

*SeeBeyond ICAN Suite*

# DB2 Connect 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, e\*Way, and e\*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e\*Insight 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.

© 2005 SeeBeyond Technology Corporation. All Rights Reserved.

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

Version 20050520110255.

# Contents

---

## Chapter 1

<b>Introducing the DB2 Connect eWay</b>	<b>6</b>
About DB2 Connect	6
About the DB2 Connect eWay	7
What's New in This Version	7
About This Document	7
What's in This Document	7
Scope	8
Intended Audience	8
Document Conventions	8
Screenshots	8
Related Documents	8
SeeBeyond Web Site	9
Feedback	9

---

## Chapter 2

<b>Installing the DB2 Connect eWay</b>	<b>10</b>
Supported Operating Systems	10
System Requirements	10
External System Requirements	11
Installing the DB2 Connect eWay	11
After Installation	12

---

## Chapter 3

<b>Configuring the DB2 Connect eWay</b>	<b>14</b>
Working with eWay Property Sheets	14
Setting the Properties in the Outbound eWay	14
ClassName	15
Description	15
InitialPoolSize	15

LoginTimeOut	16
MaxIdleTime	16
MaxPoolSize	16
MaxStatements	16
MinPoolSize	16
NetworkProtocol	17
PropertyCycle	17
RoleName	17
<b>Setting the Environment Properties in the Outbound eWay</b>	<b>17</b>
DatabaseName	18
Delimiter	18
Description	18
DriverProperties	19
Password	19
User	19
<b>Setting the Properties in the Inbound eWay</b>	<b>20</b>
Pollmilliseconds	20
PreparedStatement	20
<b>Setting the Environment Properties in the Inbound eWay</b>	<b>21</b>
DatabaseName	21
Password	21
User	21
<b>Setting the Properties in the Outbound DB2 XA Connect eWay</b>	<b>22</b>
ClassName	22
Description	22
InitialPoolSize	23
LoginTimeOut	23
MaxIdleTime	23
MaxPoolSize	23
MaxStatements	23
MinPoolSize	24
NetworkProtocol	24
PropertyCycle	24
RoleName	24
<b>Setting the Environment Properties in the Outbound DB2 XA Connect eWay</b>	<b>25</b>
DatabaseName	25
DataSourceName	25
Delimiter	26
Description	26
DriverProperties	26
Password	26
User	26

---

## Chapter 4

<b>Using the DB2 Connect eWay Database Wizard</b>	<b>27</b>
<b>Using the Database OTD Wizard</b>	<b>27</b>

---

**Chapter 5**

<b>Implementing the DB2 Connect eWay</b>	<b>38</b>
<b>eInsight and eGate Components</b>	<b>38</b>
<b>Using the Sample Project in eInsight</b>	<b>39</b>
The Business Process	40
SelectAll	42
SelectMultiple	43
SelectOne	45
Insert	47
Update	48
Delete	50
<b>Using the Sample Project in eGate</b>	<b>51</b>
Working with the Sample Project in eGate	51
Configuring the eWays	52
Creating an External Environment	53
Deploying a Project	53
Running the Sample	53
<b>Additional Sample</b>	<b>53</b>
<b>Common DataType Conversions</b>	<b>54</b>
<b>Using OTDs with Tables, Views, and Stored Procedures</b>	<b>55</b>
The Table	56
The Query Operation	56
The Stored Procedure	57
Executing Stored Procedures	57
Prepared Statement	58
Batch Operations	60
Manipulating the ResultSet and Update Count Returned by Stored Procedure	60
<b>Editing Existing OTDs</b>	<b>63</b>
<b>Alerting and Logging</b>	<b>64</b>

---

**Appendix A**

<b>Support for WebSphere Application Server</b>	<b>65</b>
<b>Uploading the Application Server Interface</b>	<b>65</b>
<b>Creating an EAR File</b>	<b>66</b>
<b>Deploying an EAR File</b>	<b>67</b>
<b>Configuring the WebSphere Application Server</b>	<b>67</b>
Creating the Topic or Queue	67
Creating a Connection Factory	69
Installing the Application	70

<b>Index</b>	<b>76</b>
--------------	-----------

# Introducing the DB2 Connect eWay

Welcome to the DB2 Connect eWay Intelligent Adapter User's Guide. This document includes information about installing, configuring, and using the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) DB2 Connect eWay Intelligent Adapter, referred to as the DB2 Connect eWay throughout this guide.

### What's in This Chapter

- [“About DB2 Connect” on page 6](#)
- [“About the DB2 Connect eWay” on page 7](#)
- [“What's New in This Version” on page 7](#)
- [“About This Document” on page 7](#)
- [“Related Documents” on page 8](#)
- [“SeeBeyond Web Site” on page 9](#)
- [“Feedback” on page 9](#)

---

## 1.1 About DB2 Connect

DB2 Connect, IBM's database management system, makes host data directly available to Personal Computers and LAN-based workstations. It connects desktop and palm-top applications to mainframe and minicomputer host databases, leveraging enterprise information no matter where it is.

DB2 Connect provides the application enablement and communication infrastructure for connecting Web, Windows, UNIX, Linux, OS/2 and mobile applications to S/390 and AS/400 data. It also enables secure access to legacy data through intranets, extranets or the public Internet, allowing you to build new Internet applications and extend existing applications.

DB2 Connect is included in many of the DB2 UDB products, providing extensive application programming tools for developing client-server and web applications using industry standard APIs such as ODBC, ADO, OLE DB, JDBC, SQLJ, DB2 CLI and Embedded SQL.

---

## 1.2 About the DB2 Connect eWay

The eWay enables eGate Integrator Projects to exchange data with external DB2 databases. This document describes how to install and configure the eWay.

*Note:* The DB2 Connect eWay connects to DB2 via the IBM DB2 Connect driver, which is NOT packaged with the eWay. The product must be separately installed and configured.

---

## 1.3 What's New in This Version

The DB2 Connect eWay includes:

- An OTD Edit feature that allows you to edit existing OTDs. For more information on editing existing OTDs refer to [Editing Existing OTDs](#) on page 63.
- Added support for DB2 Connect version 8.2
- Added support for DB2 Universal Database (UDB) version V5R2, and V5R3 when connecting to DB2 running on an AS/400 operating system.

---

## 1.4 About This Document

This guide explains how to install, configure, and operate the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) DB2 eWay Intelligent Adapter, referred to as the DB2 Connect eWay throughout this guide.

### 1.4.1 What's in This Document

This document includes the following chapters:

- **Chapter 1 “Introducing the DB2 Connect eWay”:** Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 “Installing the DB2 Connect eWay”:** Describes the system requirements and provides instructions for installing the DB2 Connect eWay.
- **Chapter 3 “Configuring the DB2 Connect eWay”:** Provides instructions for configuring the eWay to communicate with your legacy systems.
- **Chapter 4 “Using the DB2 Connect eWay Database Wizard”:** Provides information about .sag files and using the DB2 Connect wizard.
- **Chapter 5 “Implementing the DB2 Connect eWay”:** Provides instructions for installing and running the sample Projects.
- **Appendix A “Support for WebSphere Application Server”:** Describes how to deploy an Enterprise Archive (EAR) file to the WebSphere™ Application Server.

## 1.4.2 Scope

This document describes the process of installing, configuring, and running the DB2 Connect eWay. This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the DB2 Connect eWay Javadoc files from the Enterprise Manager.

## 1.4.3 Intended Audience

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

## 1.4.4 Document Conventions

The following writing conventions are observed throughout this document.

**Table 1** Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	<b>Bold</b> text	<ul style="list-style-type: none"><li>Click <b>OK</b> to save and close.</li><li>From the <b>File</b> menu, select <b>Exit</b>.</li><li>Select the <b>logicalhost.exe</b> file.</li><li>Enter the <b>timeout</b> value.</li><li>Use the <b>getClassName()</b> method.</li><li>Configure the <b>Inbound</b> File eWay.</li></ul>
Command line arguments and code samples	Fixed font. Variables are shown in <b>bold italic</b> .	<code>bootstrap -p <b>password</b></code>
Hypertext links	<b>Blue</b> text	<a href="http://www.seebeyond.com">http://www.seebeyond.com</a>

## 1.4.5 Screenshots

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

---

## 1.5 Related Documents

The following SeeBeyond documents provide additional information about the ICAN product suite:

- *eGate Integrator User's Guide*
- *SeeBeyond ICAN Suite Installation Guide*



---

## 1.6 SeeBeyond Web Site

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

<http://www.seebeyond.com>

---

## 1.7 Feedback

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

[docfeedback@seebeyond.com](mailto:docfeedback@seebeyond.com)

# Installing the DB2 Connect eWay

This chapter describes how to install the DB2 Connect eWay.

## What's in This Chapter

- “Supported Operating Systems” on page 10
- “System Requirements” on page 10
- “External System Requirements” on page 11
- “Installing the DB2 Connect eWay” on page 11
- “After Installation” on page 12

---

## 2.1 Supported Operating Systems

The DB2 Connect eWay is available on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- HP-UX 11.0, 11i (PA-RISC)
- IBM AIX 5.1L and 5.2
- Sun Solaris 8 and 9
- Red Hat Enterprise Linux AS 2.1 (Intel x86)
- Connecting to z/OS running DB2 Connect 7.1

Although the DB2 Connect eWay, the Repository, and Logical Hosts run on the operating systems listed above, the Enterprise Designer requires the Windows operating system. Enterprise Manager can run on any operating system that supports Internet Explorer.

---

## 2.2 System Requirements

The system requirements for the DB2 Connect eWay are the same as for eGate Integrator. For information, refer to the SeeBeyond *ICAN Suite 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.

DB2 Connect is available from IBM. You must install the software prior to installing and configuring the DB2 Connect eWay.

## Using IBM Drivers on Windows or Unix Operating Systems

- DB2 Connect available from IBM, installed on the same machine as the Enterprise Designer and Logical Host. This is required to utilize the build tool.

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

## Using IBM Drivers for Connecting to a z/OS Operating System

To use the DB2 Connect eWay connecting to an z/OS operating system, you must meet the System Requirements.

**Note:** *If you have chosen to use the IBM drivers for DB2, please be aware that these drivers do not support updatable ResultSets. It is recommended that you use a prepared statement to update and insert data.*

---

## 2.3 External System Requirements

The DB2 Connect eWay supports the following software on external systems:

- DB2 Connect version 8.1 and 8.2.
- DB2 Universal Database (UDB) version 7.1 when connecting to DB2 running on an z/OS operating system.
- DB2 Universal Database (UDB) version V5R1, V5R2, and V5R3 when connecting to DB2 running on an AS/400 operating system.

---

## 2.4 Installing the DB2 Connect 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 SeeBeyond *ICAN Suite Installation Guide*, and include the following steps:

- On the Enterprise Manager, select the **DB2ConnecteWay.sar** (to install the DB2 Connect 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 **DB2ConnecteWayDocs.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.

Continue installing the eGate Enterprise Designer as instructed.

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

---

## 2.5 After Installation

After installing the eWay, you will need to create the **db2java.jar** file and load it, before it can perform its intended functions.

*Note:* After installation, make sure that the following three DB2 Connect related paths are included in your System's PATH:

- ♦ C:\PROGRA~1\IBM\SQLLIB\BIN
- ♦ C:\PROGRA~1\IBM\SQLLIB\FUNCTION
- ♦ C:\PROGRA~1\IBM\SQLLIB\SAMPLES\REPL

### To Create the .jar file

- 1 Locate the **db2java.zip** file in your DB2 Connect installation directory.
- 2 Copy the zip file to:  
`<egate_logicalhost>:\ICAN50\edesigner\userdir\modules\ext\db2conne  
ctadapter`
- 3 Rename the zip file from **db2java.zip** to **db2java.jar**.

*Important:* Your **db2java.zip** file must be the same version as the DB2 Connect you are running; otherwise errors will occur because of incompatibility.

Once the .jar file is created, you can load it in a number of ways, including:

- Loading the file using Enterprise Designer
- Copying the file directly into the Logical Host

### To Load the .jar file Using Enterprise Designer

When using ICAN, do the following:

- 1 On the Environment tab, create a new Logical Host in your DB2 Environment.
- 2 Right-click the new Logical Host and select **Upload File** on the menu.
- 3 On the Upload Third Party Files window, click **Add**. Navigate to the **db2java.jar** you copied to:

```
<egate_logicalhost>:\\ICAN50\edesigner\userdir\modules\ext\db2connectadapter
```

- 4 Click **OK**.
- 5 Exit and then restart Enterprise Designer.

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.

**Note:** *Failing to follow these instructions results in the eWay not recognizing your DB2 database.*

### To Copy the .jar file Directly into the Logical Host

- 1 Copy the file to the following location and rename it from **db2java.zip** to **db2java.jar**:

```
<egate_logicalhost>:\\stcis\lib.
```

**Note:** *The <egate\_logicalhost>:\\stcis\lib location only appears after running a project.*

# Configuring the DB2 Connect eWay

This chapter describes how to set the properties of the DB2 Connect eWay.

### What's in This Chapter

- [“Setting the Properties in the Outbound eWay” on page 14](#)
- [“Setting the Environment Properties in the Outbound eWay” on page 17](#)
- [“Setting the Properties in the Inbound eWay” on page 20](#)
- [“Setting the Environment Properties in the Inbound eWay” on page 21](#)
- [“Setting the Properties in the Outbound DB2 XA Connect eWay” on page 22](#)
- [“Setting the Environment Properties in the Outbound DB2 XA Connect eWay” on page 25](#)

---

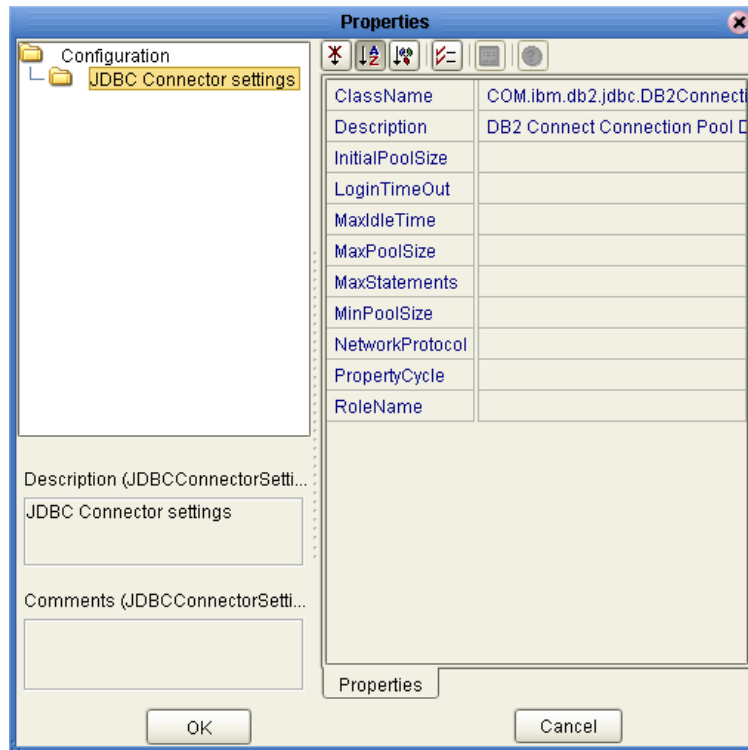
## 3.1 Working with eWay Property Sheets

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.

### 3.1.1. Setting the Properties in the Outbound eWay

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

**Figure 1** Properties of the Outbound eWay



## 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.ibm.db2jdbc.DB2ConnectionPoolDataSource`.

## 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.

## 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 can 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.

## 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.

## 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.

## 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.

## RoleName

### Description

An initial SQL role name.

### Required Values

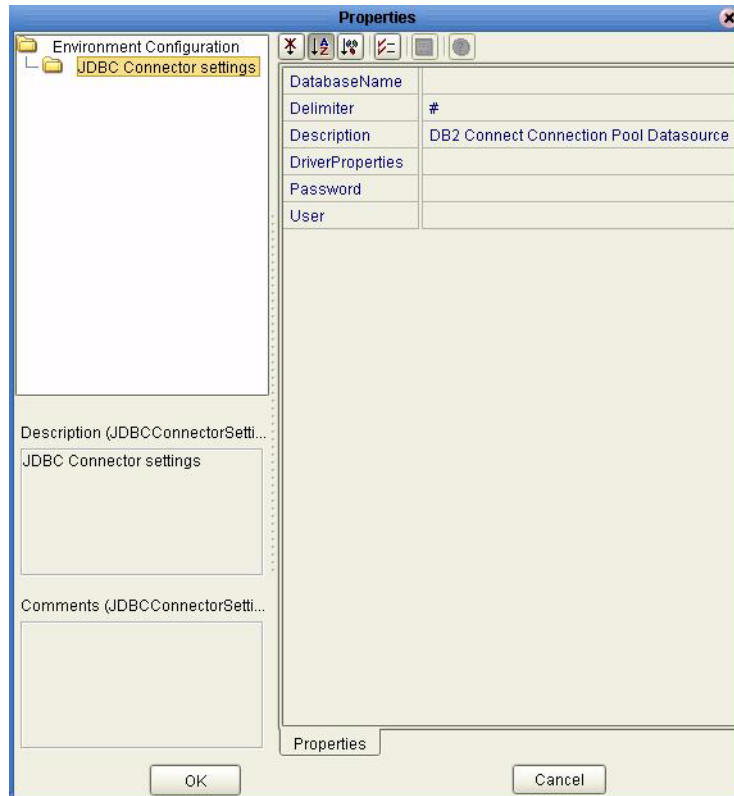
Any valid string.

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

### 3.1.2 Setting the Environment Properties in the Outbound eWay

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

**Figure 2** Environment Settings of the Outbound eWay on Windows and Unix



## DatabaseName

### Description

Specifies the name of the database instance as configured on the local client.

### Required Values

Any valid string.

## 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 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>##".

## Password

### Description

Specifies the password used to access the database.

### Required Values

Any valid string.

## User

### Description

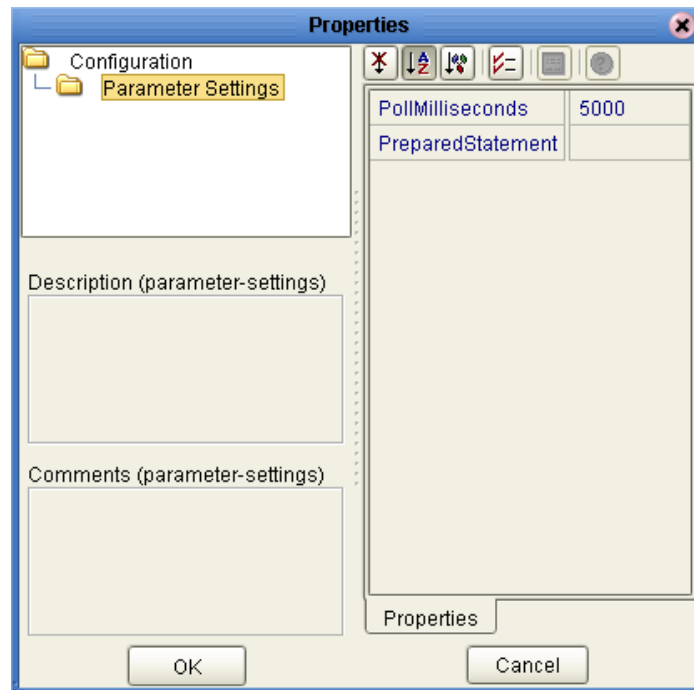
Specifies the user name the eWay uses to connect to the database.

### Required Values

Any valid string.

### 3.1.3. Setting the Properties in the Inbound eWay

**Figure 3** Properties of the Inbound eWay



#### **Pollmilliseconds**

##### **Description**

The polling interval in milliseconds.

##### **Required Value**

A valid numeric value. The default is 5000.

#### **PreparedStatement**

##### **Description**

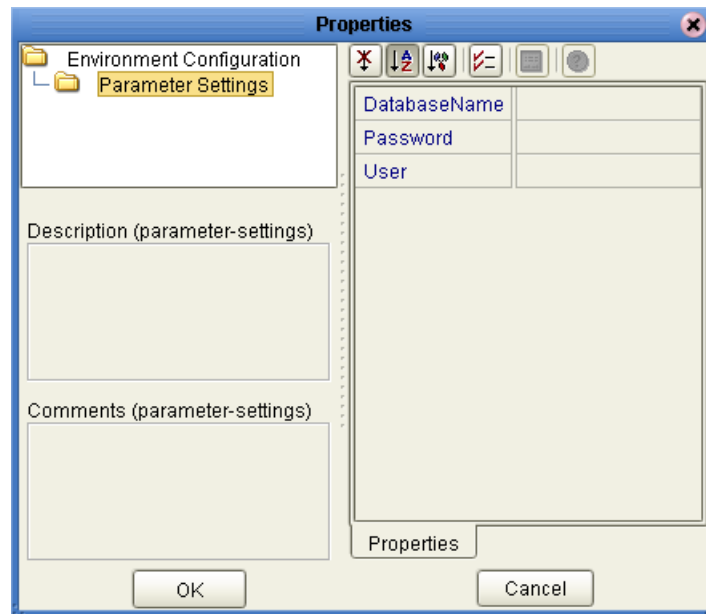
The Prepared Statement used for polling against the database.

##### **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.

### 3.1.4. Setting the Environment Properties in the Inbound eWay

**Figure 4** Environment Settings of the Inbound eWay



#### 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.

#### User

##### Description

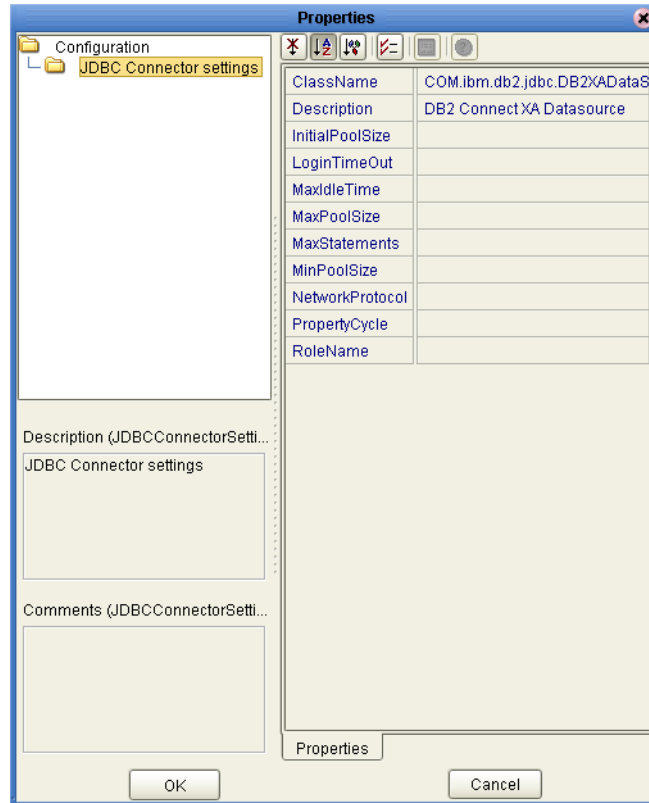
Specifies the user name the eWay uses to connect to the database.

##### Required Values

Any valid string.

### 3.1.5. Setting the Properties in the Outbound DB2 XA Connect eWay

**Figure 5** Properties of the Outbound DB2 XA Connect eWay



#### 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.ibm.db2.jdbc.DB2XADataSource**.

#### Description

#### Description

Enter a description for the database. The default is **DB2 Connect XA Datasource**.

#### 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.

## 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 can 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.

## 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.

## MinPoolSize

### Description

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.

## 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.

## RoleName

### Description

An initial SQL role name.

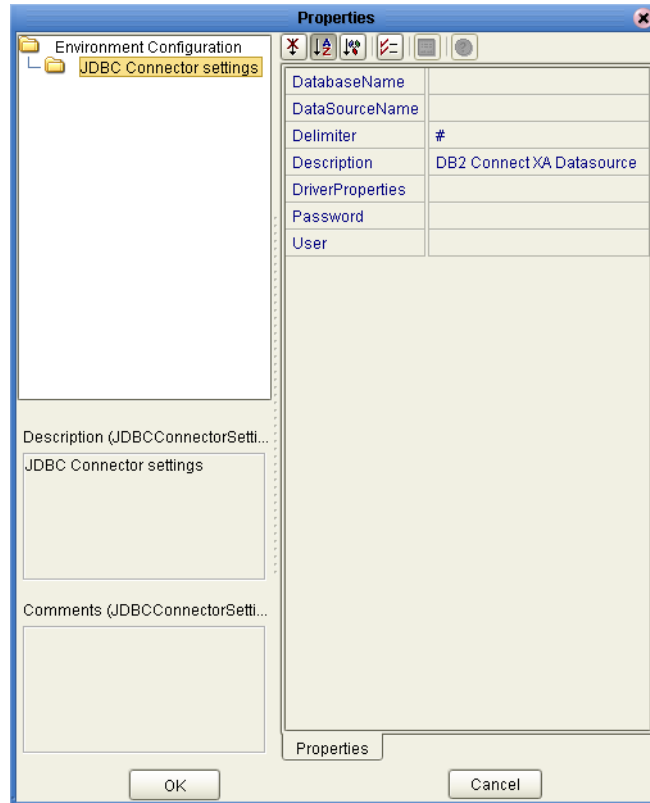
### Required Values

Any valid string.



### 3.1.6. Setting the Environment Properties in the Outbound DB2 XA Connect eWay

**Figure 6** Environment Settings of the Outbound DB2 XA Connect eWay



#### DatabaseName

#### Description

Specifies the name of the database instance.

#### Required Values

Any valid string.

#### DataSourceName

#### Description

Provides 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. The default is **DB2 Connect XA Datasource**.

### 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#jdbc:db2:<database>##".

## Password

### Description

Specifies the password used to access the database.

### 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 DB2 Connect eWay Database Wizard

This chapter describes how to use the DB2 Connect eWay Database Wizard to build OTD's.

## What's in This Chapter

- [“Select Wizard Type” on page 27](#)
- [“Connect to Database” on page 28](#)
- [“Select Database Objects” on page 29](#)
- [“Select Table/Views” on page 29](#)
- [“Select Procedures” on page 32](#)
- [“Add Prepared Statements” on page 35](#)
- [“Specify the OTD Name” on page 36](#)

---

## 4.1 Using the Database OTD Wizard

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

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

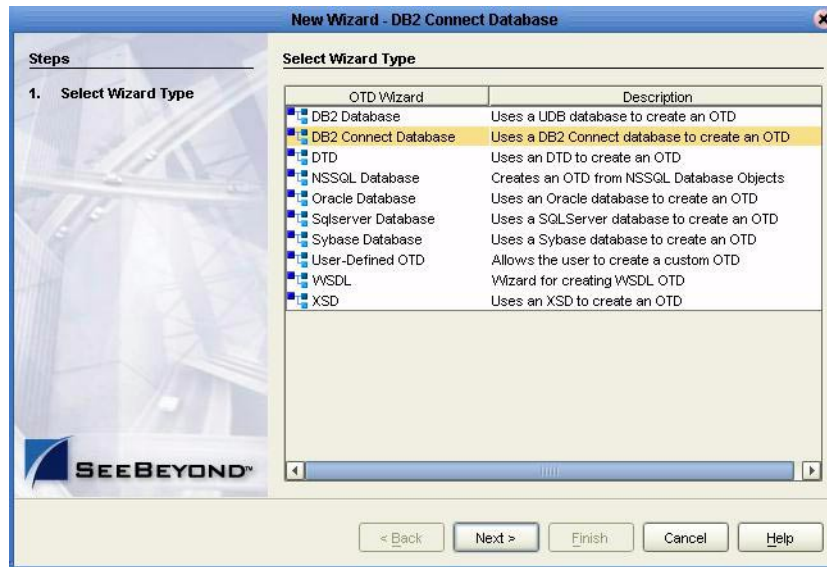
**Note:** *Database OTDs are not messagable. For more information on messagable OTDs, see the eGate Integrator User's Guide.*

### Select Wizard Type

- 1 On the Enterprise Explorer, right click on the project and select **Create an Object Type Definition** from the shortcut menu.

- From the OTD Wizard Selection window, select the **DB2 Connect Database** and click **Next**. See Figure 7.

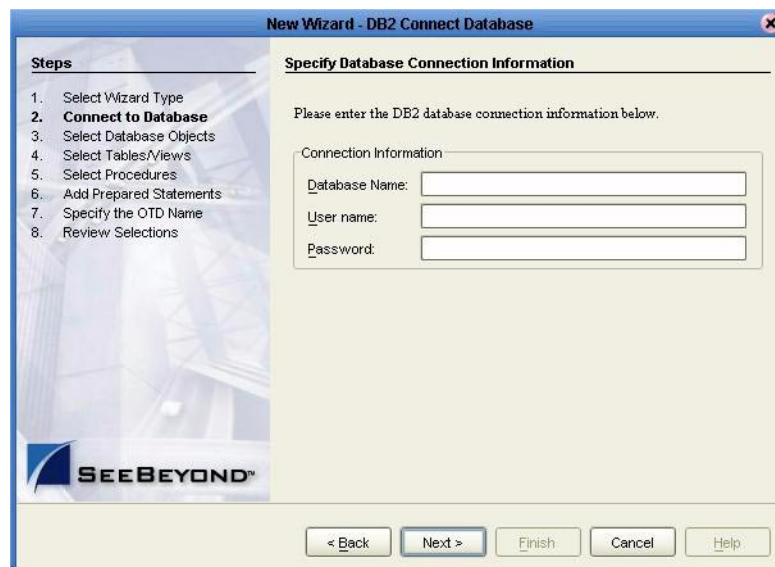
Figure 7 OTD Wizard Selection



### Connect to Database

- Specify the **Database Name** that you configured in DB2 Connect. Enter a **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.*

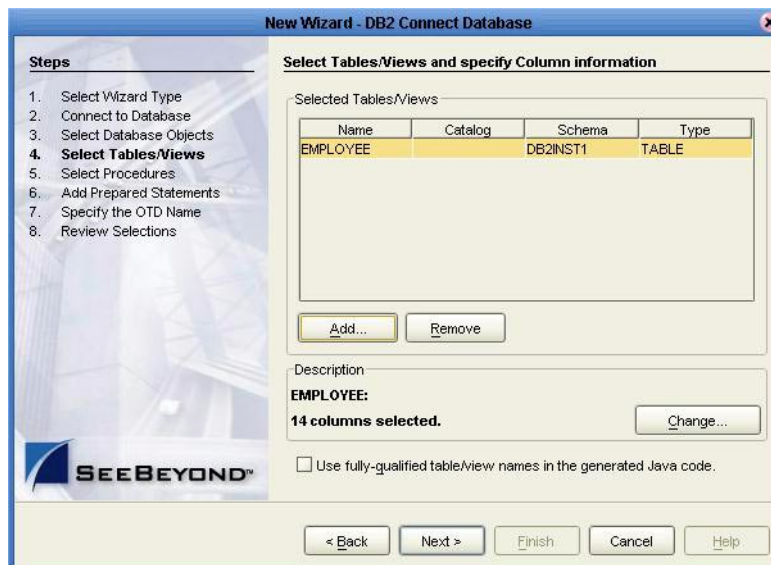
**Figure 9** Select Database Objects



### Select Table/Views

- 1 In the **Select Tables/Views** window, click **Add**. See Figure 10.

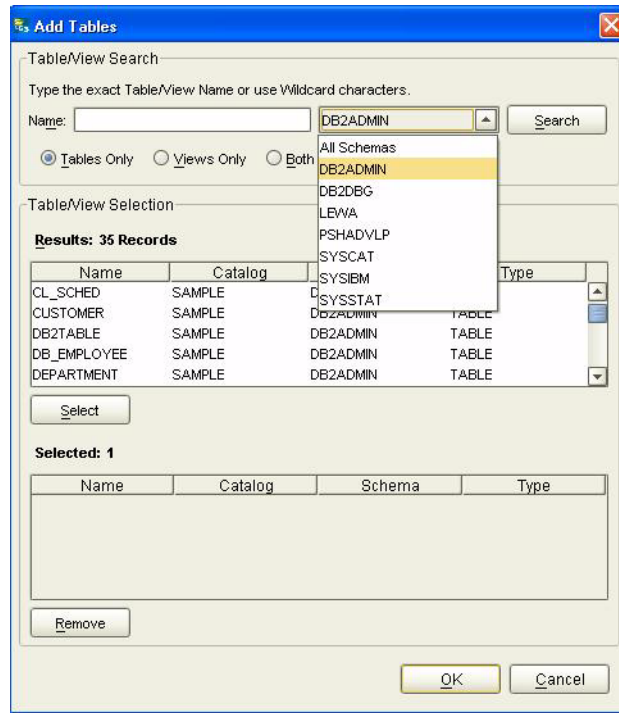
**Figure 10** Select Tables/Views



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

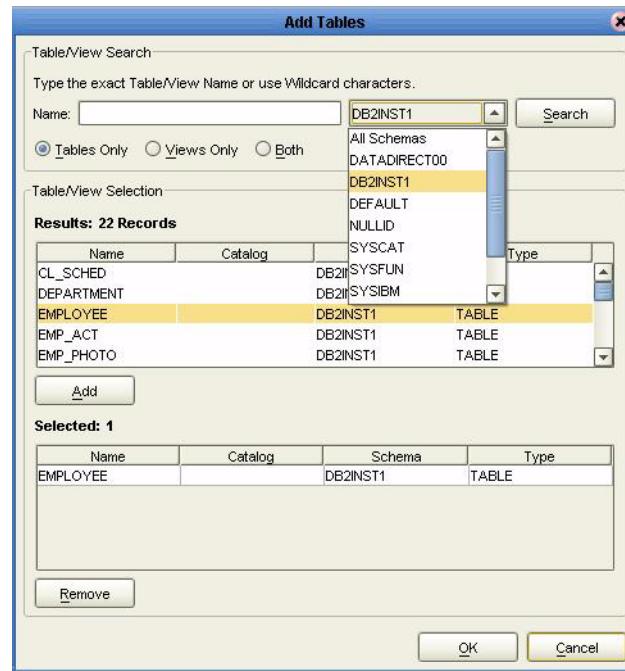
- From the **Table/View Name** drop down list, select the location of your database table and click **Search**. See Figure 11. You can search for **Table/View Names** by entering a table name. The use of wildcard characters of '?' and '\*' as part of your Table/View name search allow for greater search capabilities. For example, "AB?CD" or "AB\*CD".

**Figure 11** Database Wizard - All Schemes



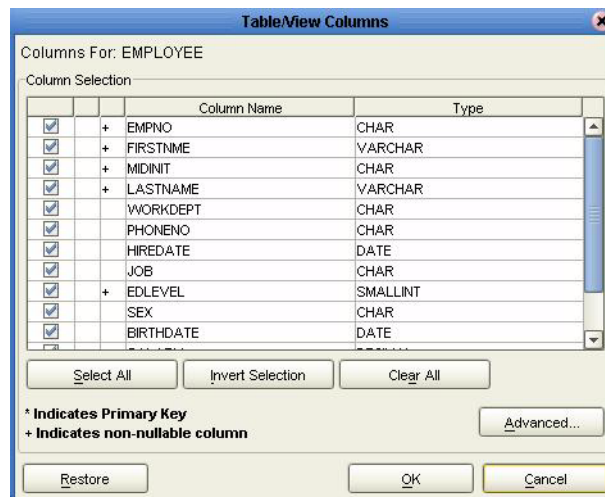
- Select the table of choice and click **OK**.  
The table selected is added to the **Selected** window. See Figure 12.

**Figure 12** Selected Tables/Views window with a table selected



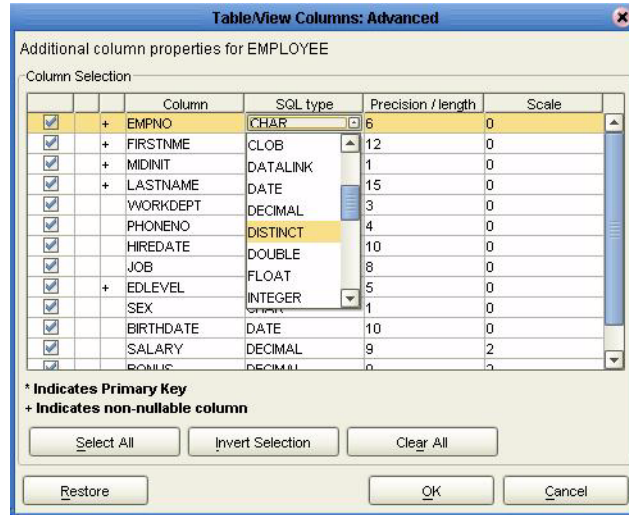
- 5 On the **Selected Tables/Views** window, review the table(s) you have selected. To make changes to the selected Table or View, click **Change**. If you do not wish to make any additional changes, click **Next** to continue.
- 6 If you clicked **Change** on the Selected Tables/Views window, you can select or deselect your table columns on the **Table/View Columns** window. You can also change the data type for each table by highlighting the data type and selecting a different one from the drop down. See Figure 13.

**Figure 13** Table/View Columns



- Click **Advanced** to change the data type, precision/length, or scale. In general, do not change the precision/length or the scale. Once you have finished your table choices, click **OK**. See Figure 14.

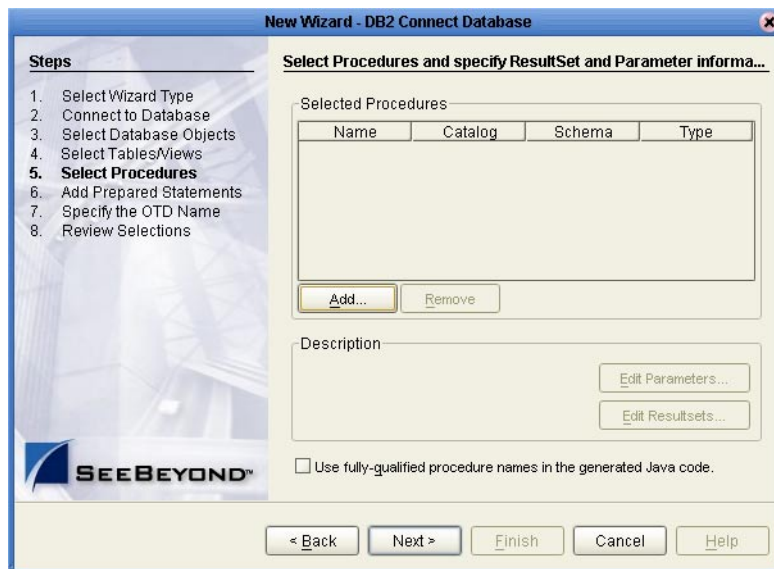
**Figure 14** Table/View Columns – Advanced



### Select Procedures

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

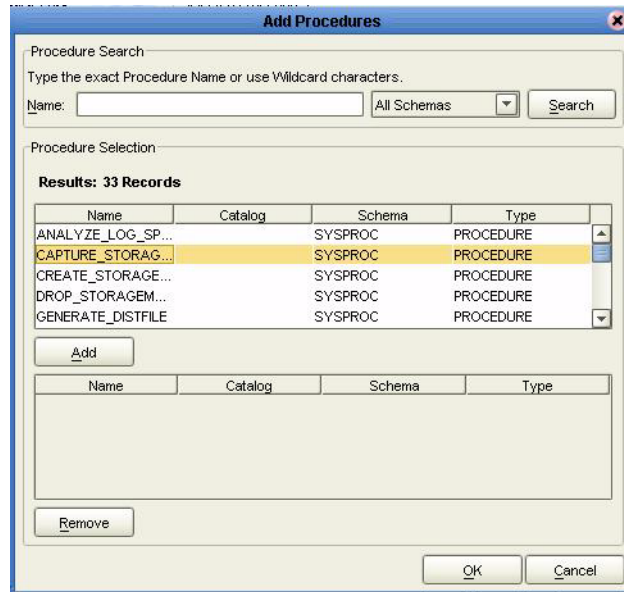
**Figure 15** Select Procedures and specify Resultset and Parameter Information



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

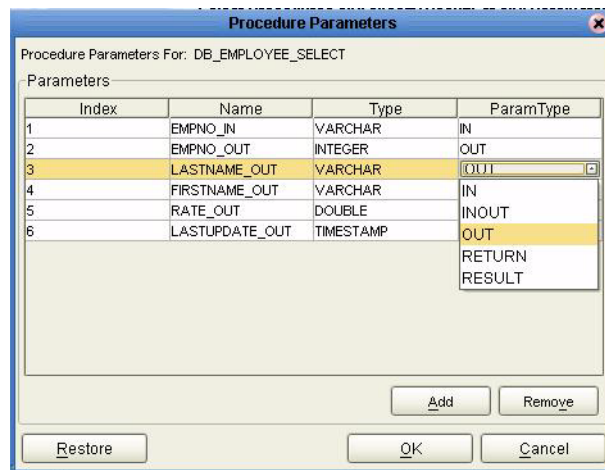


Figure 16 Add Procedures



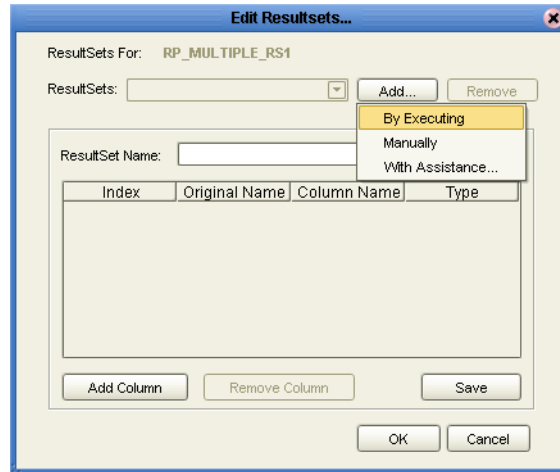
- 4 On the **Select Procedures and specify Resultset and Parameter Information** window click **Edit Parameters** to make any changes to the selected Procedure. See Figure 17.

Figure 17 Procedure Parameters



- 5 To restore the data type, click **Restore**. When finished, click **OK**.
- 6 To select how you would like the OTD to generate the nodes for the Resultset click **Edit Resultsets**.
- 7 Click **Add** to add the type of Resultset node you would like to generate (see Figure 18).

**Figure 18** 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.

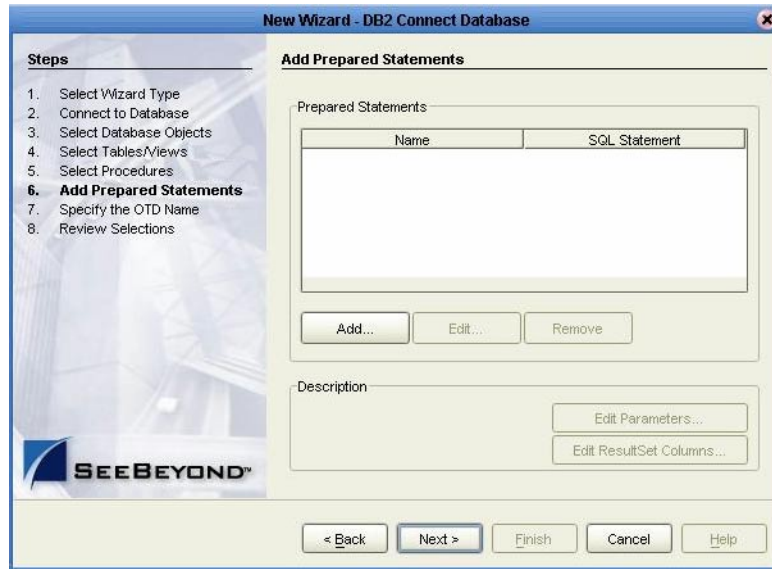
If you modify the ResultSet generated by the "**Execute**" mode of the Database Wizard you need to make sure the indexes match the Stored Procedure. This assures your ResultSet indexes are preserved.

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

## Add Prepared Statements

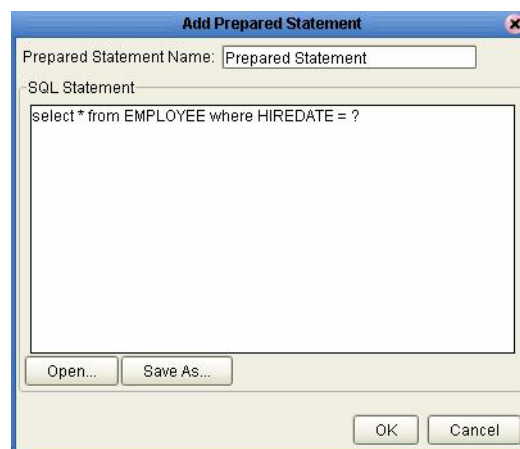
- 1 On the **Add Prepared Statements** window, click **Add**.

**Figure 19** Prepared Statement



- 2 Enter the name of a Prepared Statement or create a SQL statement by clicking in the SQL Statement window. When finished creating the statement, click **Save As** giving the statement a name. This name will appear as a node in the OTD. Click **OK**. See Figure 20.

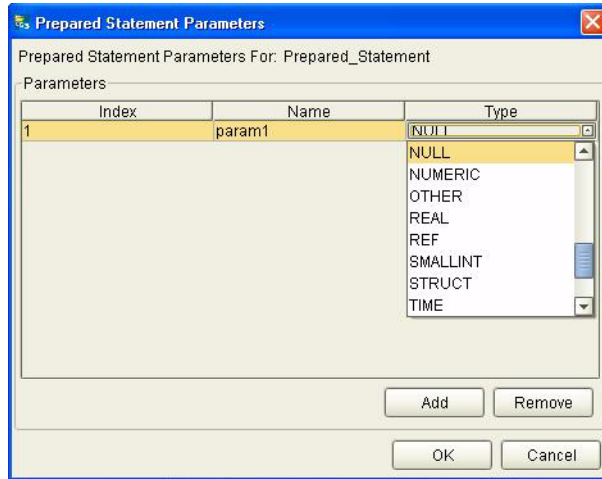
**Figure 20** Prepared SQL Statement



- 3 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 data type by clicking in the **Type** field and selecting a different type from the list.

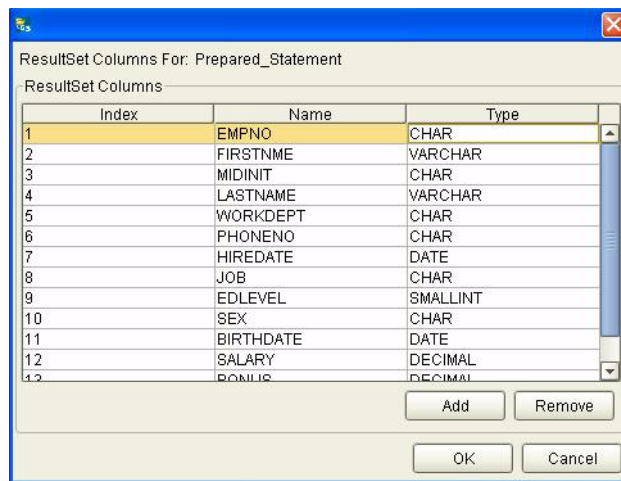
- 4 Click **Add** if you want to add additional parameters to the Statement or highlight a row and click **Remove** to remove it. Click **OK**. See Figure 21.

**Figure 21** Edit the Prepared Statement Parameters



- 1 To edit the Resultset Columns, click **Edit Resultset Columns**. Both the Name and Type are editable. Click **OK**. See Figure 22.

**Figure 22** ResultSet Columns

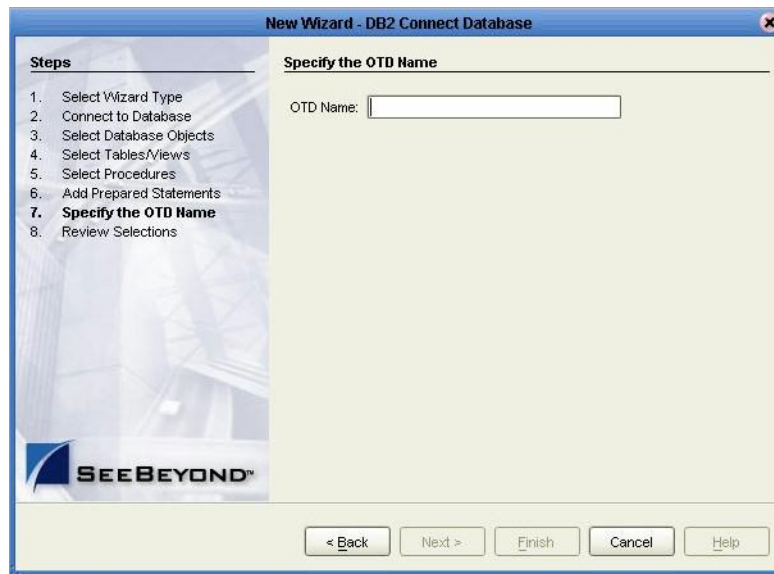


- 2 On the Add Prepared Statements window, click **OK**.

### Specify the OTD Name

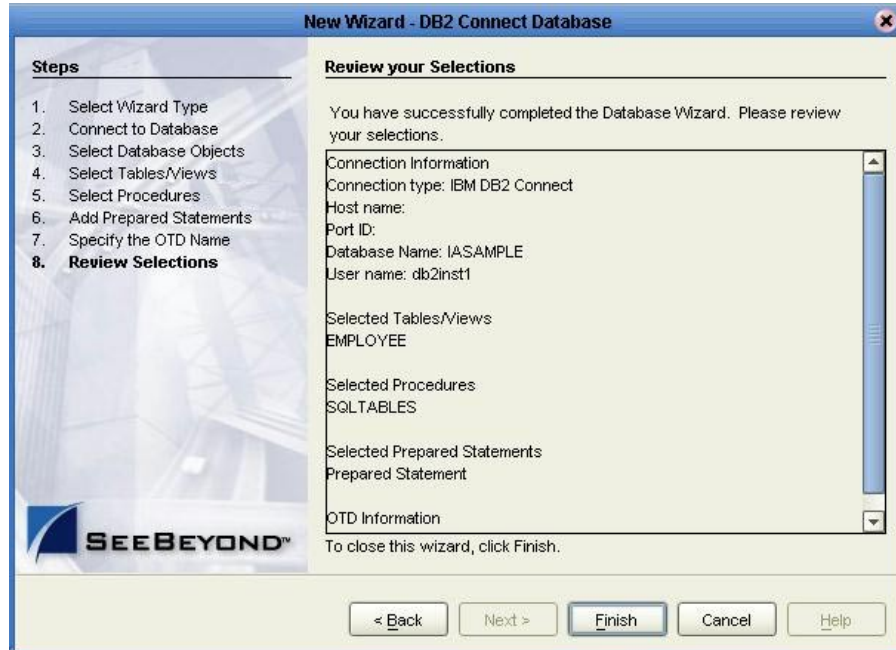
- 1 Enter a name for the OTD. The OTD contains the selected tables and the package name of the generated classes. See Figure 23.

Figure 23 Naming an OTD



- 2 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 24.

Figure 24 Database Wizard - Summary



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

# Implementing the DB2 Connect eWay

This chapter discusses how to build a DB2 Connect eWay project in a production environment.

## What's in This Chapter

- [“eInsight and eGate Components” on page 38](#)
- [“Using the Sample Project in eInsight” on page 39](#)
- [“Using the Sample Project in eGate” on page 51](#)
- [“Additional Sample” on page 53](#)
- [“Common Data Type Conversions” on page 54](#)
- [“Using OTDs with Tables, Views, and Stored Procedures” on page 55](#)
- [“Editing Existing OTDs” on page 63](#)
- [“Alerting and Logging” on page 64](#)

---

## 5.1 eInsight 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.

## 5.2 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 **DB2Connect\_BPEL\_Sample.zip** file contained in the eWay sample folder on the installation CD-ROM.

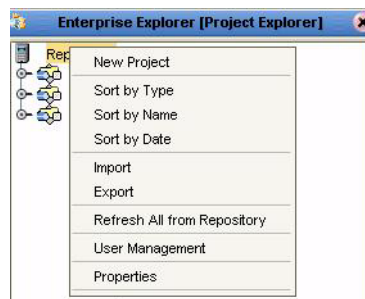
**Note:** *eInsight is a Business Process modeling tool. If you have not purchased eInsight, contact your sales representative for information on how to do so.*

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

### Importing the Sample Project

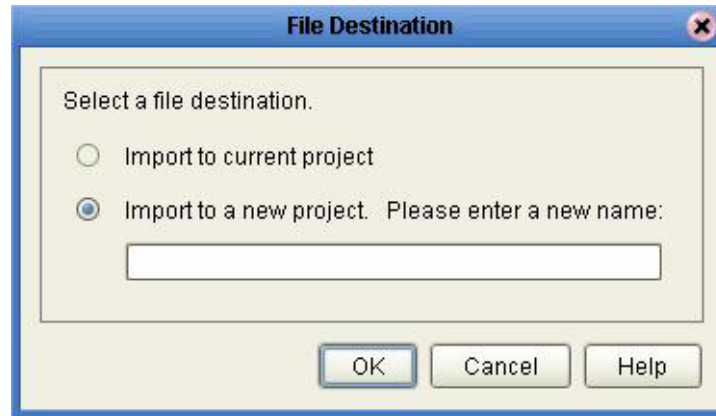
- 1 On the Enterprise Explorer highlight the repository and right click. Select **Import Project**. See Figure 25.

**Figure 25** Importing the sample project



- 1 In the **Select File to Import** window, browse to the location of the sample folder and select the following .zip file **DB2Connect\_sampleBPEL.zip** and click **Open**.
- 2 On the **File Destination** window, select **Import a new project**. Please enter a new name. Enter a name for the sample project and click OK. See [Figure 26](#).

**Figure 26** Select the project file destination



- 3 Click the **Refresh All From Repository** icon located on the **Enterprise Explorer** toolbar.

## The Business Process

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

**Table 2** Sample Project Data

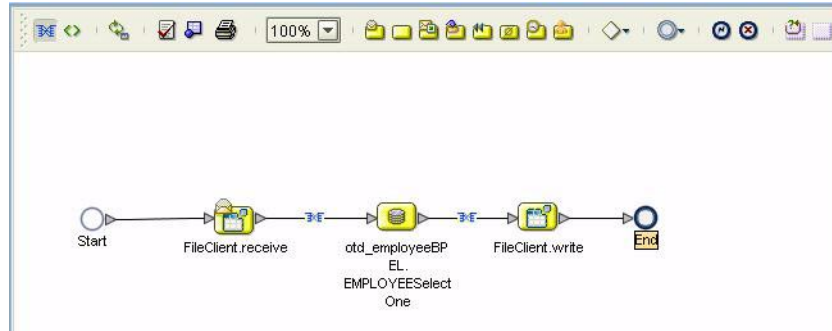
Column Name	Mapping	Data Type	Data Length
EMPNO	EMPNO	char	6
FIRSTNME	FIRSTNME	varchar	12
MIDINIT	MIDINIT	char	1
LASTNAME	LASTNAME	varchar	15
WORKDEPT	WORKDEPT	char	3
PHONENO	PHONENO	char	4
HIREDATE	HIREDATE	date	10
JOB	JOB	char	8
EDLEVEL	EDLEVEL	smallint	5
SEX	SEX	char	1
BIRTHDATE	BIRTHDATE	date	10
SALARY	SALARY	decimal	9
BONUS	BONUS	decimal	9
COMM	COMM	decimal	9

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 27** 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 DB2 Connect eWay. You can open a file specified in the eWay and view its contents before and after the Business Process is executed.

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 when using a repeating node to output all rows.
SelectOne	number of rows where() clause. Optional	Returns the first row that fits the condition of the where() clause

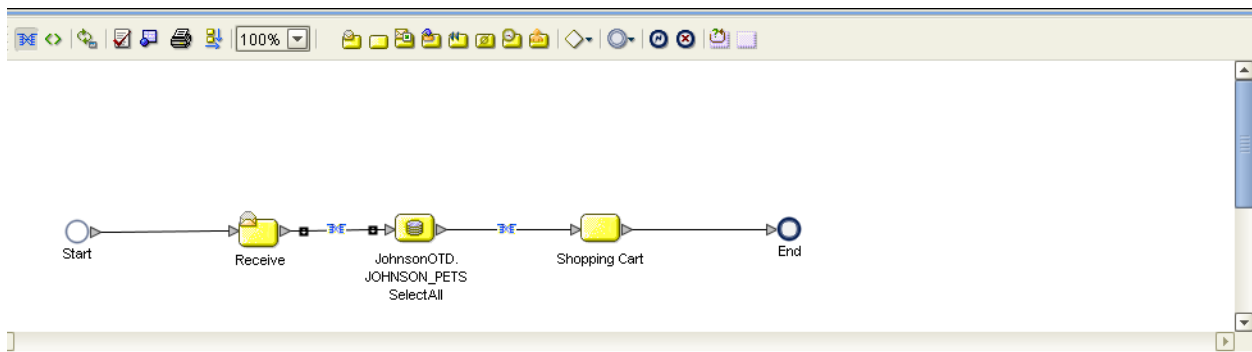
eInsight Operation	Input	Output
Insert	definition of new item to be inserted	Returns status.
Update	where() clause	Returns status.
Delete	where() clause	Returns status.

### 5.2.1 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.

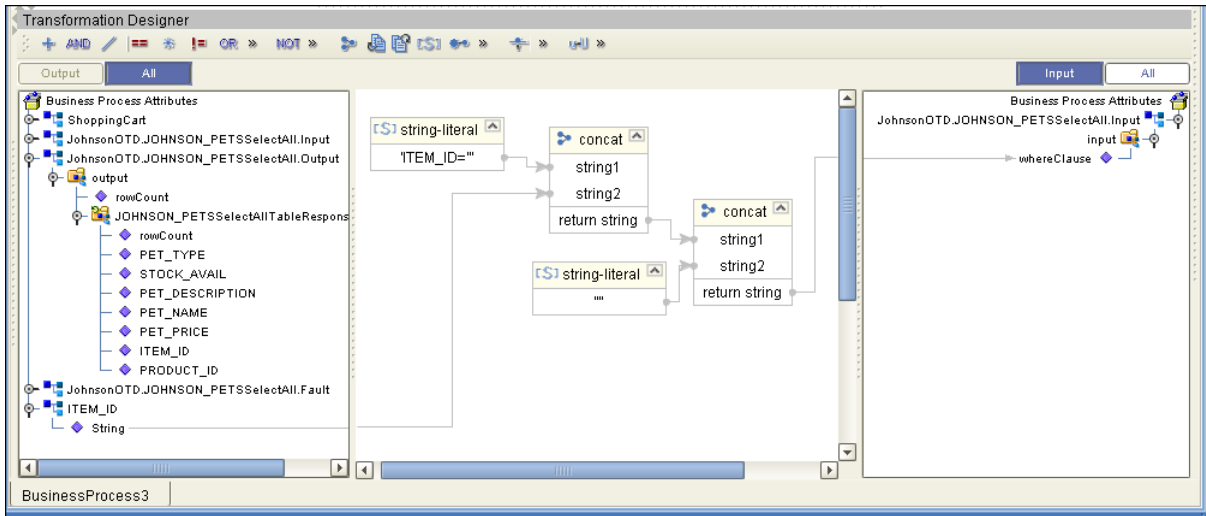
The figure below shows a sample eInsight Business Process using the SelectAll operation. In this process, the SelectAll operation returns all rows where the ITEM\_ID matches the selected ITEM\_ID to the shopping cart.

**Figure 28** SelectAll Sample Business Process



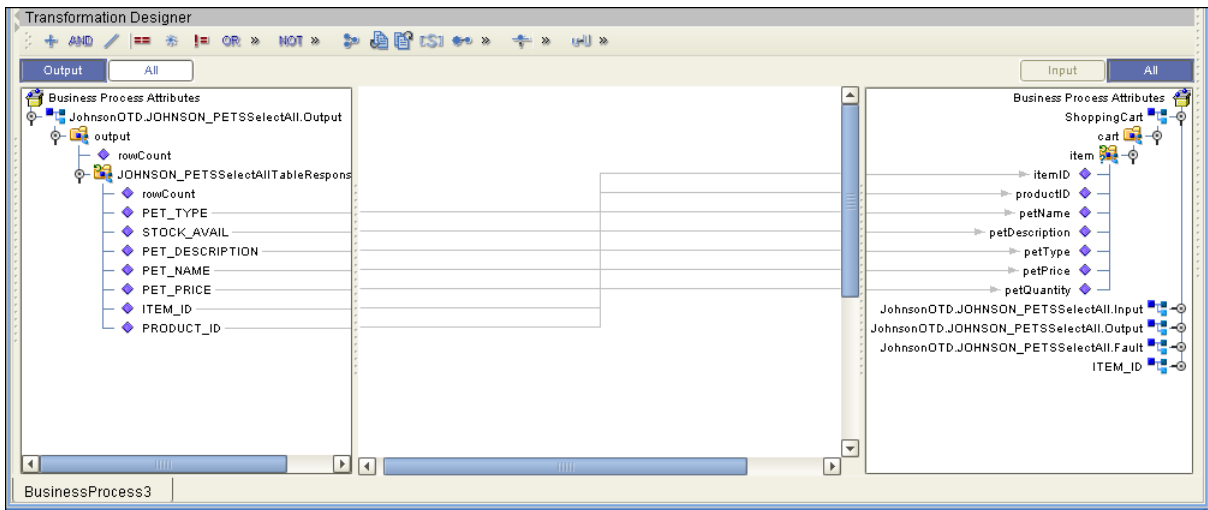
The figure below shows the definition of the where() clause for the SelectAll operation.

Figure 29 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 30 SelectAll Output

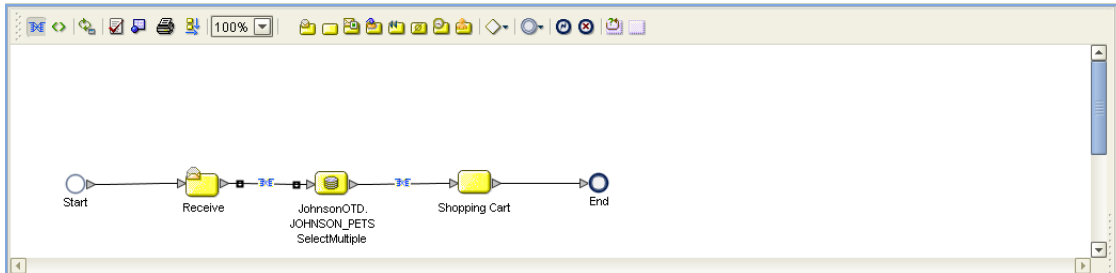


### 5.2.2 SelectMultiple

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.

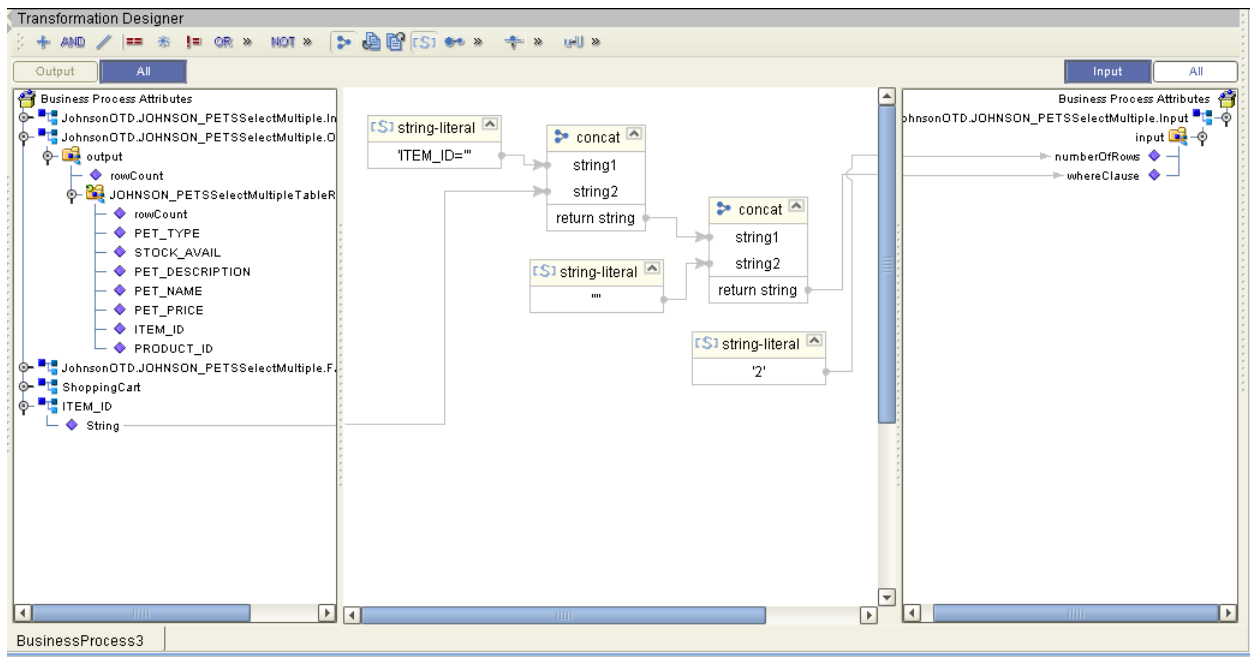
The figure below shows a sample eInsight Business Process using the SelectMultiple operation. In this process, the SelectMultiple operation returns the first two rows where the ITEM\_ID matches the selected ITEM\_ID to the shopping cart.

Figure 31 SelectMultiple Sample Business Process



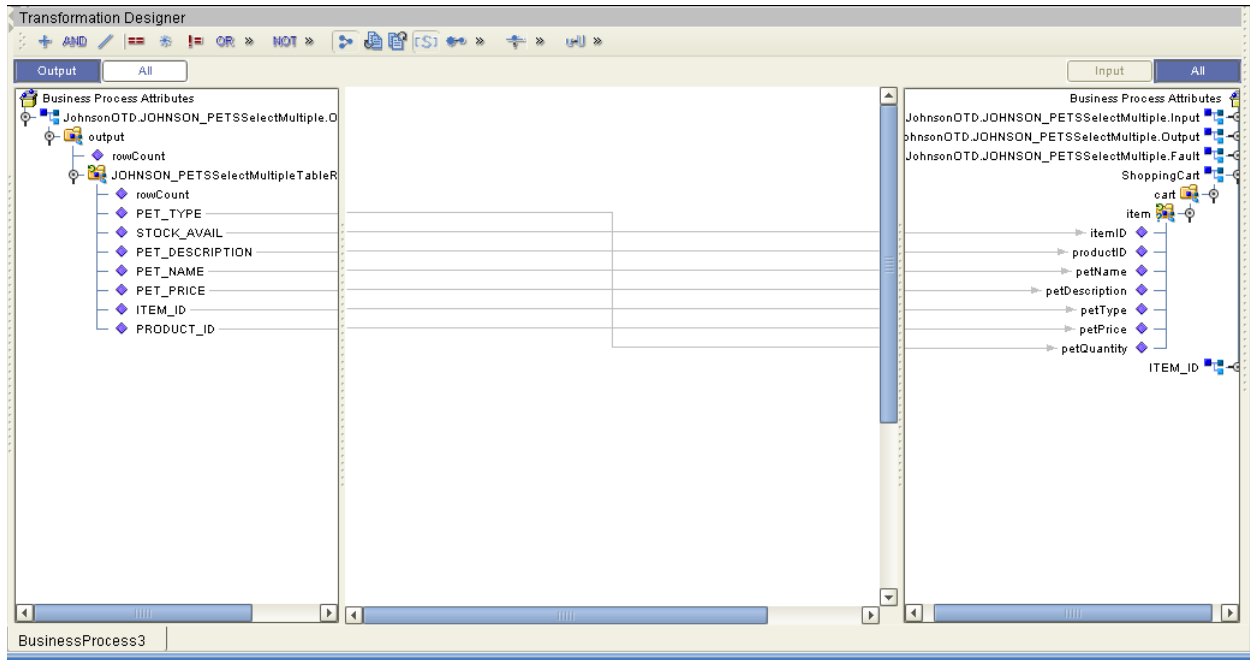
The figure below shows the definition of the number of rows and where() clause into the input for the SelectMultiple operation. You could also use an empty string or Item\_ID='123'.

Figure 32 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 33 SelectMultiple Output

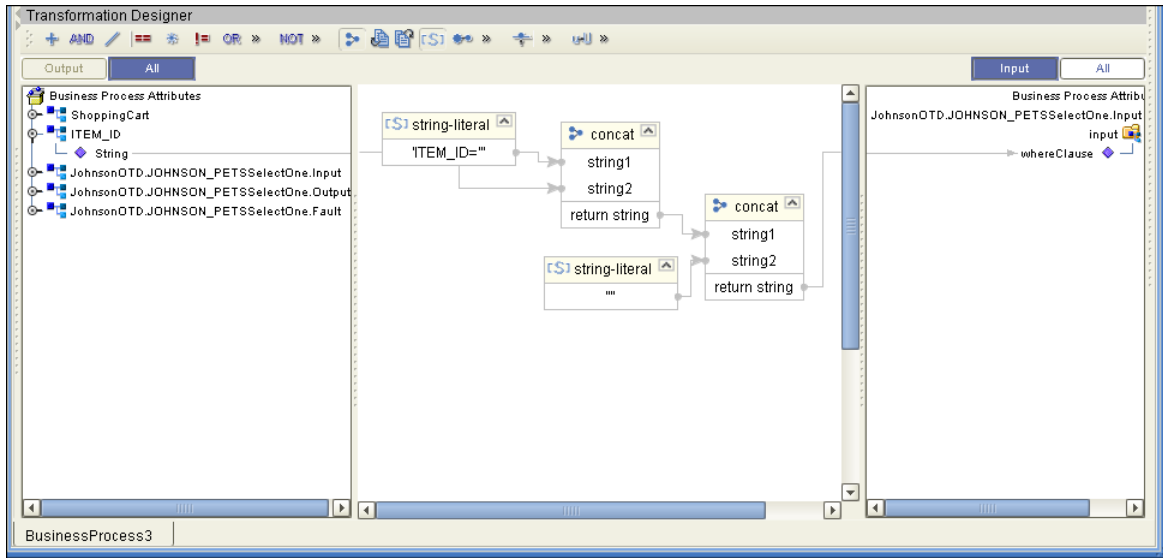


### 5.2.3 SelectOne

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.

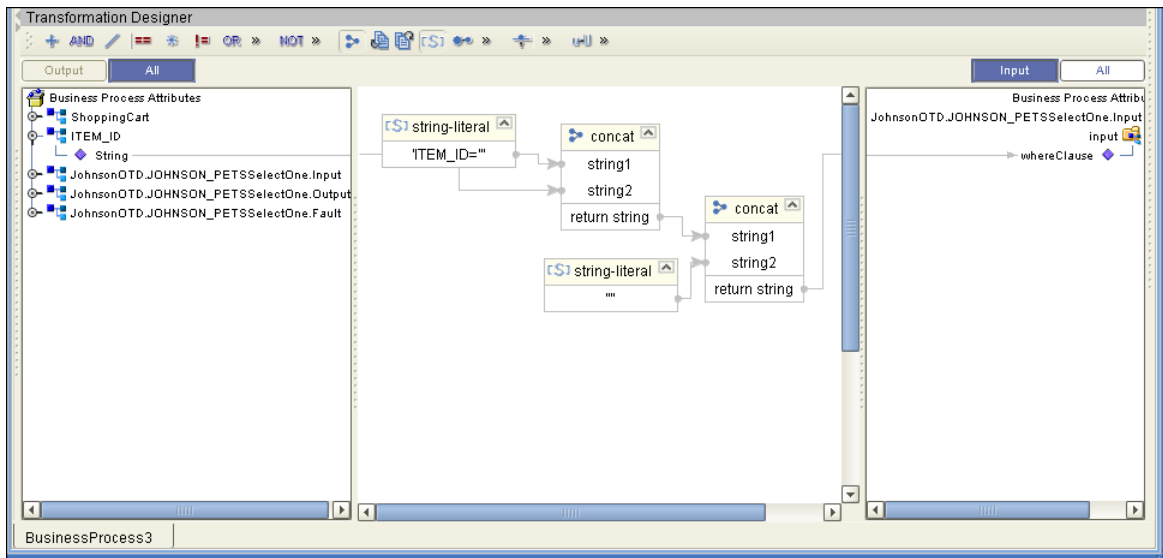
The figure below shows a sample eInsight Business Process using the SelectOne operation. In this process, the SelectOne operation returns the first row where the ITEM\_ID matches the specified ITEM\_ID to the shopping cart.

Figure 34 SelectOne Sample Business Process



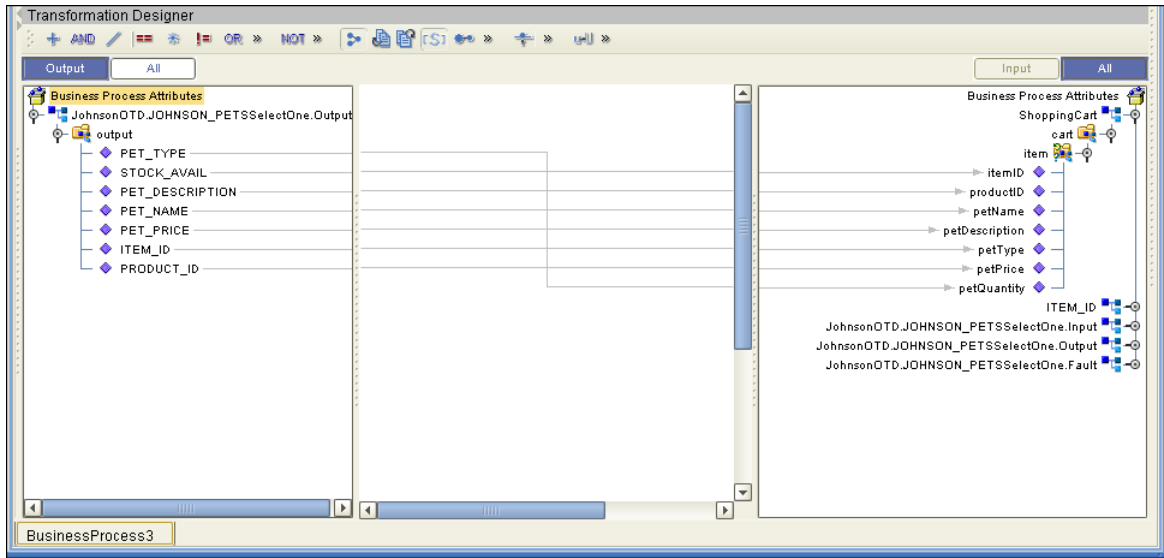
The figure below shows the definition of the where() clause for the SelectOne operation.

Figure 35 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 36 SelectOne Output



## 5.2.4 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 37 Insert Sample Business Process

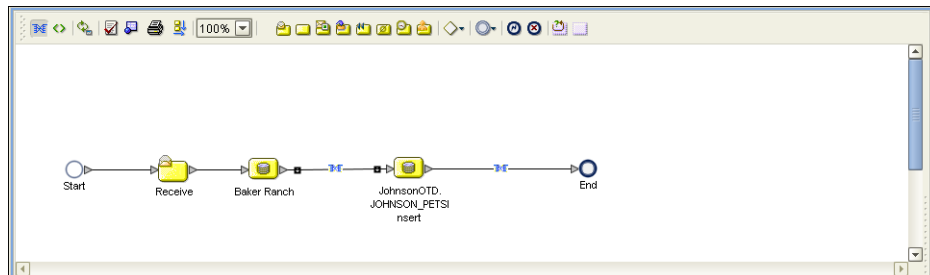
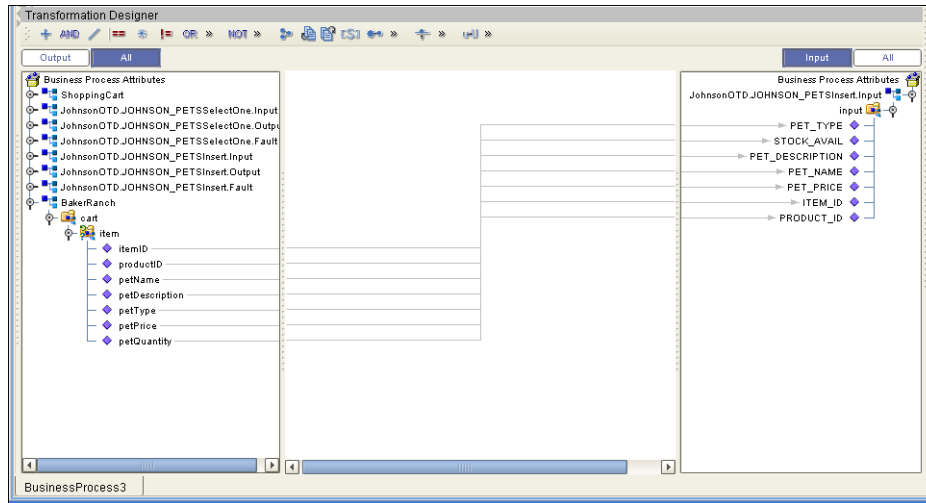


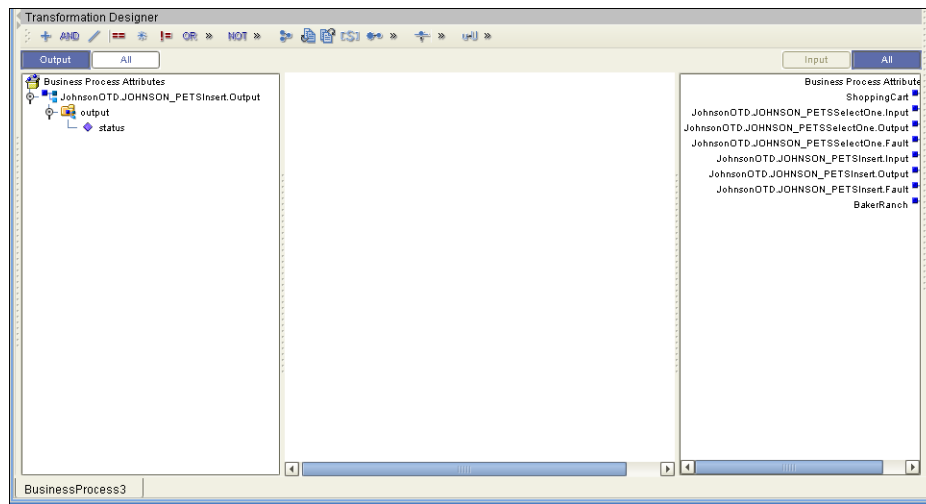
Figure 38 shows the definition of the input for the Insert operation.

Figure 38 Insert Input



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

Figure 39 Insert Output



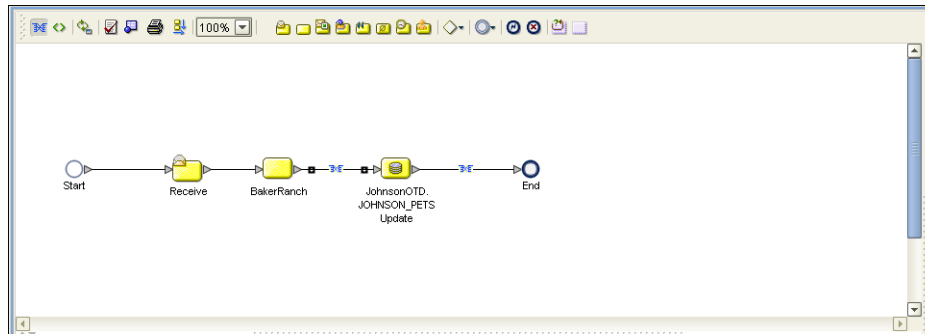
### 5.2.5 Update

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.

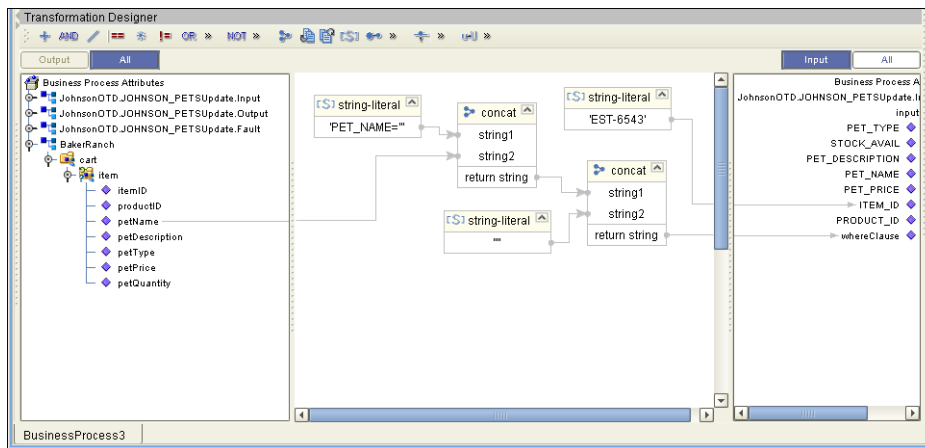


**Figure 40** Update Sample Business Process



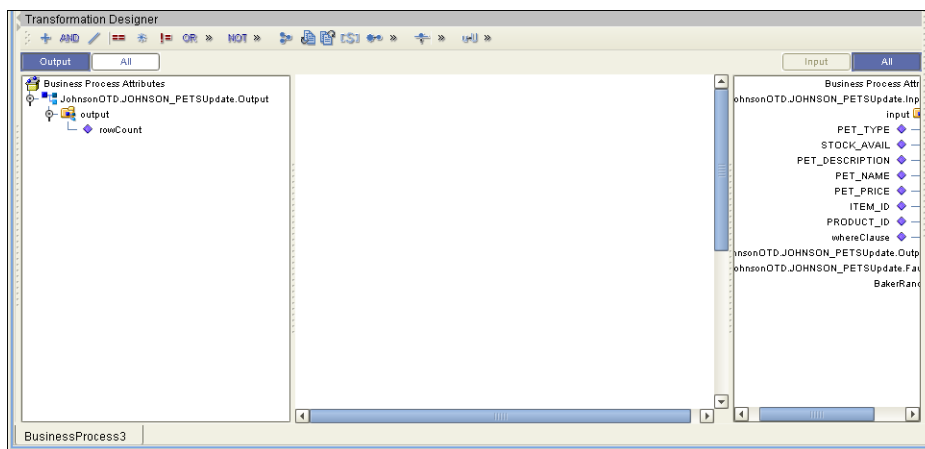
The figure below shows the definition of the where() clause for the Update operation.

**Figure 41** Update Input



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

**Figure 42** Update Output



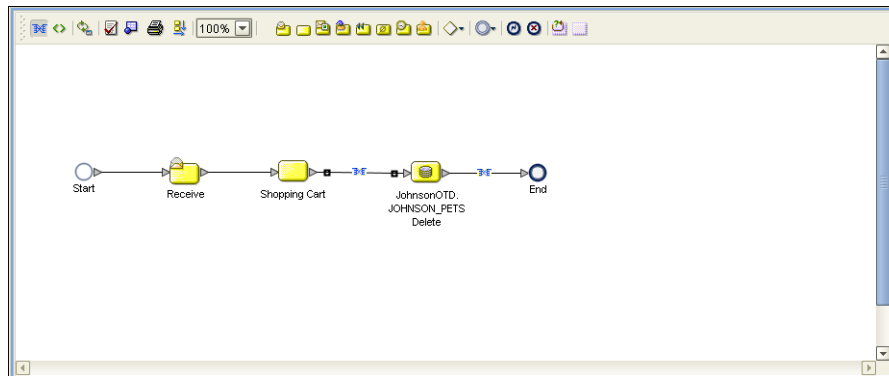
### 5.2.6 Delete

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.

*Note: If a where() clause is not defined, all rows will be deleted.*

**Figure 43** Delete Sample Business Process



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

**Figure 44** Delete Input

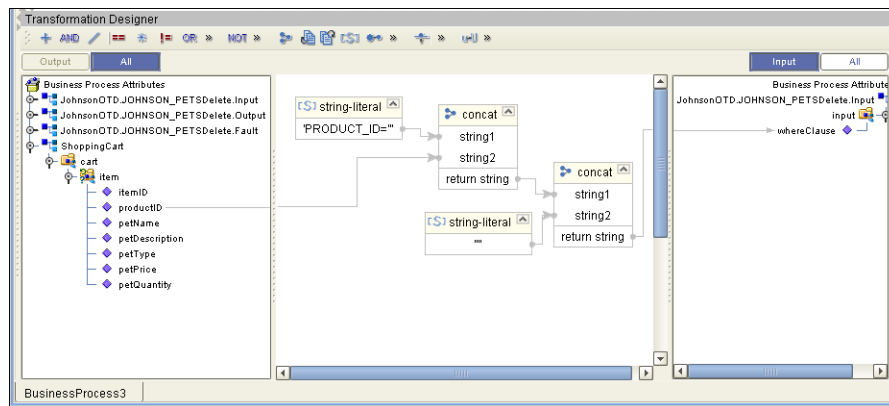
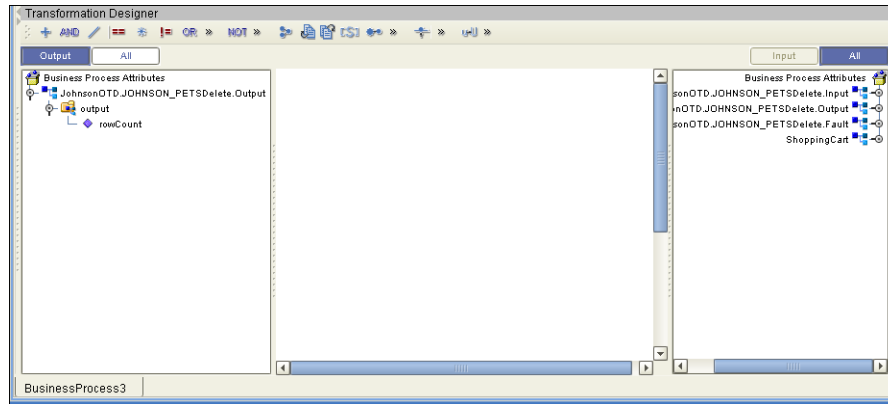


Figure 45 shows the output of the Delete operation, which is a status indicating the number of rows deleted.

**Figure 45** Delete Output



## 5.3 Using the Sample Project in eGate

To import the sample project **DB2Connect\_JCE\_Sample.zip** follow the instructions given in [Importing the Sample Project](#) on page 39.

### 5.3.1. Working with the Sample Project in eGate

This sample project selects 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:

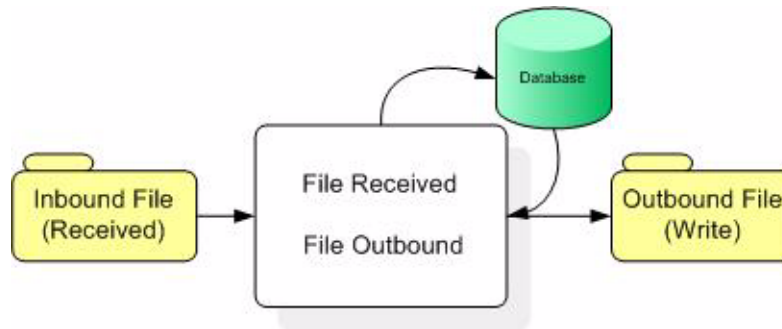
**Table 3** Sample Project Data

Column Name	Mapping	Data Type	Data Length
EMPNO	EMPNO	char	6
FIRSTNME	FIRSTNME	varchar	12
MIDINIT	MIDINIT	char	1
LASTNAME	LASTNAME	varchar	15
WORKDEPT	WORKDEPT	char	3
PHONENO	PHONENO	char	4
HIREDATE	HIREDATE	date	10
JOB	JOB	char	8
EDLEVEL	EDLEVEL	smallint	5
SEX	SEX	char	1
BIRTHDATE	BIRTHDATE	date	10
SALARY	SALARY	decimal	9

Column Name	Mapping	Data Type	Data Length
BONUS	BONUS	decimal	9
COMM	COMM	decimal	9

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 46** Database project flow



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

### 5.3.2. Configuring the eWays

The sample uses an inbound and an outbound File eWay as well as an outbound DB2 Connect eWay. To configure the sample projects eWays, use the following information. For additional information on the DB2 Connect properties, see [Working with eWay Property Sheets](#) on page 14.

To configure the Inbound File eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between the **FileIn** and **jcdEmployeeSelect1**.
- 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 DB2 Connect eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between the **jcdEmployeeSelect1** and **DB2 Connect1** database.
- 2 On the resulting **Templates** window, select **Outbound DB2** and click **OK**.
- 3 On the Properties window, enter the appropriate configurations for the Outbound DB2 eWay and click **OK**. See [Working with eWay Property Sheets](#) on page 14. 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 **jcdEmployeeSelect1** and **FileOut** 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**.

### 5.3.3. Creating an 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:

- 5 On the Environment Explorer, highlight and right-click the DB2 profile. Select **Properties**. Enter the configuration information required for your Outbound DB2 Connect eWay. See [Working with eWay Property Sheets](#) on page 14.

### 5.3.4 Deploying a Project

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

### 5.3.5. Running the Sample

For instruction on how to run the 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.

---

## 5.4 Additional Sample

An additional sample **DB2\_Connect\_Additional\_Sample.zip** is provided to illustrate an Update, Insert, and Delete within a Prepared Statement. To use this sample do the following:

- 1 On the Enterprise Manager, click the **Documentation** tab.
- 2 Click on **DB2 Connect eWay Intelligent Adapter**.
- 3 Click on the **Download Sample** link.
- 4 Click on the **DB2\_Connect\_Additional\_Sample.zip** file.
- 5 Unzip this file to a local file.

## 5.5 Common DataType Conversions

**Table 4** The DB2 Connect eWay Insert or Update Operations for Text/String Input Data

DB2 Data Type	OTD/Java Data Type	Java Method or New Constructor to Use (Default: Java Method)	Sample Data
Int	Int	<b>Integer</b> java.lang.Integer.parseInt(String s)	123
Smallint	BigDecimal	<b>Call a New Constructor BigDecimal:</b> java.math.BigDecimal(String)	123
Number	BigDecimal	<b>Call a New Constructor BigDecimal:</b> java.math.BigDecimal(String)	123
Decimal	BigDecimal	<b>Call a New Constructor BigDecimal:</b> java.math.BigDecimal(String)	123
BigInteger	Long	<b>Long:</b> java.lang.Long.parseLong(String)	123
Short	Short	<b>Short:</b> java.lang.Short.parseShort(String)	123
Real	Float	<b>Float:</b> java.lang.Float.parseFloat(String)	2454.56
Float	Double	<b>Double:</b> java.sql.Double.parseDouble(String)	2454.56
Double	Double	<b>Double:</b> java.sql.Double.parseDouble(String)	2454.56
Timestamp	Timestamp	<b>TimeStamp:</b> java.sql.Timestamp.valueOf(String)	2003-09-04 23:55:59
Time	Time	<b>Time:</b> java.sql.Time.valueOf(String)	11:15:33
Date	Date	<b>Date:</b> java.sql.Date.valueOf(String)	2003-09-04
Varchar2	String	Direct Assign	Any character
Char	String	Direct Assign	Any character

**Table 5** The DB2 Connect eWay Select Operations for Text/String Output Data

DB2 Data Type	OTD/Java Data Type	Java Method or New Constructor to Use (Default: Java Method)	Sample Data
Int	Integer	<b>Integer</b> java.lang.Integer.toString(Int)	123
Smallint	BigDecimal	<b>BigDecimal:</b> java.math.BigDecimal.toString()	123
Number	BigDecimal	<b>BigDecimal:</b> java.math.BigDecimal.toString()	123
Decimal	BigDecimal	<b>BigDecimal:</b> java.math.BigDecimal.toString()	123
Short	Short	<b>Short:</b> java.lang.Short.toString(short)	123
Real	Float	<b>Float:</b> java.lang.Float.toString(Float)	2454.56
Float	Double	<b>Double:</b> java.sql.Double.parseDouble(String)	2454.56
Double	Double	<b>Double:</b> java.sql.Double.parseDouble(String)	2454.56
Timestamp	Timestamp	<b>TimeStamp:</b> java.sql.Timestamp.toString()	2003-09-04 23:55:59
Time	Time	<b>Time:</b> java.sql.Time.toString()	11:15:33
Date	TimeStamp	<b>Date:</b> java.sql.Date.toString()	2003-09-04
Varchar2	String	Direct Assign	Any character
Char	String	Direct Assign	Any character

## 5.6 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.

## 5.6.1 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 a query on a table.

### 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 employee_db_employee_1.marshallToString to
Text
        FileClient_1.setText(
employee_db_employee_1.marshallToString() );

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

## 5.6.2 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, employee_db_employee_1, insert_DB.Insert_DBOTD insert_DB_1
)
    throws Throwable
    {
        //
        //@map:employee_db_with_top_db_employee_1.unmarshalFromString(Text)
employee_db_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(
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() ) );

        //@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();
    }

```

### 5.6.3 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.

**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.*

For example:

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

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

**To Insert**

```
public void receive( com.stc.connector.appconn.file.FileTextMessage
input, com.stc.connector.appconn.file.FileApplication
FileClient_1, pS_Insert.PS_InsertOTD PS_Insert_1 )

```

```
throws Throwable
{
    //@map:Copy Text to Param1
    PS_Insert_1.getPSInsert().setParam1( input.getText() );

    //@map:Copy "James" to Param2
    PS_Insert_1.getPSInsert().setParam2( "James" );

    //@map:Copy "M" to Param3
    PS_Insert_1.getPSInsert().setParam3( "M" );

    //@map:Copy "Smith" to Param4
    PS_Insert_1.getPSInsert().setParam4( "Smith" );

    //@map:Copy "HR" to Param5
    PS_Insert_1.getPSInsert().setParam5( "HR" );

    //@map:Copy "9995551212" to Param6
    PS_Insert_1.getPSInsert().setParam6( "9995551212" );

    //@map:PSInsert.executeUpdate
    PS_Insert_1.getPSInsert().executeUpdate();

    //@map:Copy "Record Inserted" to Text
    FileClient_1.setText( "Record Inserted" );

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

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

### To Update

```
public void receive( com.stc.connector.appconn.file.FileTextMessage
input, com.stc.connector.appconn.file.FileApplication
FileClient_1, pS_Update.PS_UpdateOTD PS_Update_1 )
throws Throwable
{
    //@map:Copy Text to Param1
    PS_Update_1.getPSUpdate().setParam1( input.getText() );

    //@map:Copy "6265551212" to Param2
    PS_Update_1.getPSUpdate().setParam2( "6265551212" );

    //@map:PSUpdate.executeUpdate
    PS_Update_1.getPSUpdate().executeUpdate();

    //@map:Copy "Record Updated" to Text
    FileClient_1.setText( "Record Updated" );
}
```

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

## 5.6.4 Batch Operations

While the Java API used by SeeBeyond does not support traditional bulk insert or update operations, there is an equivalent feature that can achieve comparable results, with better performance. This is the “Add Batch” capability. The only modification required is to include the **addBatch()** method for each SQL operation and then the **executeBatch()** call to submit the batch to the database server. Batch operations apply only to Prepared Statements.

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

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

## 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

DB2 Connect 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 you
    // 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 can 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 can 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.

## 5.7 Editing Existing OTDs

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

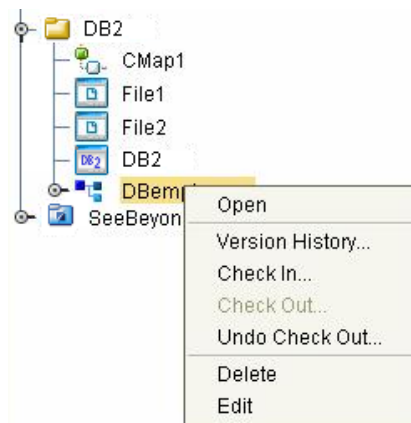
- Add or Remove Table/Views.
- Change data types by selecting a different one from a list.
- Add or Remove columns from a **Table** object.
- Add or Remove **Prepared Statement** objects.
- Edit **Prepared Statement** objects.
- Add or Remove **Stored Procedure** objects.
- Edit **Stored Procedure Resultsets**.

### To Edit an Existing OTD

When a minor change is needed for a DB2 Connect OTD, there is no need to rebuild it from scratch. Another option is to edit the OTD. To edit an OTD, complete the following steps:

- 1 In the Enterprise Explorer, right-click on the OTD. From the submenu, click **Edit** (see Figure 47). The Database Connection Information Wizard opens.

**Figure 47** OTD Edit Menu Item



- 2 Connect to the DB2 Connect database by entering the applicable information in the wizard. Once the connection is established, the Database Wizard opens, allowing you to make modifications to the OTD.
- 3 Once you have completed editing the OTD, click the **Finish** button to save the changes.

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

When editing an OTD, another instance of the database can be connected to, with the following conditions:

- The same type of DB2 database must be used. Because of incompatibility of certain features in the databases, switching between DB2 databases that run on z/OS, AS/400 and Windows/UNIX is not supported.
- The same version of the database should be used unless the newer version is compatible with the older version.
- Tables in the database must be defined with the same definition.
- The stored procedures must be identical.
- For tables/stored procedures built with 'qualified-name', the schema name for the tables/stored procedures must be identical in both database instances.

---

## 5.8 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*.



# Support for WebSphere Application Server

This section describes how to deploy an Enterprise Archive (EAR) file to the WebSphere™ Application Server. This includes information on installing the WebSphere Application Server interface and configuring the selected Application server to deploy the EAR file.

### What's in this Appendix

- [“Uploading the Application Server Interface” on page 65](#)
- [“Creating an EAR File” on page 66](#)
- [“Deploying an EAR File” on page 67](#)
- [“Configuring the WebSphere Application Server” on page 67](#)

---

## A.1 Uploading the Application Server Interface

To support the WebSphere Application Server, the following files must be uploaded to your system and installed:

- websphereintegserver.sar
- webspherejmsmessageserver.sar

For information on uploading and installing the selected .sar files see the *SeeBeyond ICAN Suite Installation Guide*.

## A.2 Creating an EAR File

To create an EAR file from an eGate project, include the following steps during the creation of the project:

- 1 If you are using Topics or Queues in your project, make the following changes to the JMS Properties Sheet (accessed from the Connectivity Map). Set the properties to the following values:
  - ◆ Set the **JMS Client > Basic > Transaction Mode** property to **XA**
  - ◆ Set the **JMS Client > Basic > Run-as principal > Use for JMS connection** property to **false**
  - ◆ Set the **JMS Client > Advanced > Durability** property to **Nondurable**
  - ◆ Set the **JMS Client > Advanced > Security > Audit** property to **no**

**Note:** *The JMS Client > Basic > Run-as principal > Name property is limited to 12 characters*

- 2 During the creation of the project Environment do the following:
  - A Create the Logical Host in the Environment.
  - B From the Environment Explorer tree, right-click the logical host and select **New WebSphere Application Server** from the shortcut menu. The application server, **WebSphereSvr1**, is added to the Logical Host box and the Environment Explorer tree.
  - C If you are using JMS (Topics or Queues) you must also add the **New WebSphere JMS Server** to the Logical Host. From the Environment Explorer tree, right-click the logical host and select **New WebSphere JMS Server** from the shortcut menu. The WebSphere JMS message server, **WSMessageSvr1**, is added to the Logical Host box and the Environment Explorer tree.
- 3 During the creation of the Deployment Profile, do the following:
  - A If the service containing the Collaboration fails to be mapped to the WebSphere Application Server, drag and drop the Service to the WebSphere Application Server, **WebSphereSvr1**, in the Logical Host box.
  - B If any JMS Topics or Queues fail to be mapped to the WebSphere JMS Server, drag and drop the Topic or Queue to the WebSphere JMS Server, **WSMessageSvr1**, in the Logical Host box.
  - C Once all of the components have been mapped, click **Activate**. When prompted whether to apply the project to the logical host immediately, click **No**.
- 4 The project's EAR file is now available in the following location:

```
<ICAN>/repository/data/files/<environment name>/<logical host name>/<application server name> directory
```

where <ICAN> is the directory where the ICAN Suite is installed, <environment name> is the name of the project Environment, <logical host name> is the name of the logical host, and <application server name> is the name of the selected Application Server.

**Note:** For an eGate project EAR file to deploy to the WebSphere Application Server, the project must not contain any inbound eWays.

### A.3 Deploying an EAR File

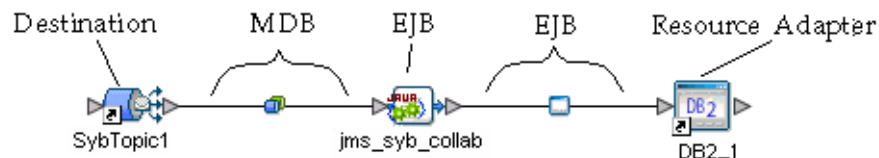
To deploy an EAR file using the WebSphere Application Server, you create the following components from the Application Server's Administrative Console:

- **Topics or Queues:** One Topic or Queue is needed for each Topic or Queue in the eGate project.
- **Connection Factories:** A Connection Factory enables JMS Clients to create JMS connections with predefined attributes. If the project contains Topics, a Topic Connection Factory is required. If it contains Queues a Queue Connection Factory is needed. If the project contains both Topics and Queues, create both a Queue and a Topic Connection Factory.

WebSphere also requires the following:

- **Listener Ports:** A listener port must be created for every MDB the project contains. The MDB is represented in the project's Connectivity Map as the connection between a **Topic or Queue** and a **Service** (see Figure 1).

**Figure 1** Project/EAR File



### A.4 Configuring the WebSphere Application Server

To deploy an EAR file in the WebSphere Application Server, start the Administrative Console, and do the following:

#### Creating the Topic or Queue

- 1 From the left pane of the WebSphere Administrative console, select **Resources** and click **WebSphere JMS Provider**.

- From the **WebSphere JMS Provider** window, under **Additional Properties** (shown in Figure 50), click **WebSphere Queue Destinations** or **WebSphere Topic Destinations**, depending on whether you are creating a topic or queue. For this example click **WebSphere Topic Destinations** to create a topic.

**Figure 2** WebSphere Server Administrative Console - Additional Properties

Additional Properties	
<a href="#">WebSphere Queue Connection Factories</a>	
<a href="#">WebSphere Topic Connection Factories</a>	
<a href="#">WebSphere Queue Destinations</a>	
<a href="#">WebSphere Topic Destinations</a>	

- From the **WebSphere Topics Destinations** (or **Queue Destinations**) window click the **New** button.
- From the **Topic** or **Queue Destinations** configuration window, enter a name for the new Topic or Queue in the **Name** field. Enter the same name in the **JNDI Name** field. Enter the string value used to identify the Topic in the **Topic** field (for this example, **Topic1**) as displayed in Figure 3. Click **OK**.

**Figure 3** WebSphere Server Administrative Console - Create a Topic

WebSphere Application Server Administrative Console  
Version 5

Home | Save | Preferences | Logout | Help

User ID: USER

Messages(s)  
Changes have been made to your local configuration. Click [Save](#) to apply changes to the master configuration.  
The server may need to be restarted for these changes to take effect.

WebSphere JMS Provider > WebSphere Topic Destinations >  
**Topic1**

Topic destinations provided for publish/subscribe messaging by the internal WebSphere JMS provider. Use WebSphere Topic Destination administrative objects to manage topic destinations for the internal WebSphere JMS provider.

**Configuration**

**General Properties**

Scope	* cells:LocalhostX260.nodes:LocalhostX260	<input type="checkbox"/> The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* Topic1	<input type="checkbox"/> The required display name for the resource.
JNDI Name	* Topic1	<input type="checkbox"/> The JNDI name for the resource.
Description		<input type="checkbox"/> An optional description for the resource.
Category		<input type="checkbox"/> An optional category string which can be used to classify or group the resource.
Topic	* Topic1	<input type="checkbox"/> This is the string value used to identify the Topic. It can be dot notation and include wildcard characters.
Persistence	APPLICATION DEFINED	<input type="checkbox"/> Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.
Priority	APPLICATION DEFINED	<input type="checkbox"/> Whether the message priority for this destination is defined by the application or the Specified priority property.
Specified Priority		<input type="checkbox"/> When priority is SPECIFIED, this value specifies the priority for the topic. Valid values are in the range 0-9 with 0 as the lowest priority and 9 as the highest priority.
Expiry	APPLICATION DEFINED	<input type="checkbox"/> Whether the expiry timeout for this queue is defined by the application or the Specified expiry property, or messages on the queue never expire (have an unlimited expiry timeout).
Specified Expiry		<input type="checkbox"/> When expiry is SPECIFIED, this value contains the expiration period for the Topic in milliseconds. Valid values are any long value greater than zero.

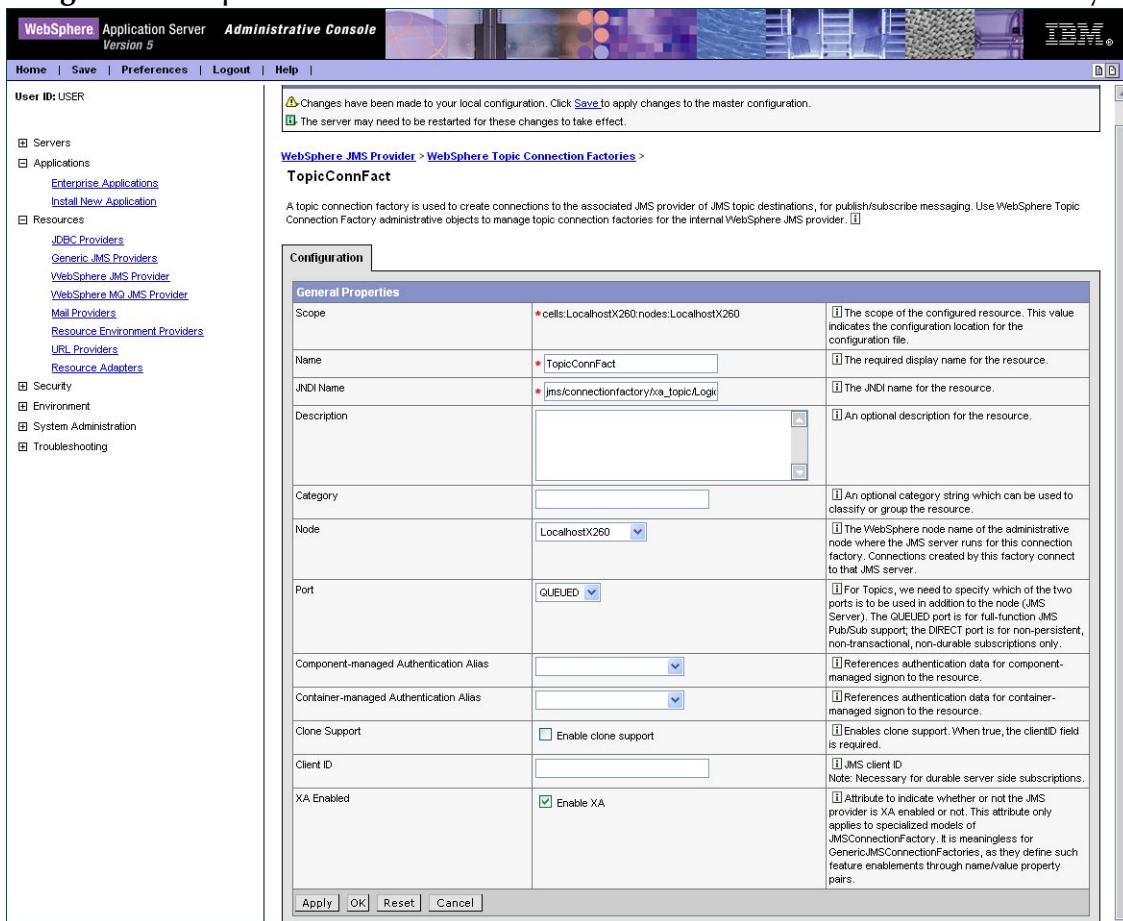
Apply OK Reset Cancel

- From the **WebSphere Topic Destinations** window, click **WebSphere JMS Provider** to return to the JMS Provider window.

## Creating a Connection Factory

- 1 From the **WebSphere JMS Provider** window, under **Additional Properties**, click **WebSphere Topic Connection Factories** (or Queue Connection Factories if you created a Queue).
- 2 From the **WebSphere Topic Connection Factories** window, click the **New** button.
- 3 From the **WebSphere Topic Connection Factories** configuration window, enter the Topic Connection Factory name in the **Name** field (for this example, **TopicConnFact**).
- 4 Enter the Connection Factory JNDI Name in the **JNDI Name** field. The pattern for the JNDI Name is **jms/connectionfactory/<xa\_topic or xa\_queue (just topic or queue if xa is not selected as the Transaction Mode)>/<Logical Host name>\_<JMS Message Server name (from the Environment)>**.  
  
For this example, the Connection Factory JNDI name is **jms/connectionfactory/xa\_topic/LogicalHost1\_WLMMessageSvr1**
- 5 Make sure that **XA Enabled** is selected (see Figure 4). Click **OK**.

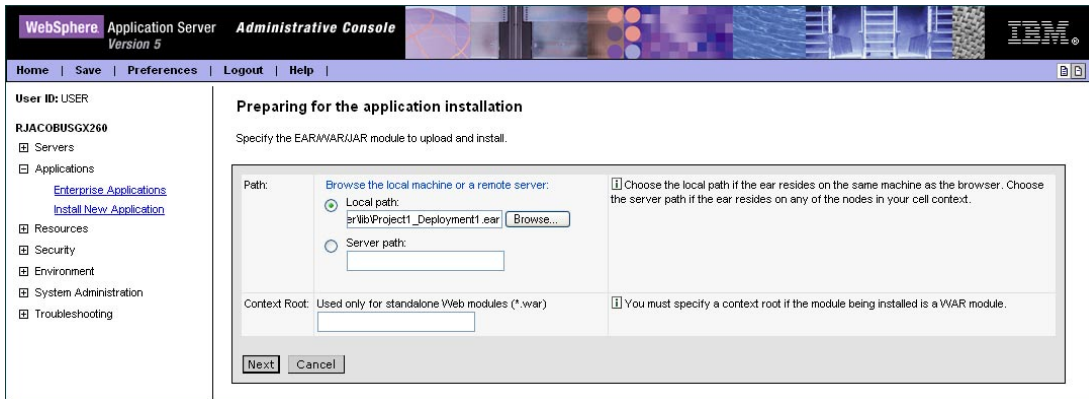
**Figure 4** WebSphere Server Administrative Console - Create Connection Factory



## Installing the Application

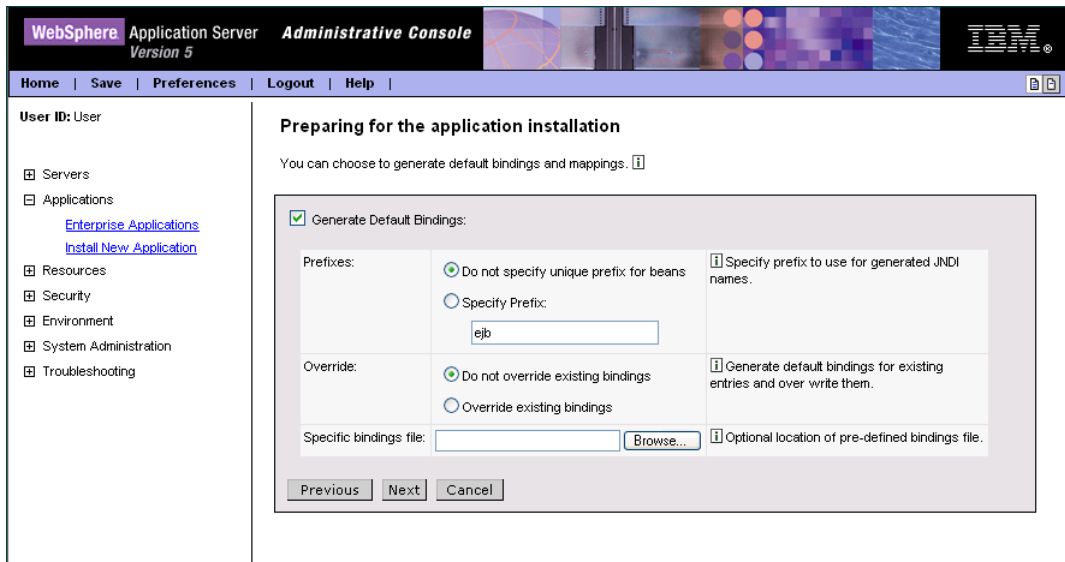
- 1 From the left pane of the WebSphere Administrative console, select **Applications**, and click **Install New Application**.
- 2 From the **Preparing for the application installation** window, select **Local path** or **Server path** and click **Browse**. Locate and select the appropriate EAR file (see Figure 5). Click **Next**.

**Figure 5** WebSphere Server Administrative Console - Install New Application



- 3 From the next window select **Generate Default Bindings** (see Figure 6). Click **Next**.

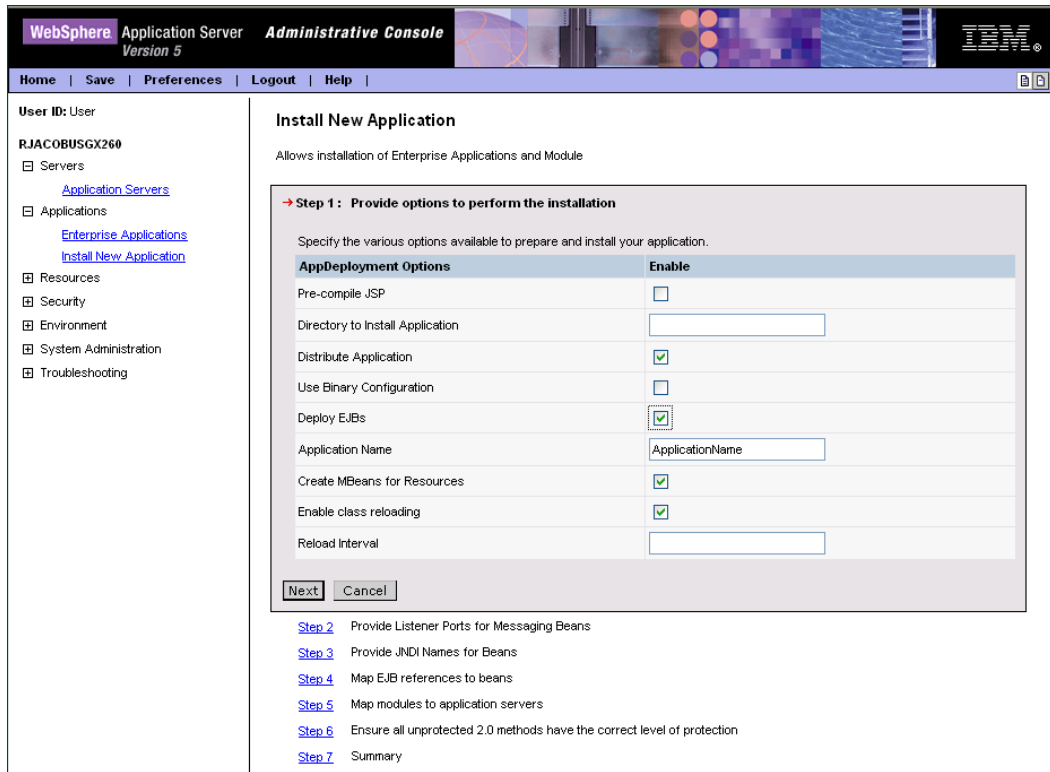
**Figure 6** WebSphere Server Administrative Console - Generate Default Bindings



**Note:** The following application installation steps may differ depending on the nature of the EAR file.

- 4 From the **Step 1: Provide options to perform the installation** window (see Figure 7), enter a name for the application and select **Deploy EJBs** and **Enable Class Reloading** (see Figure 7 on page 71). Click **Next**.

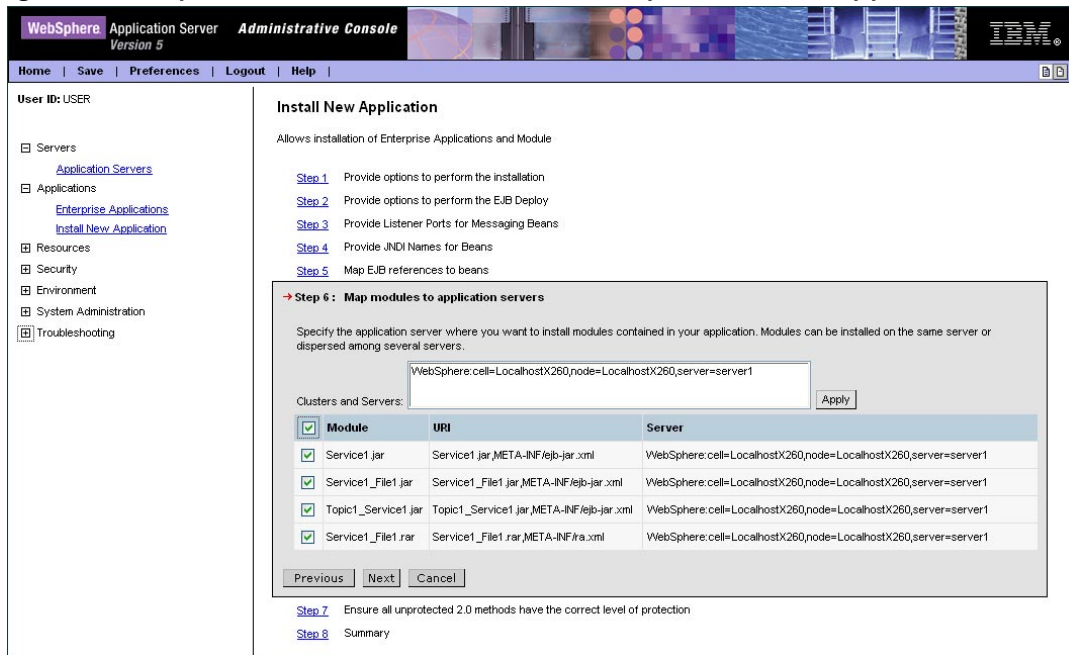
**Figure 7** WebSphere Administrative Console - Provide options to perform the installation



- 5 From the **Step 3: Provide Listener Ports for Messaging Beans** window, make a note of the name that you used for the Listener Port. This Listener Port name will be used later in step 12. Proceed to the **Step 6: Map modules to application servers** window.
- 6 From the **Step 6: Map modules to application servers** window, make a note of the Reference Binding field value (save this name as it appears). This name will be used in step 12-G.
- 7 From the **Step 6: Map modules to application servers** window, click the Module checkbox to select all of the modules. (see [Figure 8 on page 72](#)). Click **Next**.



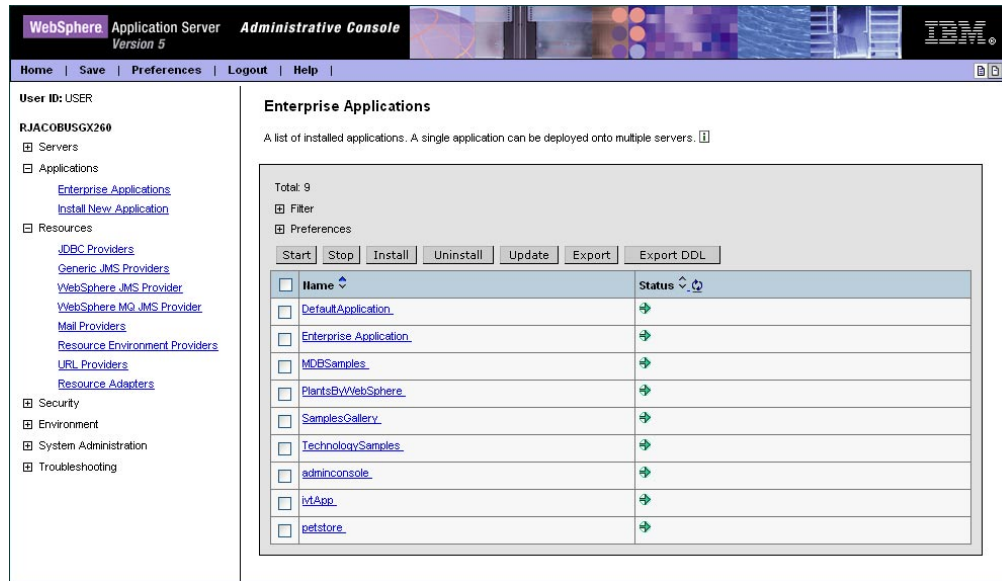
**Figure 8** WebSphere Administrative Console - Map Modules to Application Servers



- 8 From the **Step 7: Ensure all unprotected 2.0 methods have the correct level of protection** window, click the Module checkbox to select all of the modules. Click **Next**.
- 9 From the **Step 8: Summary** window, review the selected values and click **Finish**.
- 10 WebSphere begins installing the application. This can take several minutes. When the application installs successfully, click **Save to Master Configuration**.
- 11 To review the application's status or start or stop the application, from the right pane of the Administrative Console, click **Applications**, and click **Enterprise Applications** (see Figure 9).



**Figure 9** WebSphere Administrative Console - Enterprise Application



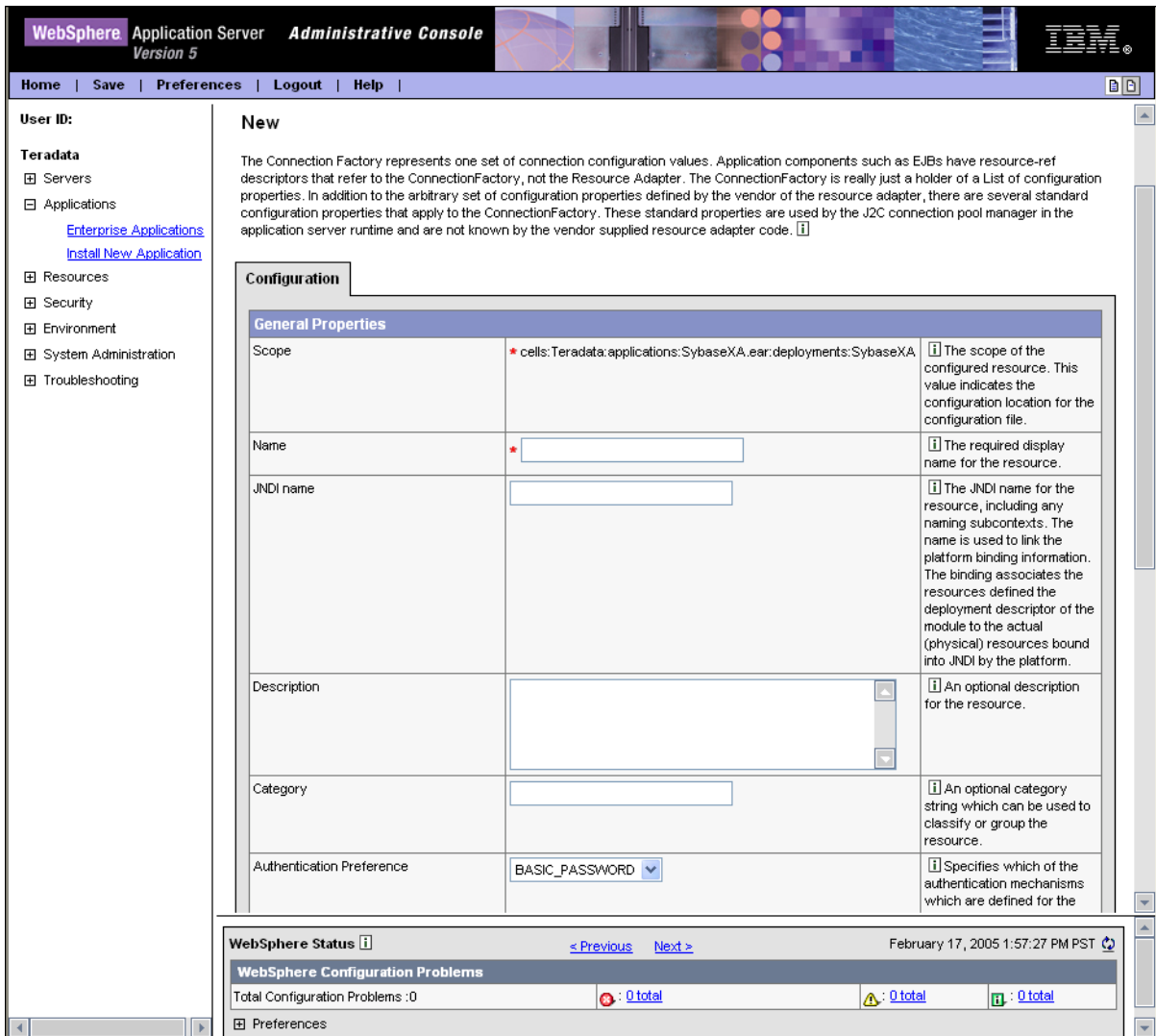
- 12 Complete the following steps to configure the J2C Connection Factories:
  - A From the Enterprise Applications window, select your application.
  - B From the Related Items box, click **Connector Modules**.
  - C From the Connector Modules window, click the deployed RAR file.
  - D From the deployed RAR file window's **Additional Properties** box, click **Resource Adapter** (see Figure 58).

**Figure 10** WebSphere Administrative Console - Resource Adapter

Additional Properties	
<b>Resource Adapter</b>	A resource adapter is created for every connector module in the application as a part of application installation. This resource adapter can be used to further create Connection Factories that can be used to bind resources defined in the application.
<a href="#">View Deployment Descriptor</a>	View the Deployment Descriptor

- E From the Resource Adapter window, Additional Properties box, click **J2C Connection Factories**. The J2C Connection Factories window appears.
- F Click the **New** button to create a new **J2C Connection Factory** (see [Figure 11 on page 74](#)).

Figure 11 WebSphere Administrative Console - J2C Con




- G From the **J2C Connection Factory's New** window, do the following:
  - ◆ Enter the **Name** for the new J2C Connction Factory.
  - ◆ Enter the **JNDI Name** for the new J2C Connction Factory. This is the same name that you saved in **step 6 on page 71** (This name can also be found as the **res-ref-name** in the **ejb-jar.xml** file)
  - ◆ Set **Mapping-Configuration Alias** to **DefaultPrincipleMapping**
- H Click **OK** to create the new J2C Connection Factory..










- 13 Create a Listener Port by completing the following steps:
  - A From the left pane of the WebSphere Application Server Administrative Console, expand Servers and click **Application Servers**.
  - B From Application Servers click **server1**.
  - C From the Additional Properties box, of the server1 window, click **Message Listener Service**.
  - D From the Additional Properties box, of the Message Listener Service window, click **Listener Ports**.
  - E From the Listener Ports window click **New** to create a new Listener Port.
  - F In the Configuration Tab, of the new Listener Port being created, fill in the following required fields (see Figure 58):
    - ♦ **Name** (Use the name of the Listener Port from Step 5)
    - ♦ **Initial State**
    - ♦ **Connection Factory JNDI Name**
    - ♦ **Destination JNDI Name**
  - G Click **OK** to create the Listener Port.

**Figure 12** WebSphere Administrative Console - Creating a Listener Port

[Application Servers](#) > [server1](#) > [Message Listener Service](#) > [Listener Ports](#) >

**New**

Listener ports for Message Driven Beans to listen upon for messages. Each port specifies the JMS Connection Factory and JMS Destination that an MDB, deployed against that port, will listen upon. 

Runtime		Configuration
<b>General Properties</b>		
Name	* Topic1_jms_collabPort	 Name of the listener port
Initial State	* Started 	 The execution state requested when the server is first started.
Description		 A description of the listener port, for administrative purposes
Connection factory JNDI name	* jms/connectionfactory/xa_topicLogi	 The JNDI name for the JMS connection factory to be used by the listener port, for example, jms/connFactory1.
Destination JNDI name	* Topic1	 The JNDI name for the destination to be used by the listener port, for example, jms/destn1.
Maximum sessions	1	 The maximum number of concurrent JMS server sessions used by a listener to process messages, in the range 1 through 2147483647.
Maximum retries	5	 The maximum number of times that the listener tries to deliver a message before the listener is stopped, in the range 0 through 2147483647.
Maximum messages	1	 The maximum number of messages that the listener can process in one JMS server session, in the range 0 through 2147483647.
<input type="button" value="Apply"/> <input type="button" value="OK"/> <input type="button" value="Reset"/> <input type="button" value="Cancel"/>		

# Index

## A

Add Prepared Statements 35

## B

book 65

## C

ClassName 15  
Connect to Database 28

## D

DatabaseName 18, 21, 25  
DataSourceName 25  
Delimiter 18, 26  
Description 15, 18, 26  
driver class, JDBC 15  
DriverProperties 19, 26

## E

EAR file  
    deploying an EAR file 67  
Environment Properties  
    DatabaseName 18  
    Delimiter 18  
    Description 18  
    DriverProperties 19  
    Password 19  
    User 19

## I

Inbound Environment Properties  
    Database 21  
    Password 21  
    User 21  
index  
    book 65  
InitialPoolSize 15

## J

JDBC  
    driver class 15

## L

LoginTimeOut 16

## M

MaxIdleTime 16  
MaxPoolSize 16  
MaxStatements 16  
MinPoolSize 16

## N

NetworkProtocol 17

## O

Outbound Environment Properties  
    DatabaseName 25  
    DataSourceName 25  
    Delimiter 26  
    Description 26  
    DriverProperties 26  
    Password 26  
    User 26  
Outbound Properties  
    ClassName 15  
    Description 15  
    InitialPoolSize 15  
    LoginTimeOut 16  
    MaxIdleTime 16  
    MaxPoolSize 16  
    MaxStatements 16  
    MinPoolSize 16  
    NetworkProtocol 17  
    PropertyCycle 17  
    RoleName 17

## P

Password 19, 21, 26  
Property settings, Environment  
    DatabaseName 18  
    Delimiter 18  
    Description 18  
    DriverProperties 19  
    Password 19  
    User 19  
Property settings, Inbound Environment

## Index

- Database 21
- Password 21
- User 21
- Property settings, Outbound
  - ClassName 15
  - Description 15
  - InitialPoolSize 15
  - LoginTimeOut 16
  - MaxIdleTime 16
  - MaxPoolSize 16
  - MaxStatements 16
  - MinPoolSize 16
  - NetworkProtocol 17
  - PropertyCycle 17
  - RoleName 17
- Property settings, Outbound Environment
  - DatabaseName 25
  - DataSourceName 25
  - Delimiter 26
  - Description 26
  - DriverProperties 26
  - Password 26
  - User 26
- PropertyCycle 17

## R

- RoleName 17

## S

- Scope
  - ADABAS Natural eWay 8
- Select Database Objects 29
- Select Procedures 32
- Select Table/Views 29
- Select Wizard Type 27
- Specify the OTD Name 36
- System Requirements 10

## U

- User 19, 21, 26

## W

- WebSphere Application Server
  - support 7, 65