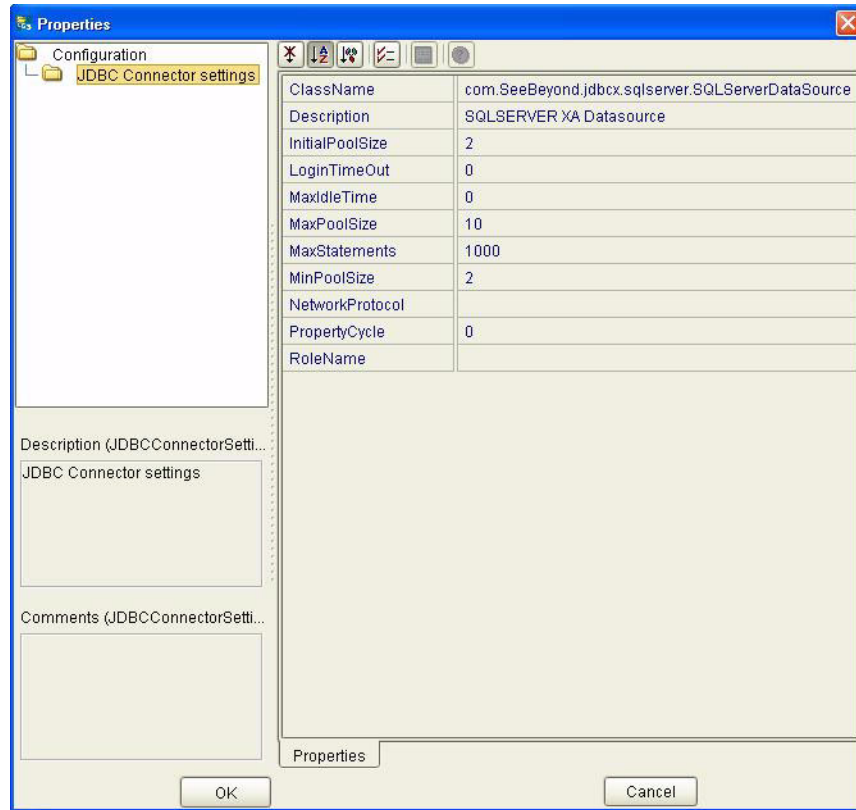
Specifies the user name the eWay uses to connect to the database.

Required Values

Any valid string.

4.1.5. Setting the Properties in the Outbound eWay with XA Support

Figure 5 Oubound SQL Server eWay with XA Support



ClassName

Description

Specifies the Java class in the JDBC driver that is used to implement the `ConnectionPoolDataSource` interface.

Required Values

A valid class name.

The default is `com.SeeBeyond.jdbcx.sqlserver.SQLServerDataSource`.

Description

Description

Enter a description for the database.

Required Value

A valid string.

InitialPoolSize

Description

Enter a number for the physical connections the pool should contain when it is created.

Required Value

A valid numeric value. The default is 2.

LoginTimeout

Description

The number of seconds driver will wait before attempting to log in to the database before timing out.

Required Value

A valid numeric value.

MaxIdleTime

Description

The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.

Required Value

A valid numeric value.

MaxPoolSize

Description

The maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum.

Required Value

A valid numeric value. The default is 10.

MaxStatements

Description

The maximum total number of statements that the pool should keep open. 0 (zero) indicates that the caching of statements is disabled.

Required Value

A valid numeric value. The default is 1000.

MinPoolSize

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.

Required Value

A valid numeric value.

NetworkProtocol

Description

The network protocol used to communicate with the server.

Required Values

Any valid string. The default is 2.

PropertyCycle

Description

The interval, in seconds, that the pool should wait before enforcing the current policy defined by the values of the other connection pool properties in this deployment descriptor.

Required Values

A valid numeric value. Default is 0.

RoleName

Description

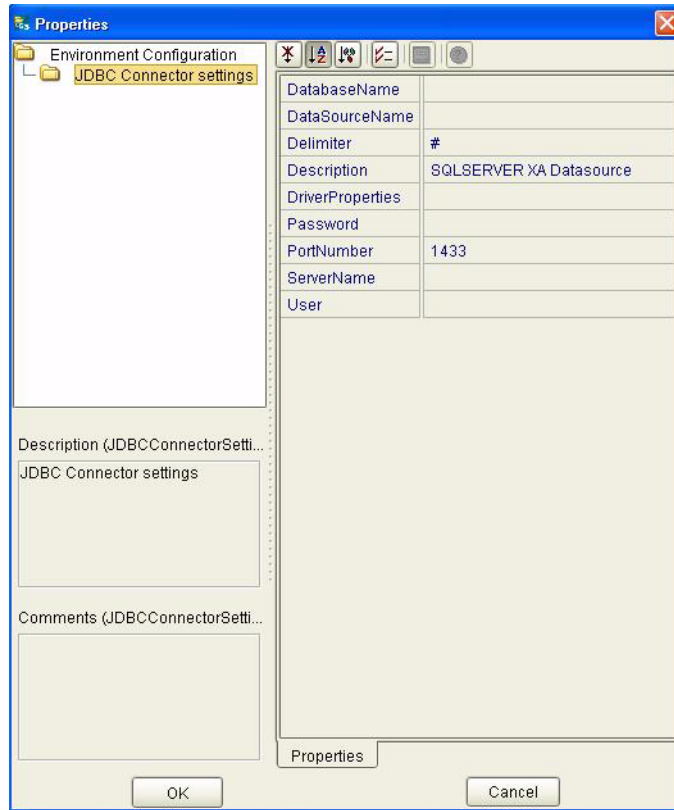
An initial SQL role name.

Required Values

Any valid string.

4.1.6. Setting the Properties in the Outbound eWay XA Environment

Figure 6 Environment Properties of the SQL Server eWay with XA



DatabaseName

Description

Specifies the name of the database instance.

Required Values

Any valid string.

DataSourceName

Description

Provide the name of the ConnectionPoolDataSource object that the DataSource object delegates behind the scenes when connection pooling or distributed transaction management is being done.

Required Value

Optional. In most cases, leave this box empty.

Delimiter

Description

This is the delimiter character to be used in the DriverProperties prompt.

Required Value

The default is #

Description

Description

Enter a description for the database.

Required Value

A valid string.

DriverProperties

Description

If you choose to not to use the JDBC driver that is shipped with this eWay, you will need to add the drivers properties to the eWay. Often times the DataSource implementation will need to execute additional methods to assure a connection. The additional methods will need to be identified in the Driver Properties.

Required Value

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>##`".

If you are using Spy Log. Optional:

```
"setURL#jdbc:SeeBeyond:sqlserver://<server>:446;locationName=<location>;collectionId=<collection>##setLocationName#<location>##setCollectionID#<collection>##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##"
```

Password

Description

Specifies the password used to access the database.

Required Values

Any valid string.

PortNumber

Description

Specifies the I/O port number on which the server is listening for connection requests.

Required Values

A valid port number. The default is 1521.

ServerName

Description

Specifies the host name of the external database server.

Required Values

Any valid string.

User

Description

Specifies the user name the eWay uses to connect to the database.

Required Values

Any valid string.

Using the SQL Server eWay Database Wizard

This chapter describes how to use the SQL Server eWay Database Wizard to build OTD's.

This Chapter Includes:

- [Select Wizard Type](#) on page 27
- [Connect to Database](#) on page 27
- [Select Database Objects](#) on page 28
- [Select Table/Views](#) on page 29
- [Select Procedures](#) on page 32
- [Add Prepared Statements](#) on page 33
- [Specify the OTD Name](#) on page 35

5.1 Using the Database OTD Wizard

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

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

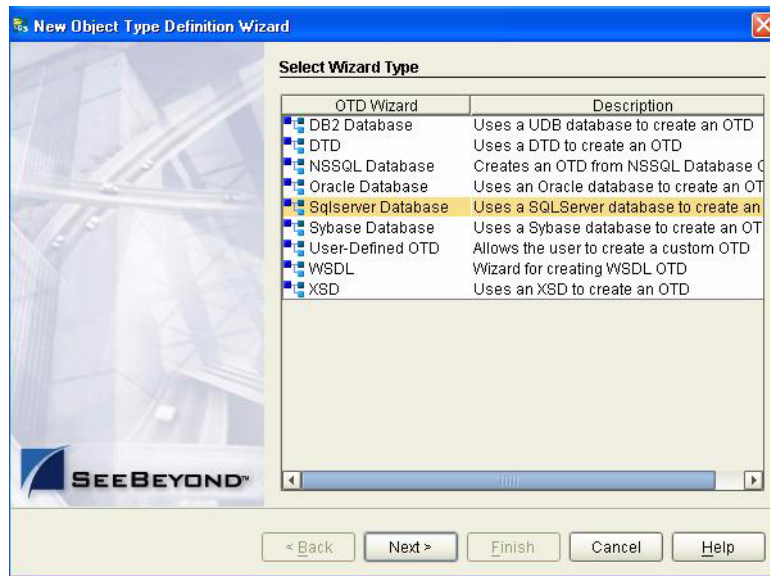
Note: *Database OTDs are not messagable. For more information on messagable OTDs, see the eGate Integrator User's Guide.*

To create a new OTD using the Database Wizard

Select Wizard Type

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

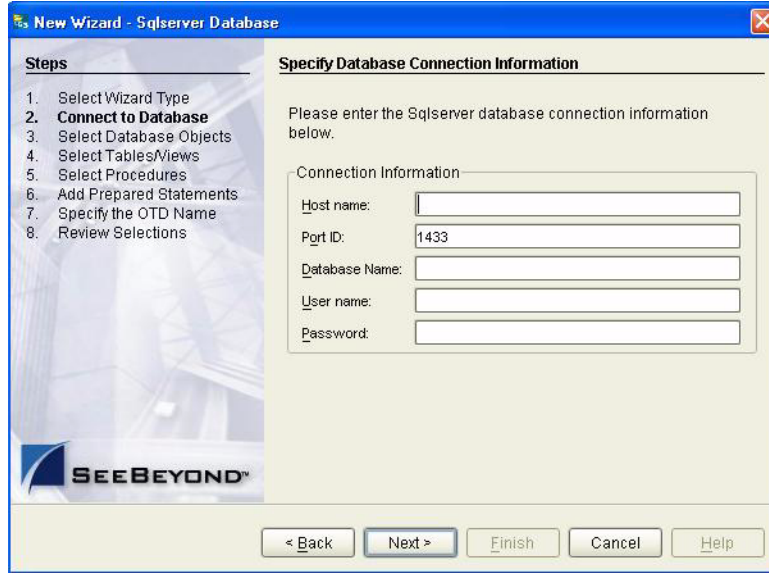
Figure 7 OTD Wizard Selection



Connect to Database

- 3 Specify the connection information for your database including your **UserName** and **Password** and click **Next**. See [Figure 8](#).

Figure 8 Database Connection Information

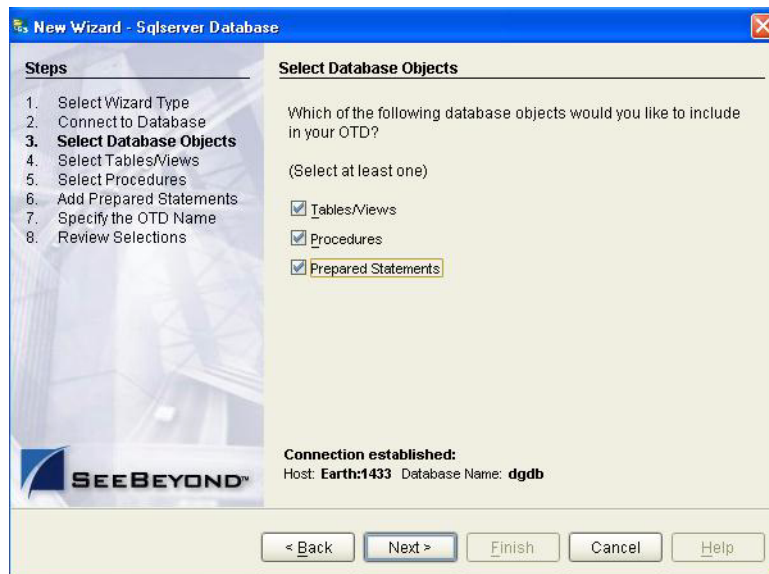


Select Database Objects

- 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 9](#).

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

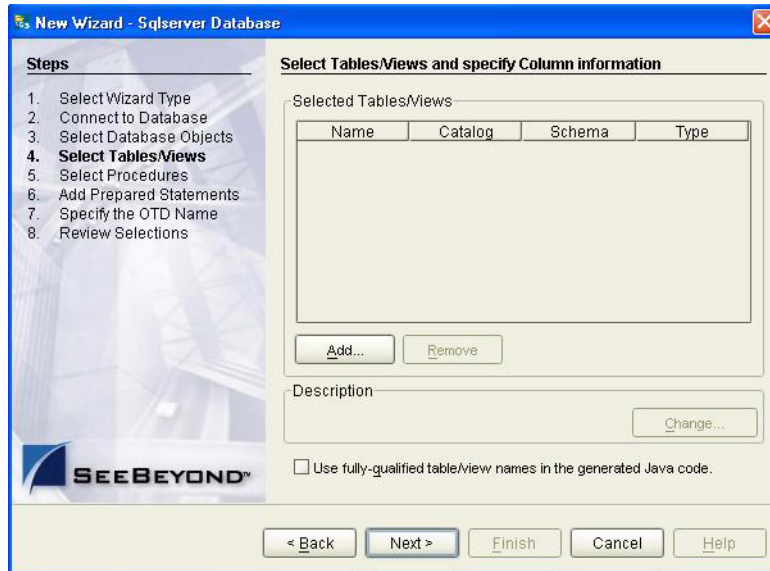
Figure 9 Select Database Objects



Select Table/Views

- 1 In the **Select Tables/Views** window, click **Add**. See [Figure 10](#).

Figure 10 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 11](#).

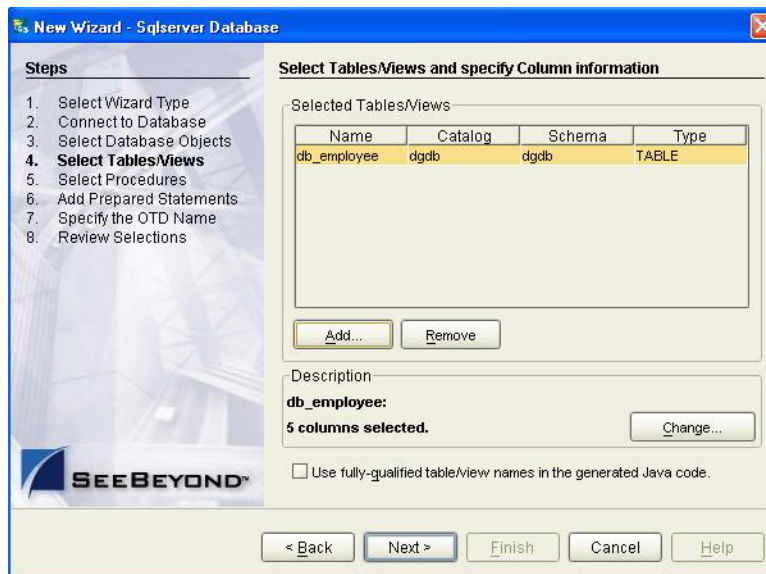
Figure 11 Database Wizard - All Schemes



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

The table selected is added to the **Selected Tables/Views** window. See [Figure 12](#).

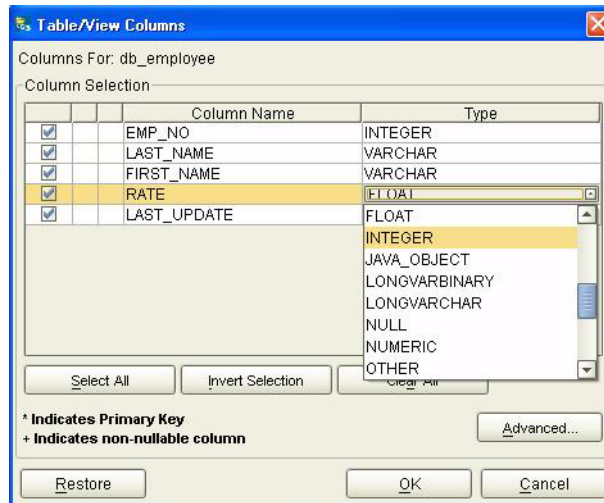
Figure 12 Selected Tables/Views window with a table selected



- 5 In the **Selected Tables/Views** window, review the table(s) you have selected. To make changes to the selected Table or View, click **Change**. If you do not wish to make any additional changes, click **Next** to continue.
- 6 In the **Table/View Columns** window, you can select or deselect your table columns. You can also change the data type for each table by highlighting the data type and

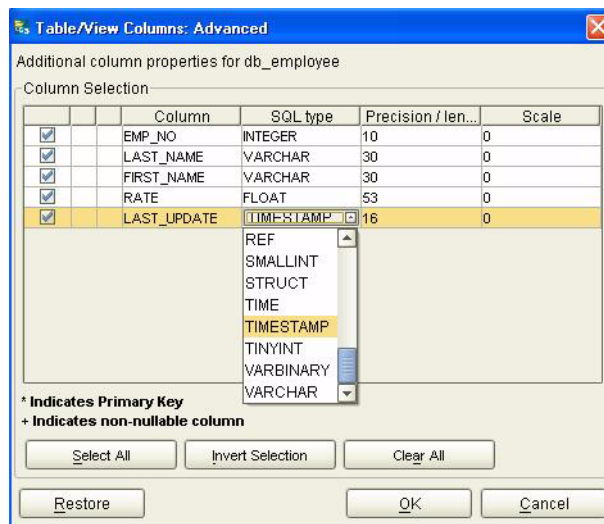
selecting a different one from the drop down list. If you would like to change any of the tables columns, click **Change**. See [Figure 13](#).

Figure 13 Table/View Columns



- 7 Click **Advanced** to change the data type, percision/length, or scale. Once you have finished your table choices, click **OK**. In general, you will not need to change these settings. See [Figure 14](#).

Figure 14 Table/View Columns – Advanced

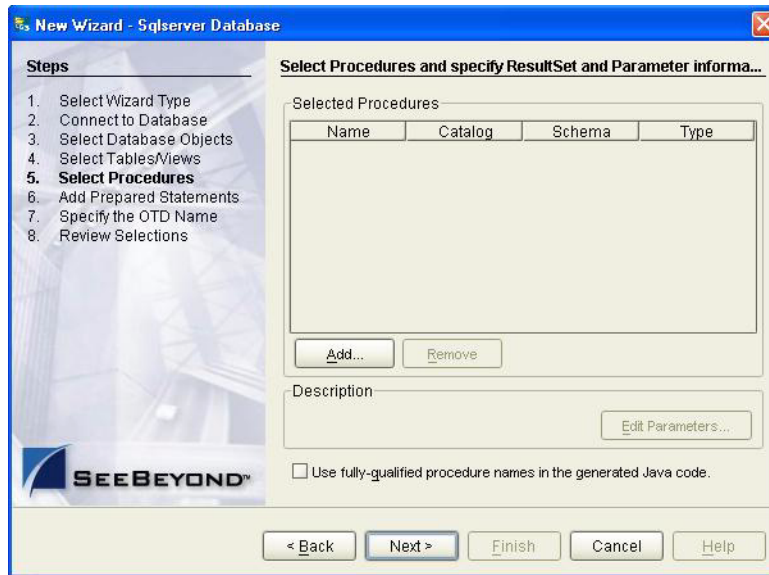


- 8 When using Prepared Statement packages, select **Use fully qualified table/view names in the generated Java code**. See [Figure 12](#).

Select Procedures

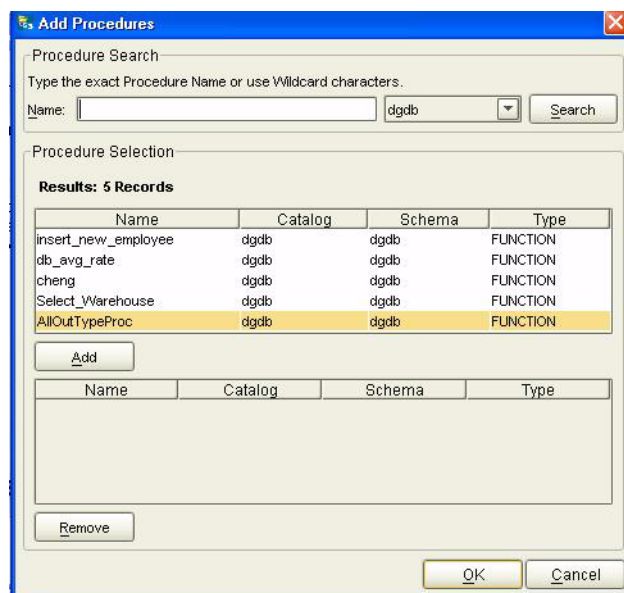
- 1 On the **Select Procedures and specify Resultset and Parameter Information** window, click **Add**.

Figure 15 Select Procedures and specify Resultset and Parameter Information



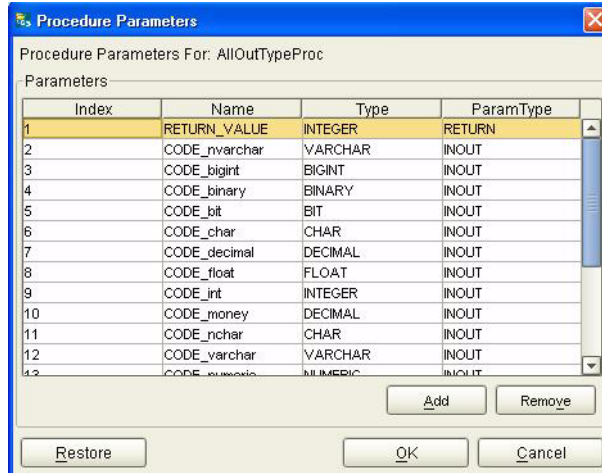
- 2 On the **Select Procedures** window, enter the name of a Procedure or select a table from the drop down list. Click **Search**. Wildcard characters can also be used.
- 3 In the resulting **Procedure Selection** list box, select a Procedure. Click **OK**.

Figure 16 Add Procedures



- On the **Select Procedures and specify Resultset and Parameter Information** window click **Edit Parameters** to make any changes to the selected Procedure. See **Figure 17**.

Figure 17 Procedure Parameters

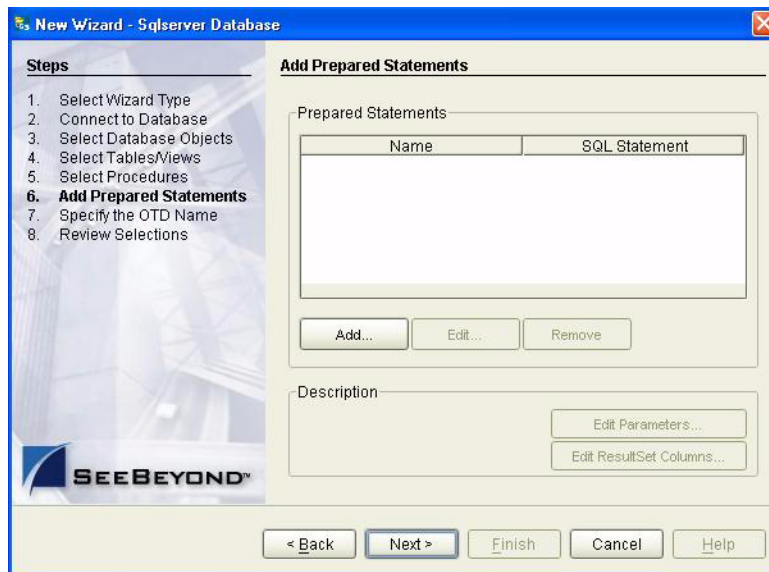


- To restore the data type, click **Restore**. When finished, click **OK**.
- On the **Select Procedures and specify Resultset and Parameter Information** window click **Next** to continue.

Add Prepared Statements

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

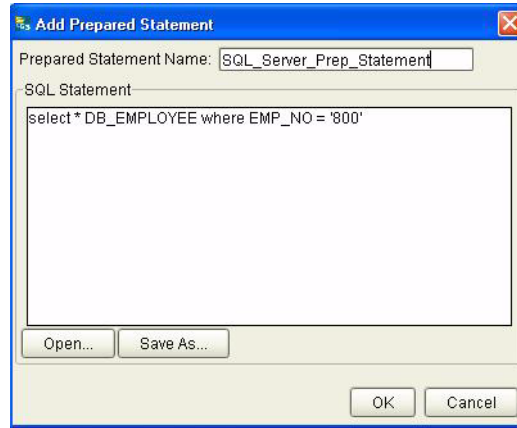
Figure 18 Prepared Statement



- Enter the name of a Prepared Statement or create a SQL statement by clicking in the SQL Statement window. When finished creating the statement, click **Save As** giving

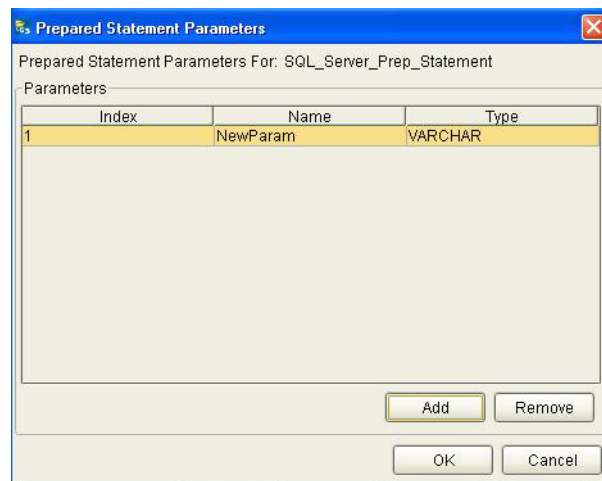
the statement a name. This name will appear as a node in the OTD. Click **OK**. See [Figure 19](#).

Figure 19 Prepared SQL Statement



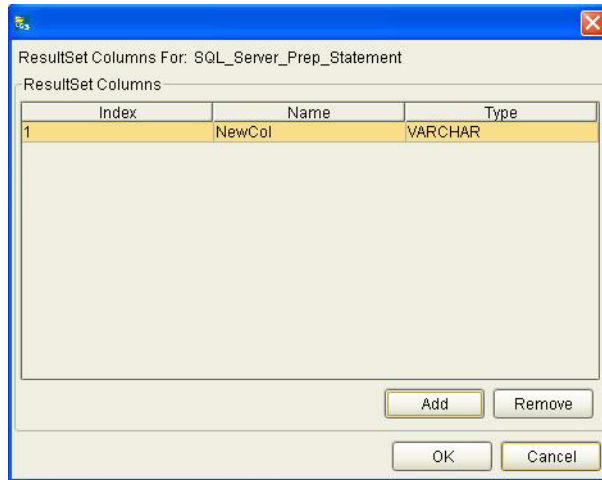
- 3 On the **Add Prepared Statement** window, the name you assigned to the Prepared Statement appears. To edit the parameters, click **Edit Parameters**. You can change the datatype by clicking in the **Type** field and selecting a different type from the list.
- 4 Click **Add** if you want to add additional parameters to the Statement or highlight a row and click **Remove** to remove it. Click **OK**. See [Figure 20](#).

Figure 20 Edit the Prepared Statement Parameters



- 1 To edit the Resultset Columns, click **Edit Resultset Columns**. Both the Name and Type are editable but it is recommend you do not changed the Name. Doing so will cause a loss of inegraty between the Resultset and the Database. Click **OK**. See [Figure 21](#).

Figure 21 ResultSet Columns



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 View the summary of the OTD. If you find you have made a mistake, click **Back** and correct the information. If you are satisfied with the OTD information, click **Finish** to begin generating the OTD. See [Figure 23](#).

Figure 23 Database Wizard - Summary



The resulting OTD will appear on the Enterprise Designer's canvas.

Working with the Sample Project(s)

This chapter describes how to build an SQL Server eWay project in a production environment.

This Chapter Includes:

- [eInsight Engine and eGate Components](#) on page 37
- [Using the Sample Project in eInsight](#) on page 37
- [Using the Sample Project in eGate](#) on page 49
- [Common DataType Conversions](#) on page 51
- [Using OTDs with Tables, Views, and Stored Procedures](#) on page 53
- [Alerting and Logging](#) on page 59

6.1 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:

- Java Messaging Service (JMS)
- 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 runs the Business Process, it automatically invokes that component via its Web Services interface.

6.2 Using the Sample Project in eInsight

To begin using the sample eInsight Business Process project, you will need to import the project and view it from within the Enterprise Designer using the Enterprise

Designer Project Import utility. Import the **SelectwithBPELDBEmployee.zip** file contained in the eWay sample folder on the installation CD-ROM.

Note: *eInsight is a Business Process modeling tool. If you have not purchased eInsight, contact your sales representative for information on how to do so.*

Before recreating the sample Business Process, review the *eInsight Business Process Manager User's Guide* and the *eGate Tutorial*.

Importing the Sample Project

1 On the Enterprise Designer's Project Explorer, right-click the Repository and select **Import**.

1 In the **Import Manager** window, browse to the directory that contains the sample Project zip files.

Sample Projects are contained within the zip file **SQL_Server_eWay_Sample.zip**, which is downloaded from the Repository to a folder of your choosing, (See **Installing the SQL Server eWay** on page 8). Once downloaded, unzip the file and extract the following sample Projects:

- ♦ SelectwithJBPELDBEmployee.zip
- ♦ SelectwithBPELDBEmployee.zip

2 Select a sample Project zip file and click **Import**.

3 After importing the file, click **OK** on the Import Status window. You can now import another zip file, or click **Close** to exit the Import Manager window.

The Business Process

The data used for this sample project is contained within a table called DBEmployee. The table has the following columns:

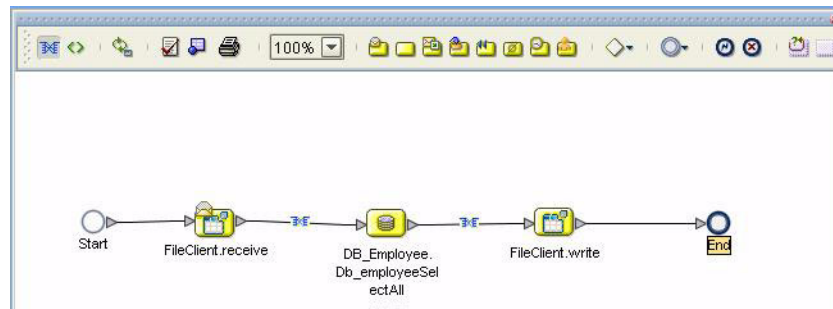
Table 1 Sample project data
Table 2

Column Name	Mapping	Data Type	Data Length
EMP_NO	employee_no	integer	10
LAST_NAME	employee_lname	varchar	30
FIRST_NAME	employee_fname	varchar	30
LAST_UPDATE	update_date	timestamp	16
RATE	rate	float	53

The sample project consists of an input file containing data that is passed into a database collaboration, and then written out to an output file

4 Refer to the *eInsight Business Process Manager User's Guide* for specific information on how to create and use a Business Process

Figure 24 Sample Project Business Process



You can associate an eInsight Business Process Activity with the eWay, both during the system design phase and during run time. To make this association, select the desired **receive** or **write** operation under the eWay in the Enterprise Explorer and drag it onto the eInsight Business Process canvas. The following operations are available:

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

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 SQL Server eWay. You can open a file specified in the eWay and view its contents before and after the Business Process is executed.

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

The table below shows the inputs and outputs to each of these eInsight operations:

eInsight Operation	Input	Output
SelectAll	where() clause (optional)	Returns all rows that fit the condition of the where() clause
SelectMultiple	number of rows where() clause (optional).	Returns the number of rows specified that fit the condition of the where() clause
SelectOne	where() clause (optional).	Returns the first row that fits the condition of the where() clause

eInsight Operation	Input	Output
Insert	definition of new item to be inserted	Returns status.
Update	where() clause	Returns status.
Delete	where() clause	Returns status.

6.2.1 whereClause()

A BPEL whereClause() statement may be joined by AND/OR with conditions of "=", "!=", "<>", "<", ">", "<=", ">=".

For example:

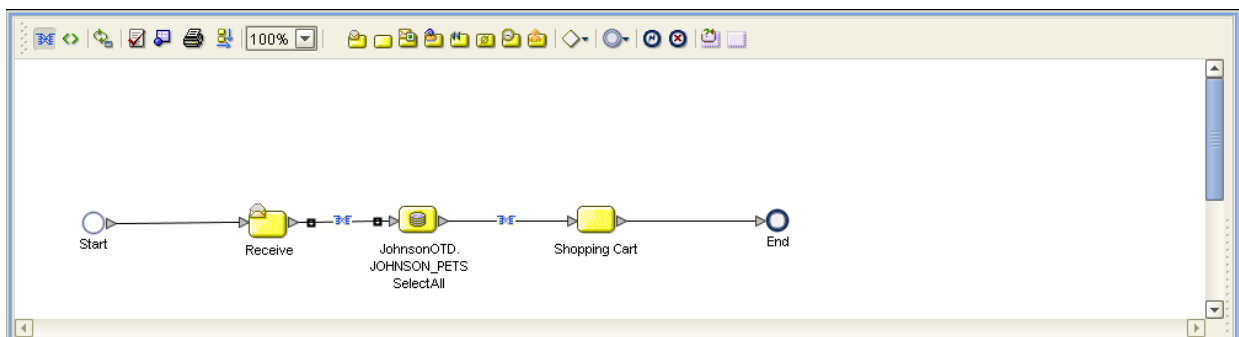
```
whereClause such as where column2=2 AND column1=1 OR column3=3 is valid
```

6.2.2 SelectAll

The input to a SelectAll operation is an optional where() clause. The where() clause defines to which criteria rows must adhere to be returned. In the SelectAll operation, all items that fit the criteria are returned. If the where() clause is not specified, all rows are returned.

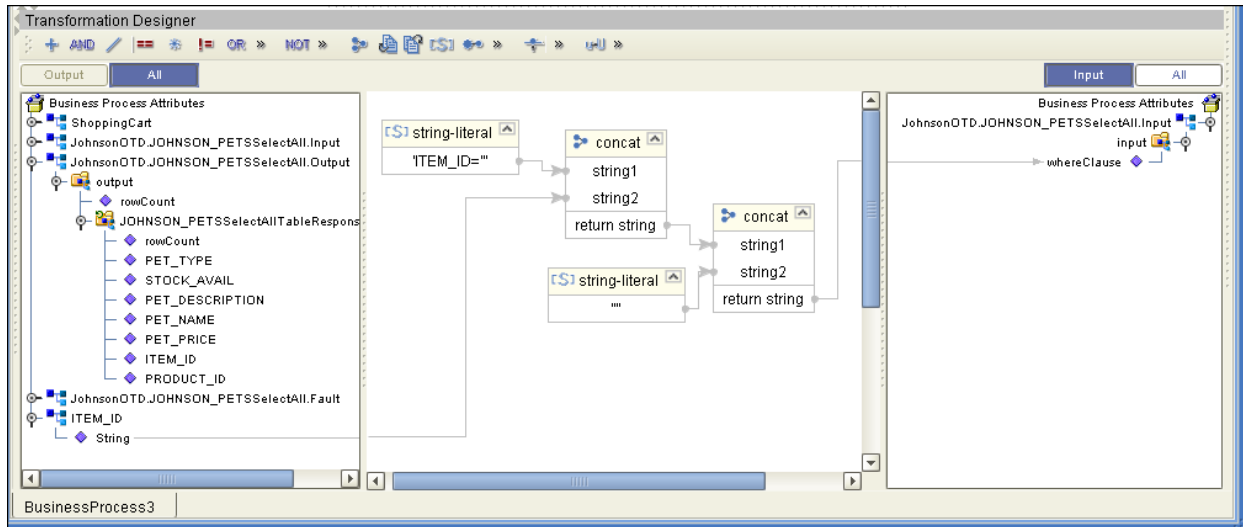
The figure below shows a sample eInsight Business Process using the SelectAll operation. In this process, the SelectAll operation returns all rows where the ITEM_ID matches the selected ITEM_ID to the shopping cart.

Figure 25 SelectAll Sample Business Process



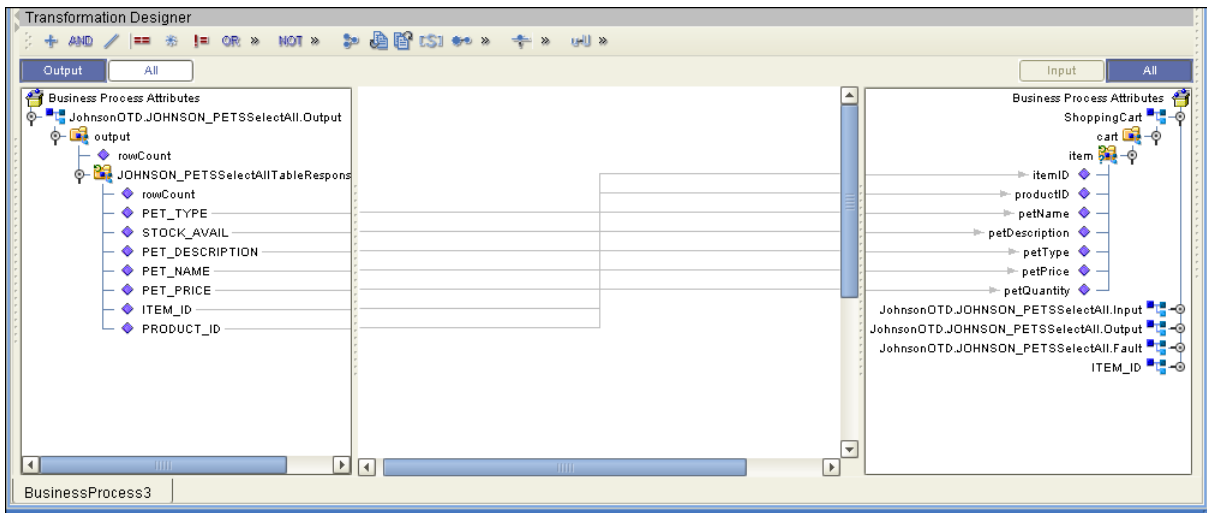
The figure below shows the definition of the where() clause for the SelectAll operation.

Figure 26 SelectAll Input



The figure below shows the definition of the output for the SelectAll operation. For each row selected during the operation, the shopping cart shows the columns of those rows as defined here.

Figure 27 SelectAll Output

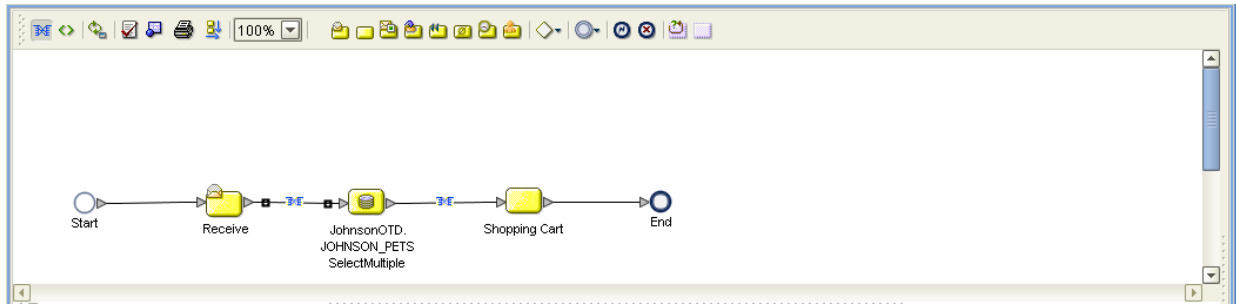


6.2.3 SelectMultiple

The input to a SelectMultiple operation is the number of rows to be selected and a where() clause. The number of rows indicates how many rows the SelectMultiple operation returns. The where() clause defines to which criteria rows must adhere to be returned.

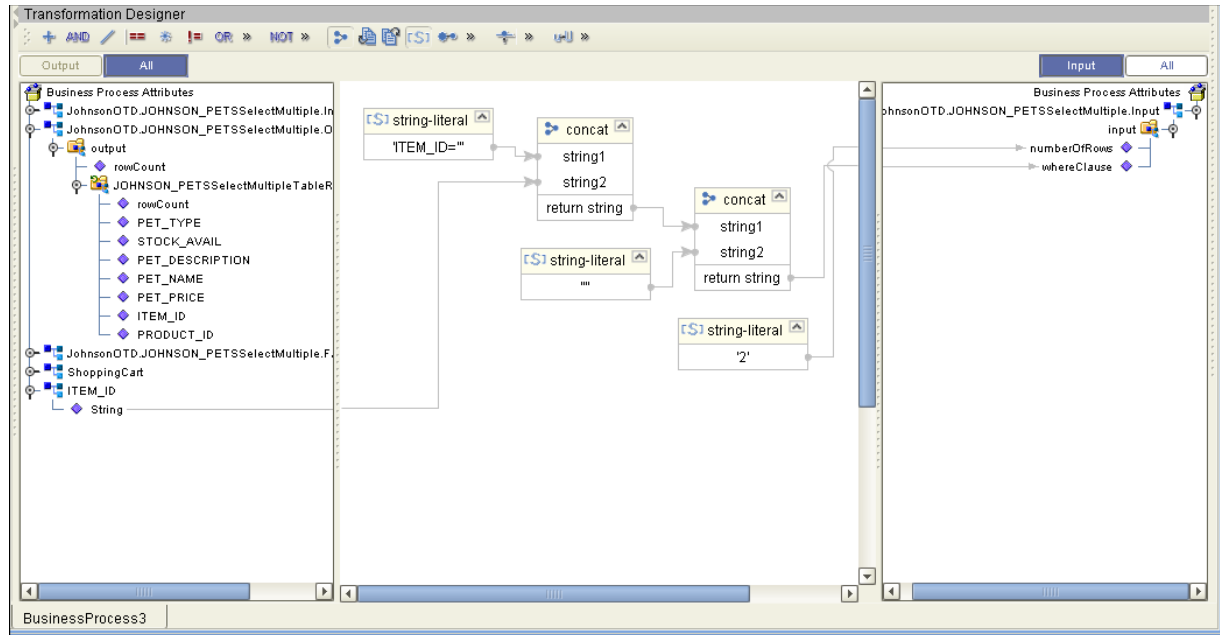
The figure below shows a sample eInsight Business Process using the SelectMultiple operation. In this process, the SelectMultiple operation returns the first two rows where the ITEM_ID matches the selected ITEM_ID to the shopping cart.

Figure 28 SelectMultiple Sample Business Process



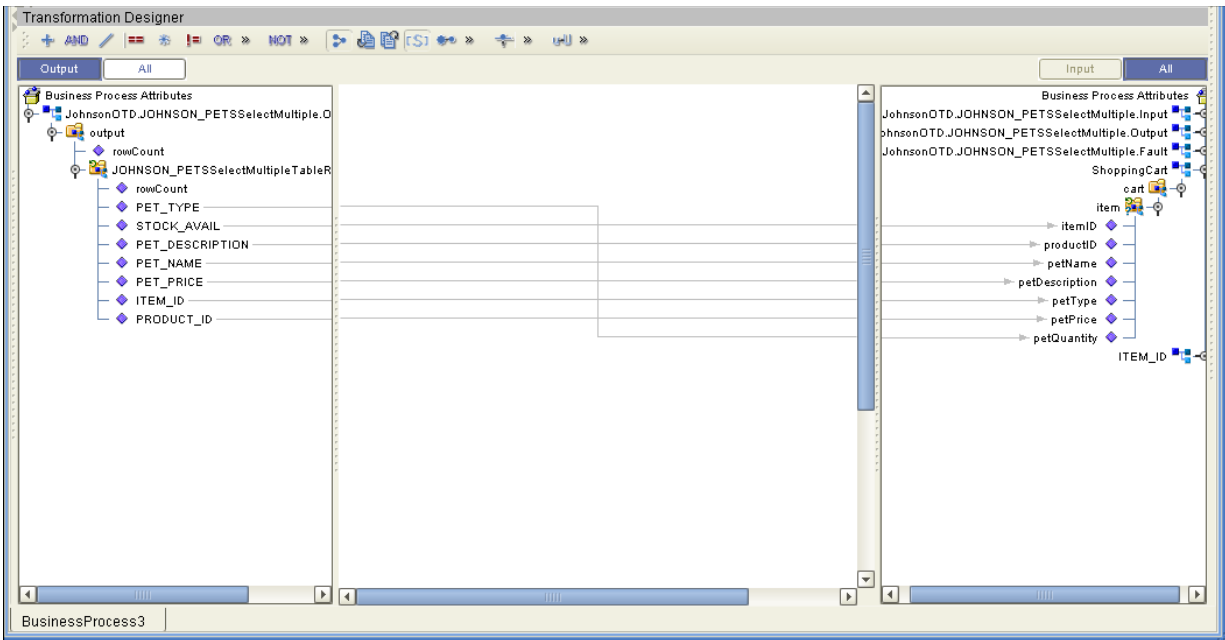
The figure below shows the definition of the number of rows and where() clause clause into the input for the SelectMultiple operation. You could also use an empty string or Item_ID='123'..

Figure 29 SelectMultiple Input



The figure below shows the definition of the output for the SelectMultiple operation. For each row selected during the operation, the shopping cart shows the columns of those rows as defined here.

Figure 30 SelectMultiple Output

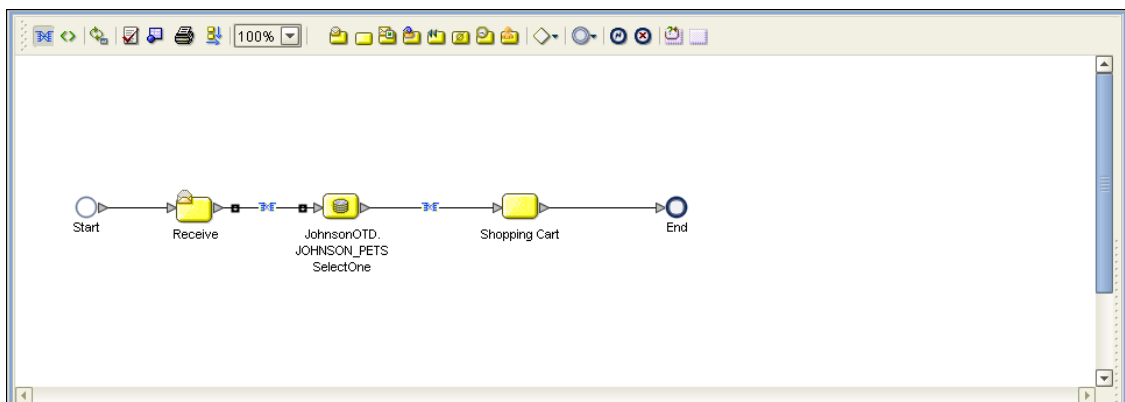


6.2.4 SelectOne

The input to a SelectOne operation is a where() clause. The where() clause defines to which criteria rows must adhere to be selected for the operation. In the SelectOne operation, the first row that fits the criteria is returned.

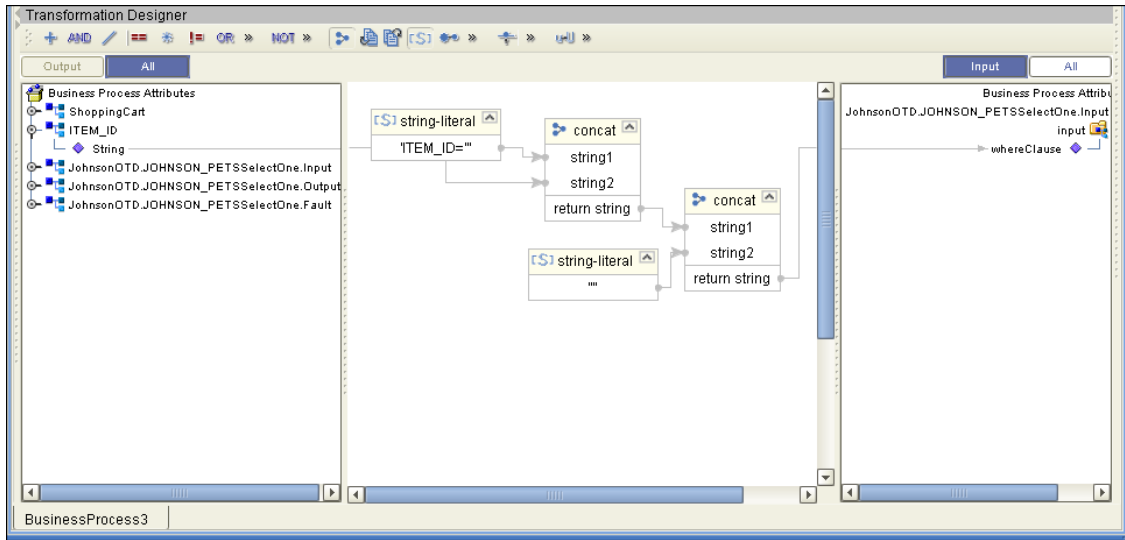
The figure below shows a sample eInsight Business Process using the SelectOne operation. In this process, the SelectOne operation returns the first row where the ITEM_ID matches the specified ITEM_ID to the shopping cart.

Figure 31 SelectOne Sample Business Process



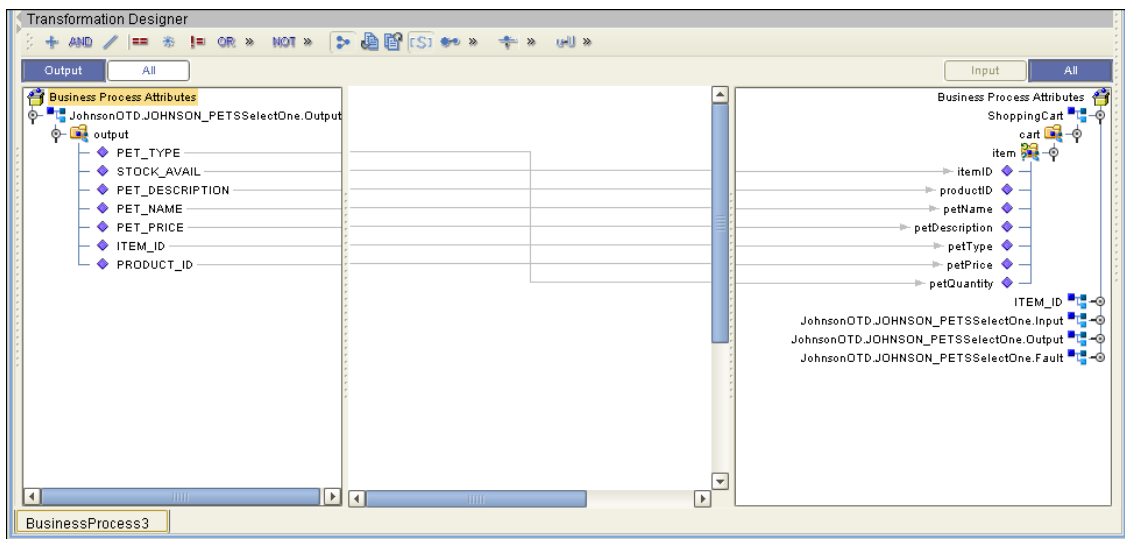
The figure below shows the definition of the where() clause for the SelectOne operation.

Figure 32 SelectOne Input



The figure below shows the definition of the output for the SelectOne operation. For the first row selected during the operation, the shopping cart shows the columns of that row as defined here.

Figure 33 SelectOne Output

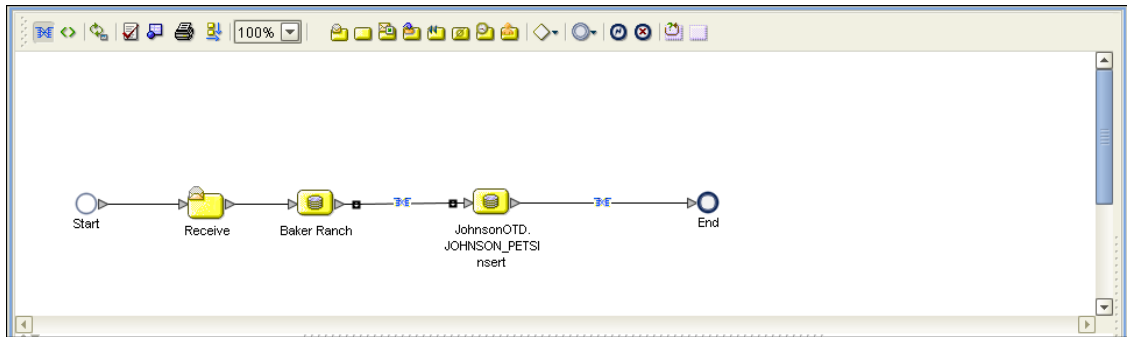


6.2.5 Insert

The Insert operation inserts a row. The input to an Insert operation is a where() clause. The where() clause defines to which criteria rows must adhere to be selected for the operation. In the Insert operation, the first row that fits the criteria is returned.

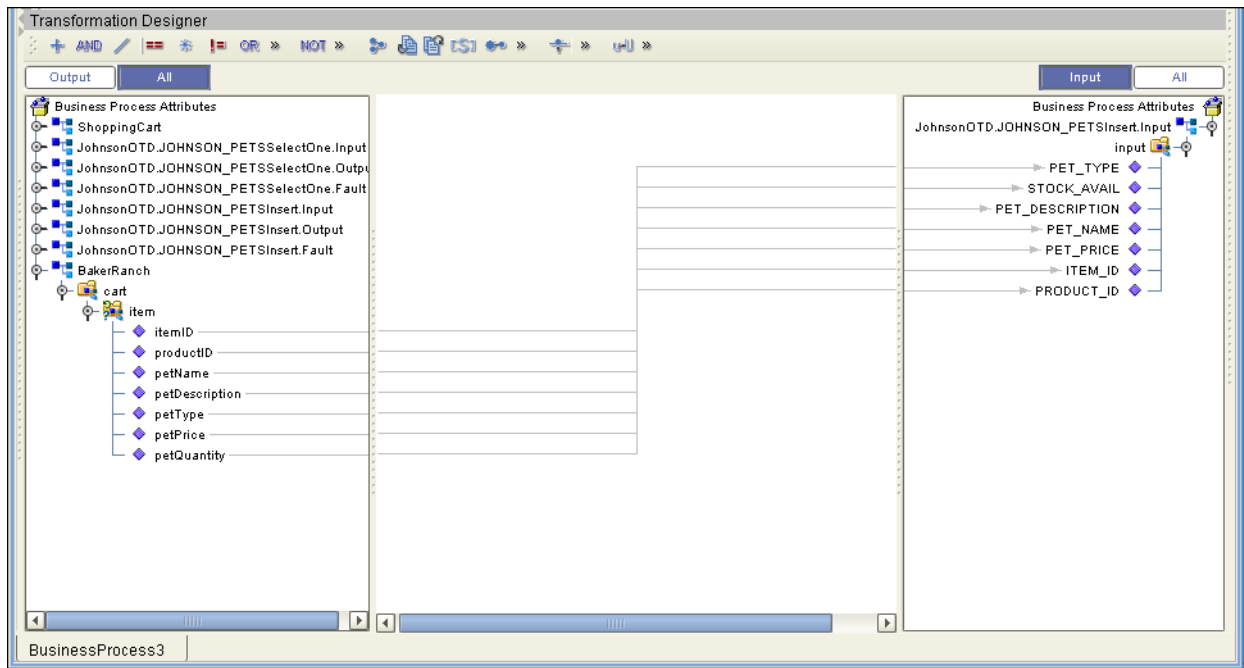
The figure below shows a sample eInsight Business Process using the Insert operation. In this process, the operation inserts a new row into the database to accommodate a new item provided by a vendor.

Figure 34 Insert Sample Business Process



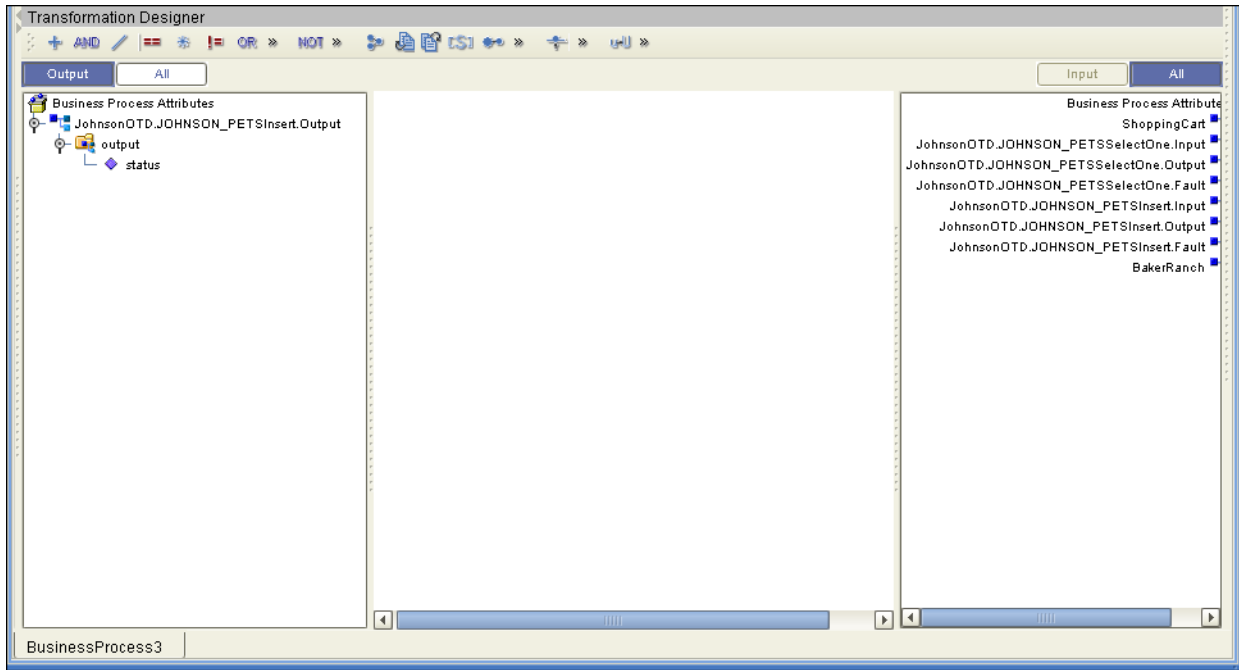
The figure below shows the definition of the input for the Insert operation.

Figure 35 Insert Input



The figure below shows the output of the Insert operation, which is a status indicating the number of rows created.

Figure 36 Insert Output

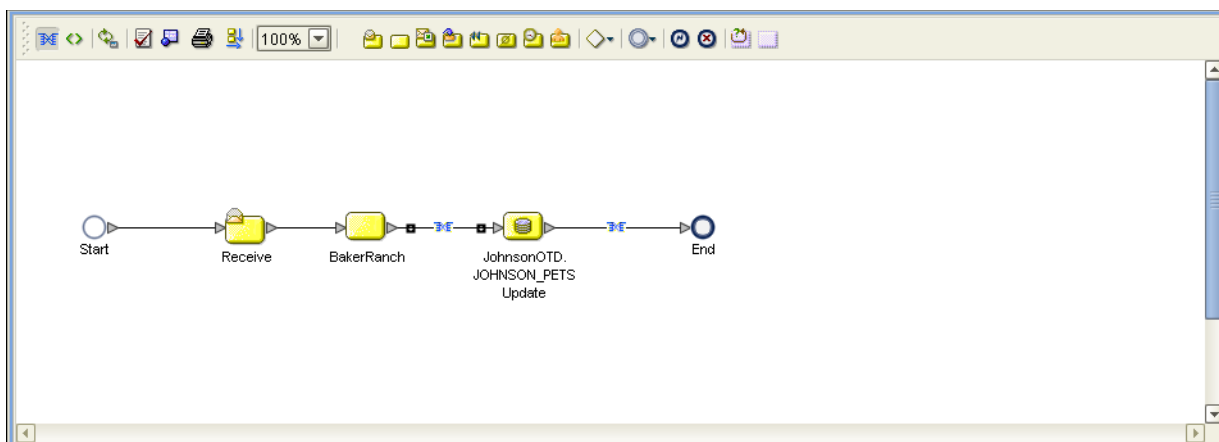


6.2.6 Update

The Update operation updates rows that fit certain criteria defined in a where() clause.

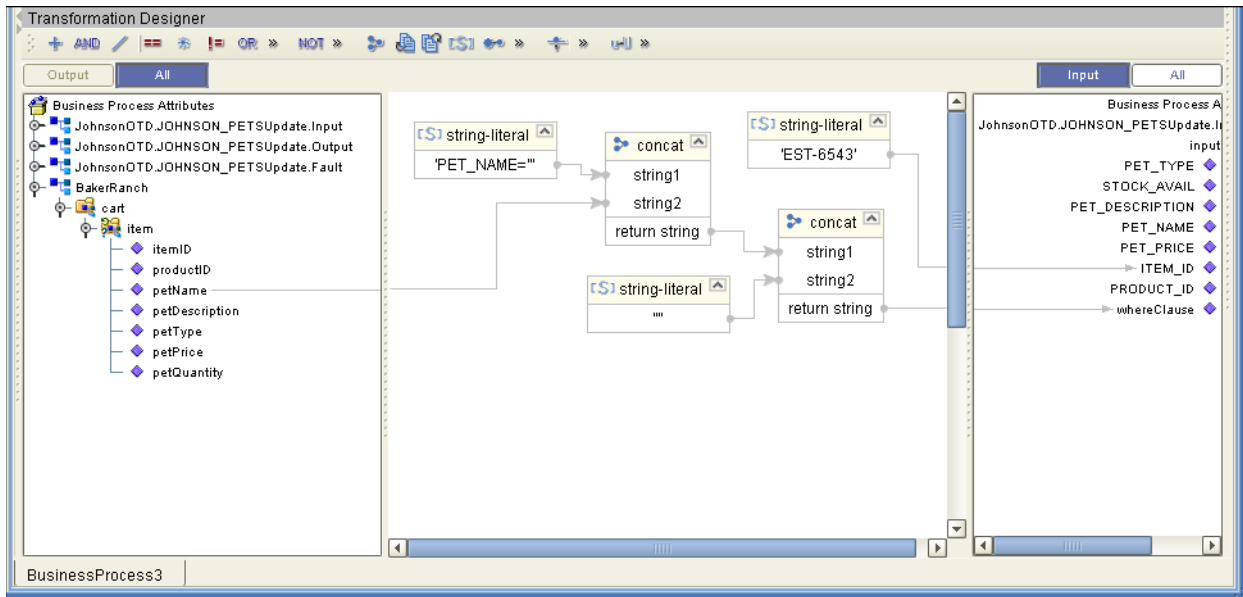
The figure below shows a sample eInsight Business Process using the Update operation. In this process, the operation updates the ITEM_ID for all items with a certain name to ESR_6543.

Figure 37 Update Sample Business Process



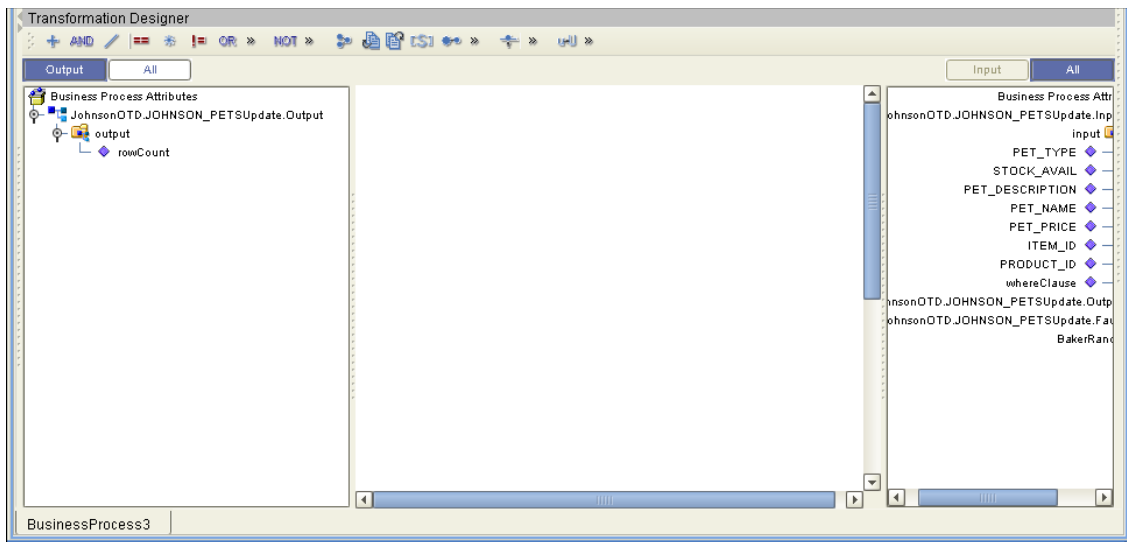
The figure below shows the definition of the where() clause for the Update operation.

Figure 38 Update Input



The figure below shows the output of the Update operation, which is a status indicating the number of rows updated.

Figure 39 Update Output

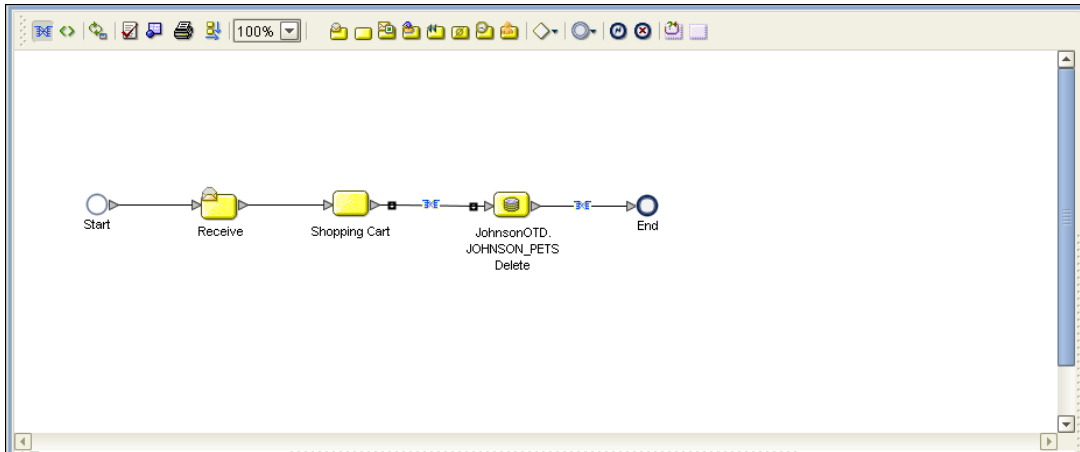


6.2.7 Delete

The Delete operation deletes rows that match the criteria defined in a where() clause. The output is a status of how many rows were deleted.

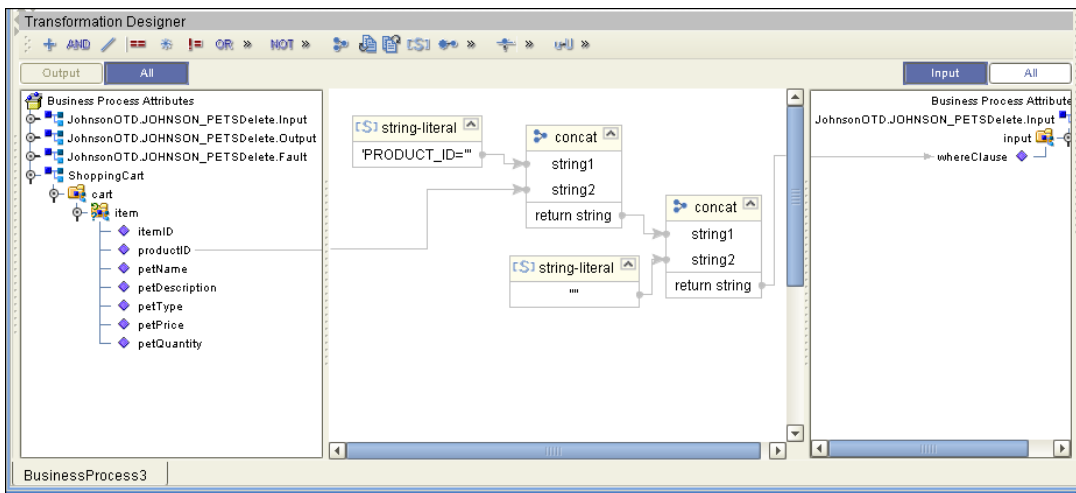
The figure below shows a sample eInsight Business Process using the Delete operation. In this process, the operation deletes rows with a certain product ID from the shopping cart.

Figure 40 Delete Sample Business Process



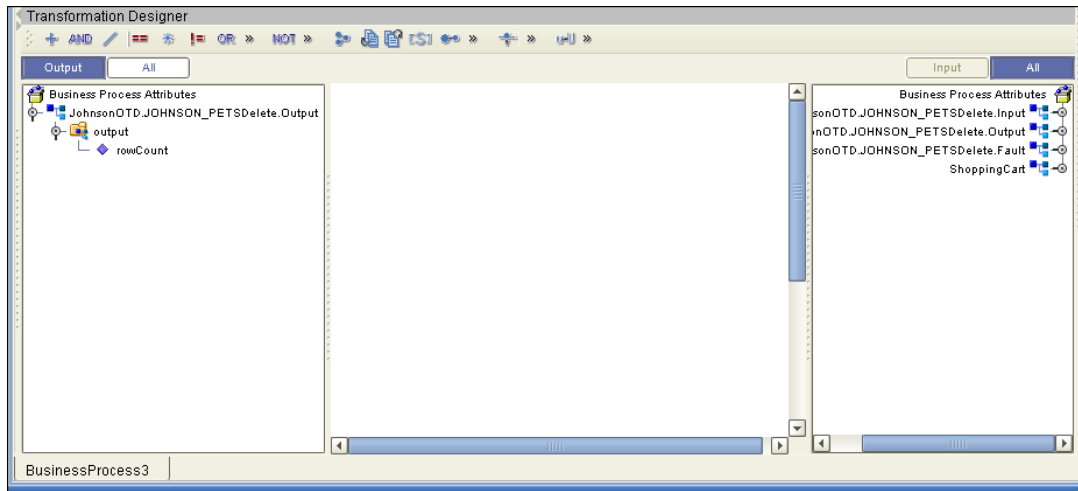
The figure below shows the definition of the where() clause for the Delete operation.

Figure 41 Delete Input



The figure below shows the output of the Delete operation, which is a status indicating the number of rows deleted.

Figure 42 Delete Output



6.3 Using the Sample Project in eGate

To import the sample project **JDEDBEMPLOYEE.zip** follow the instructions given in [Importing the Sample Project](#) on page 38.

6.3.1. Working with the Sample Project in eGate

This sample project selects the EMP_NO, LAST_NAME, FIRST_NAME, SS_NUMBER, and the HIRE_DATE columns from the table DBEmployee and publishes the record to an output file.

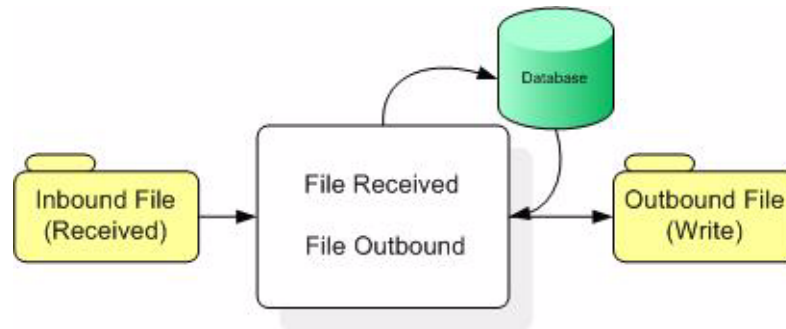
The data used for this projects is within a table called DBEmployee. The table contains the following columns:

Table 3 Sample project data
Table 4

Column Name	Mapping	Data Type	Data Length
EMP_NO	Empno	varchar2	10
LAST_NAME	Lastname	varchar2	30
FIRST_NAME	Firstname	varchar2	30
SS_NUMBER	SSnumber	varchar2	20
HIRE_DATE	HireDate	varchar2	12

The sample project consists of an input file containing data that is passed into a collaboration and out to the database from which data is retrieved and passed back into the collaboration and then to an output file.

Figure 43 Database project flow



To work with the sample project, follow the instructions given in the *eGate Tutorial*.

Note: *Outbound database eWays are available when using a JCE Collaboration. To poll the database, you must use the Scheduler.*

6.3.2. Configuring the eWays

The sample uses an inbound and an outbound File eWay as well as an outbound SQL Server eWay. To configure the sample projects eWays, use the following information. For additional information on the SQL Server properties, see [Setting the eWay Properties in the Connectivity Map](#) on page 10.

To configure the Inbound File eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between the **File1** and **Service1**.
- 2 On the resulting **Templates** window, select **Inbound File eWay** and click **OK**.
- 3 On the **Properties** window, enter the appropriate configurations for the Inbound File eWay. See the *File eWay User's Guide* for information on how to specifically configure the File eWay. For this sample, the default settings are used.
- 4 When you have completed your selections, click **OK**.

To configure the Outbound SQLServer1 eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between the **Service1** and **SQLServer1** database.
- 2 On the resulting **Templates** window, select **Outbound SQLServer1** and click **OK**.
- 3 On the Properties window, enter the appropriate configurations for the Outbound SQL Server eWay and click **OK**. See [Setting the Properties in the Outbound eWay](#) on page 10. For this sample, the default settings are used.
- 4 When you have completed your selections, click **OK**.

To configure the Outbound File eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between **Service1** and **File2** eWay.

- 2 On the resulting **Templates** window, select **Outbound File eWay** and click **OK**.
- 3 On the **Properties** window, enter the appropriate configurations for the Outbound File eWay. See the *File eWay User's Guide* for information on how to specifically configure the File eWay. For this sample, change the Directory field to **<valid path to the directory where the output file will be stored>**. The Output File Name to **Output1.dat**. For the remaining parameters, the default settings are used.
- 4 When you have completed your selections, click **OK**.

6.3.3. Creating the External Environment

To review the components of the Sample project, there is an Inbound and an Outbound File eWay, an eWay, and a Service.

To create the external environment for the Sample project:

- 1 On the Environment Explorer, highlight and right-click the SQL Server profile. Select **Properties**. Enter the configuration information required for your Outbound SQL Server eWay. See [Setting the Properties in the Outbound eWay Environment](#) on page 15.

6.3.4 Deploying a Project

To deploy a project, please see the *"eGate Integrators User's Guide"*.

6.3.5. Running the Sample

For instruction on how to run a Sample project, see the *eGate Tutorial*.

Once the process has completed, the Output file in the target directory configured in the Outbound File eWay will contain all records retrieved from the database in an .xml format.

6.4 Common Data Type Conversions

Figure 44 The SQL Server eWay Datatype Conversions

SQL Server Data Type	OTD/Java Data Type	Methods to Use	Sample Data
BigInt	Long	Long: <code>java.lang.Long.parseLong(String)</code>	123
Int	Int	Integer: <code>java.lang.Integer.parseInt(String)</code>	123

SQL Server Data Type	OTD/Java Data Type	Methods to Use	Sample Data
tinyInt	Byte	Byte: java.lang.Byte.parseByte(String)	123
SmallInt	Short	Short: java.lang.Short.parseShort(String)	123
Number	BigDecimal	Call a NewConstructor BigDecimal: java.math.BigDecimal(String)	145.78
Decimal	BigDecimal	Call a NewConstructor BigDecimal: java.math.BigDecimal(String)	145.78
Bit	Boolean	Boolean: java.lang.Boolean.getBoolean(String)	0 or 1
Real	Float	Float: java.lang.Float.parseFloat(String)	3468.494
Float	Double	Double: java.lang.Double.parseDouble(String)	3468.494
Money	BigDecimal	Call a NewConstructor BigDecimal: java.math.BigDecimal(String)	2456.95
Smallmoney	BigDecimal	Call a NewConstructor BigDecimal: java.math.BigDecimal(String)	2456.95
Smalldatetime	TimeStamp	TimeStamp: java.sql.TimeStamp.valueOf(String)	2003-09-28 11:35:00
Timestamp	Binary	N/A (Used by the Database Internally)	N/A
DateTime	TimeStamp	Date: java.sql.Date.valueOf(String)	2003-09-28 11:35:42
Varchar	String	Direct Assign	Any Characters
Char	String	Direct Assign	Any Characters
Text	String	Direct Assign	Any Characters
Binary(1)	Byte[]	String: java.lang.String.getBytes()	0 or 1

6.5 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.

6.5.1 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
    employeeedb_db_employee_1.setEmployee_no(
java.lang.Integer.toString(
db_employee_1.getDb_employee().getEMP_NO() ) );
//@map:Copy LAST_NAME to Employee_lname
    employeeedb_db_employee_1.setEmployee_lname(
db_employee_1.getDb_employee().getLAST_NAME() );
//@map:Copy FIRST_NAME to Employee_fname
    employeeedb_db_employee_1.setEmployee_fname(
db_employee_1.getDb_employee().getFIRST_NAME() );
//@map:Copy RATE to Rate
    employeeedb_db_employee_1.setRate( java.lang.Double.toString(
db_employee_1.getDb_employee().getRATE() ) );
//@map:Copy LAST_UPDATE to Update_date
    employeeedb_db_employee_1.setUpdate_date(
db_employee_1.getDb_employee().getLAST_UPDATE().toString() );
}
//@map:Copy employeeedb_db_employee_1.marshallToString to Text
    FileClient_1.setText(
    employeeedb_db_employee_1.marshallToString() );
//@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(
    employeeedb_with_top_db_employee_1.getEmployee_no() ) );

//@map:Copy Employee_lname to Employee_Lname
    insert_DB_1.getInsert_new_employee().setEmployee_Lname(
    employeeedb_with_top_db_employee_1.getEmployee_lname() );

//@map:Copy Employee_fname to Employee_Fname
    insert_DB_1.getInsert_new_employee().setEmployee_Fname(
    employeeedb_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(
    employeeedb_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(
    employeeedb_with_top_db_employee_1.getUpdate_date() ) );
    Table_OTD_1.getDB_EMPLOYEE().insertRow();

//Table_OTD_1.commit
    Table_OTD_1.commit();

```

```
}
```

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();
}
//DB_sales_orders_1.commit
DB_sales_orders_1.commit();

//Copy "Update completed" to Text
FileClient_1.setText( "Update completed" );

//FileClient_1.write
FileClient_1.write();
}
```

The Delete Operation

To perform a delete operation on a table

- 1 Execute the **delete()** method.
In this example DELETE an employee.

```
//DB_EMPLOYEE.delete("EMP_NO = '".concat(EMP_NO).concat("'")
Table_OTD_1.getDB_EMPLOYEE().delete( "EMP_NO = '".concat(
employeeedb_with_top_db_employee_1.getEMP_NO() ).concat( "'") );
}
```

6.5.2 The Stored Procedure

A Stored Procedure OTD represents a database stored procedure. Fields correspond to the arguments of a stored procedure while methods are the operations that you can apply to the OTD. It allows you to execute a stored procedure. Remember that while in the Collaboration Editor you can drag and drop nodes from the OTD into the Collaboration Editor.

Executing Stored Procedures

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

Below are the steps for executing the Stored Procedure:

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

For example:

```
package Storedprocedure;

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

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

    public void receive(
com.stc.connector.appconn.file.FileTextMessage
input, com.stc.connector.appconn.file.FileApplication
FileClient_1, employeeDb.Db_employee
employeeDb_with_top_db_employee_1, insert_DB.Insert_DBOTD insert_DB_1
)
    throws Throwable
    {
        //
        @map:employeeDb_with_top_db_employee_1.unmarshalFromString(Text)
            employeeDb_with_top_db_employee_1.unmarshalFromString(
input.getText() );

        // @map:Copy java.lang.Integer.parseInt(Employee_no) to
Employee_no
            insert_DB_1.getInsert_new_employee().setEmployee_no(
java.lang.Integer.parseInt(
employeeDb_with_top_db_employee_1.getEmployee_no() ) );

        // @map:Copy Employee_lname to Employee_Lname
            insert_DB_1.getInsert_new_employee().setEmployee_Lname(
employeeDb_with_top_db_employee_1.getEmployee_lname() );

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



```
        insert_DB_1.getInsert_new_employee().setEmployee_Fname(
employeeedb_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(
employeeedb_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(
employeeedb_with_top_db_employee_1.getUpdate_date() ) );

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

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

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

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

Manipulating the ResultSet and Update Count Returned by Stored Procedure

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

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

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

The `resultsAvailable()` method, added to the 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.

6.6 Alerting and Logging

eGate provides an alerting and logging feature. This allows monitoring of messages and captures any adverse messages in order of severity based on configured severity level and higher. To enable Logging, please see the *eGate Integrator User's Guide*.

Index

A

Add Prepared Statements 33

C

ClassName 11, 20
 Configuring eWay Connections 10
 Connect to Database 27

D

Database Wizard 27
 DatabaseName 15, 18, 23
 DataSourceName 15, 23
 Delimiter 16, 24
 Description 11, 16, 20, 24
 driver class, JDBC 11, 20
 DriverProperties 16, 24

E

eWay properties 10
 External System Requirements 7

I

Inbound Environment Properties
 Database 18
 Password 18
 PortNumber 19
 ServerName 19
 User 19
 Inbound Properties
 Pollmilliseconds 14
 PreparedStatement 14
 InitialPoolSize 12, 21

J

JDBC
 driver class 11, 20

L

LoginTimeOut 12, 21

M

MaxIdleTime 12, 21
 MaxPoolSize 12, 21
 MaxStatements 12, 21
 MinPoolSize 13, 22

N

NetworkProtocol 13, 22

O

Outbound Environment Properties
 DatabaseName 15
 DataSourceName 15
 Delimiter 16
 Description 16
 DriverProperties 16
 Password 16
 PortNumber 16
 ServerName 17
 User 17
 Outbound Properties
 ClassName 11
 Description 11, 20
 InitialPoolSize 12, 21
 LoginTimeOut 12, 21
 MaxIdleTime 12, 21
 MaxPoolSize 12, 21
 MaxStatements 12, 21
 MinPoolSize 13, 22
 NetworkProtocol 13, 22
 PropertyCycle 13
 RoleName 13
 Outbound XA Environment Properties
 DatabaseName 23
 DataSourceName 23
 Delimiter 24
 Description 24
 DriverProperties 24
 Password 24
 PortNumber 25
 ServerName 25
 User 25
 Outbound XA Properties
 ClassName 20
 Description 20
 InitialPoolSize 21
 LoginTimeOut 21

Index

- MaxIdle 21
- MaxPoolSize 21
- MaxStatements 21
- MinPoolSize 22
- NetworkProtocol 22
- PropertyCycle 22
- RoleName 22
- Outbound XA Property settings, Outbound
 - ClassName 20
 - Description 20
 - InitialPoolSize 21
 - LoginTimeOut 21
 - MaxIdle 21
 - MaxPoolSize 21
 - MaxStatements 21
 - MinPoolSize 22
 - NetworkProtocol 22
- Outbound XA Property settings, Outbound Environment
 - DatabaseName 23
 - DataSourceName 23
 - Delimiter 24
 - Description 24
 - DriverProperties 24
 - Password 24
 - PortNumber 25
 - ServerName 25
 - User 25
- P**
 - Password 16, 18, 24
 - Pollmilliseconds 14
 - PortNumber 16, 19, 25
 - PreparedStatement 14
 - Properties of the eWay
 - creating 10
 - Property settings, Inbound
 - Pollmilliseconds 14
 - PreparedStatement 14
 - Property settings, Inbound Environment
 - Database 18
 - Password 18
 - PortNumber 19
 - ServerName 19
 - User 19
 - Property settings, Outbound
 - ClassName 11
 - Description 11, 20
 - InitialPoolSize 12, 21
 - LoginTimeOut 12, 21
 - MaxIdleTime 12, 21
 - MaxPoolSize 12, 21
 - MaxStatements 12, 21
 - MinPoolSize 13, 22
 - NetworkProtocol 13, 22
 - PropertyCycle 13
 - RoleName 13
 - Property settings, Outbound Environment
 - DatabaseName 15
 - DataSourceName 15
 - Delimiter 16
 - Description 16
 - DriverProperties 16
 - Password 16
 - PortNumber 16
 - ServerName 17
 - User 17
 - Property settings, Outbound XA
 - PropertyCycle 22
 - RoleName 22
 - PropertyCycle 13, 22
- R**
 - requirements
 - system 6
 - RoleName 13, 22
- S**
 - Select Database Objects 28
 - Select Procedures 32
 - Select Table/Views 29
 - Select Wizard Type 27
 - ServerName 17, 19, 25
 - Setting 10
 - SQL Server eWay Database Wizard 26
 - system requirements 6
- U**
 - User 17, 19, 25