SeeBeyond ICAN Suite

# Oracle eWay Intelligent Adapter User's Guide

Release 5.0.4



SeeBeyond Proprietary and Confidential

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.

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 This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence. Version 20040708132149.

# Contents

Chapter 1	
Introducing the Oracle eWay	6
Overview	6
Supported Operating Systems WebLogic and WebSphere Application Server Support	6 7
System Requirements	7
External System Requirements	8
XA Requirements	8
Chapter 2	
Installing the Oracle eWay	9
Before Installing the eWay	9
Installing the Oracle eWay	9
After Installation To Load using Enterprise Designer—Using 5.0.3	<b>10</b> 10

To Copy Directly into the Logical Host–Using 5.0.1 or 5.0.2

# Chapter 3

Properties of the Oracle eWay	12
Working with eWay Property Sheets	12
Setting the Properties in the Outbound eWay	12
ClassName	13
Description	13
InitialPoolSize	14
LoginTimeOut	14
MaxIdleTime	14
MaxPoolSize	14
MaxStatements	14
MinPoolSize	15
NetworkProtocol	15
PropertyCycle	15
RoleName	15
Setting the Properties in the Inbound eWay	16

11

Contents

Pollmilliseconds	
PreparedStatement	
Setting the Properties in the Outbound eWay Environment	
DatabaseName	
DataSourceName	
Delimiter	
Description	
DriverProperties	
Password	
PortNumber	
ServerName	
TNSEntry	
User	
Setting the Properties in the Inbound eWay Environment	
DatabaseName	
Password	
PortNumber	
ServerName	
User	
Setting the Properties in the Outbound eWay with XA Support	
ClassName	
Description	
InitialPoolSize	
LoginTimeOut	
MaxIdleTime	
MaxPoolSize	
MaxStatements	
MinPoolSize	
NetworkProtocol	
PropertyCycle	
RoleName	
Setting the Properties in the eWay XA Environment	
DatabaseName	
DataSourceName	
Delimiter	
Description	
DriverProperties	
Password	
PortNumber	
ServerName	
TNSEntry	
User	

Using the Oracle eWay Database Wizard	28
Using the Database OTD Wizard	28

# Chapter 5

Importing the Project	40
eInsight Engine and eGate Components	40
Using the Sample Project in elnsight	40
The Business Process	41
Working with the BPEL Operations	42
whereClause()	43
SelectAll	43
SelectMultiple	45
SelectOne	47
Insert Update	48 49
Delete	51
	52
Using the Sample Project in eGate Working with the Sample Project in eGate	52
Configuring the eWays	53
Creating Rules within the Collaboration	54
Creating the External Evironment	60
Deploying a Project	60
Running the Sample	60
Supported Data Types	61
Converting Data Types in the Oracle eWay	62
Using OTDs with Tables, Views, and Stored Procedures	63
Data Types	64
The Table	64
The Query Operation	64
The Insert Operation	65
The Update Operation	66
The Delete Operation Using CLOBs	66 67
The Stored Procedure	0/ 71
Executing Stored Procedures	71
Manipulating the ResultSet and Update Count Returned by Stored Procedure	73
Creating or Editing an OTD from an Existing OTD	75
Editing an Existing OTD	75
Creating a New OTD from an Existing OTD	76
Using XA	76
Alerting and Logging	76
Index	77

# Chapter 1

# Introducing the Oracle eWay

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

This Chapter Includes:

- Overview on page 6
- Supported Operating Systems on page 6
- System Requirements on page 7
- External System Requirements on page 8

# 1.1 **Overview**

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

# **1.2 Supported Operating Systems**

The Oracle eWay is available on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- HP Tru64 5.1A
- HP-UX 11.0, 11i (PA-RISC), and 11i v2.0 (11.23)
- IBM AIX 5.1L and 5.2
- Red Hat Enterprise Linux AS 2.1 (Intel x86)
- Red Hat Linux 8 (Intel x86)
- Sun Solaris 8 and 9
- SuSE Linux Enterprise Server 8 (Intel x86)
- Japanese Windows 2000, Windows XP, and Windows Server 2003
- Japanese HP-UX 11.0, 11i (PA-RISC), and 11i v2.0 (11.23)

- Japanese IBM AIX 5.1L and 5.2
- Japanese Sun Solaris 8 and 9
- Korean Windows 2000, Windows XP, and Windows Server 2003
- Korean HP-UX 11.0, 11i (PA-RISC), and 11i v2.0 (11.23)
- Korean IBM AIX 5.1L and 5.2
- Korean Sun Solaris 8 and 9

Although the Oracle Universal Database eWay, the Repository, and Logical Hosts run on the platforms listed above, the Enterprise Designer requires the Windows operating system. Enterprise Manager can run on any platform that supports Internet Explorer 6.0.

# 1.2.1 WebLogic and WebSphere Application Server Support

In addition to the Operating Systems listed above, this eWay is also supported on the following application servers:

- WebSphere Application Server, version 5.0
- WebLogic Application Server, version 8.1

These are limited to outbound mode using Java Collaborations. For additional information, see the *eGate Integrator User's Guide*.

# **1.3** System Requirements

The system requirements for the Oracle 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.

The Oracle eWay installation installs the type 4 Oracle JDBC 9.0.1.3.0 driver required to connect to the external Oracle database.

To use the OCI driver with the Oracle eWay, you will need to install the Oracle client on the logical host. Because the OCI driver supports both the most current Oracle database and any previous versions, it is recommend that you use the most current version of the OCI driver. By doing this, you can assure compatibility between the various versions of any Oracle databases you may have.

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

# **1.4 External System Requirements**

The Oracle eWay supports the following software for external systems:

- Oracle version 8i with patch 8.1.7
- Oracle version 9i with patch 9.0.1.3
- Oracle version 9i release 9.2.0
- Oracle 10g

# 1.5 XA Requirements

When using XA with multiple Oracle databases, it is recommended that you use the latest version of the database.

# Chapter 2

# Installing the Oracle eWay

This chapter describes how to install the Oracle eWay.

This Chapter Includes:

- Before Installing the eWay on page 9
- Installing the Oracle eWay on page 9
- After Installation on page 10

# 2.1 Before Installing the eWay

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

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

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

Continue installing the eGate Enterprise Designer as instructed.

# 2.3 After Installation

If you choose to use the OCI driver, you will need to copy the **classes12.jar** file from the Oracle client to the Logical Host.

Oracle 10g users must copy the **ojdbc14.jar** file from the Oracle client, and rename it to the **classes12.jar** file before copying into the Logical Host.

*Note:* Be sure to create a back-up of the original classes12.jar file before applying these changes.

Ways to load the classes12.jar file include:

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

# To Load using Enterprise Designer-Using 5.0.3

1 Right-click the Logical Host on the Environment tab, and select **Upload File** on the menu.

If you are running on Unix, you must first copy the Oracle driver (classes12.jar or ojdbc14.jar) from the Oracle client on your Unix machine, to a location on Windows.

2 On the Upload Third Party Files window, click **Add** and select the **classes12.jar** from the Oracle client installation directory. At run time, the **classes12.jar** file is loaded into the following directory:

<egate\_logicalhost>:\\ICAN50\logicalhost\stcis\lib\

Failing to follow the above will result in the eWay failing to recognize your Oracle database.

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.

# To Copy Directly into the Logical Host–Using 5.0.1 or 5.0.2

1 Copy the **classes12.jar** file to:

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

Replacing the **classes12.jar** file already in this location with Oracle's **classes12.jar** file.

- *Note: The <egate\_logicalhost>:\\stcis\lib location only appears after running a project.*
- *Note:* The Oracle eWay only supports the OCI driver for run-time.

# Chapter 3

# **Properties of the Oracle eWay**

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

This Chapter Includes:

- Setting the Properties in the Outbound eWay on page 12
- Setting the Properties in the Inbound eWay on page 16
- Setting the Properties in the Outbound eWay Environment on page 17
- Setting the Properties in the Inbound eWay Environment on page 20
- Setting the Properties in the Outbound eWay with XA Support on page 22
- Setting the Properties in the eWay XA Environment 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 DataSource settings define the properties used to interact with the external database.

Properties		
Configuration	* 12 18 1= 1	
DBC Connector settings	ClassName	oracle.jdbc.pool.OracleConnectionPoolDataSource
	Description	Oracle thin driver Connection Pool Datasource.
	InitialPoolSize	2
	LoginTimeOut	0
	MaxIdleTime	0
	MaxPoolSize	10
1	MaxStatements	1000
	MinPoolSize	2
	NetworkProtocol	
1	PropertyCycle	0
	RoleName	
Description (JDBCConnectorSetti JDBC Connector settings Comments (JDBCConnectorSetti		
	Properties	
ОК		Cancel

# Figure 1 The Outbound eWay Properties

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

# ClassName

#### Description

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

#### **Required Values**

A valid class name.

The default is **oracle.jdbc.pool.OracleConnectionPoolDataSource**.

# Description

#### Description

Enter a description for the database.

#### **Required Value**

# 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 may remain unused before it is closed. 0 (zero) indicates that there is no limit.

#### **Required Value**

A valid numeric value.

#### MaxPoolSize

#### Description

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

#### **Required Value**

A valid numeric value.

#### **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**

# 3.1.2. Setting the Properties in the Inbound eWay

Properties 🗙			
Configuration			
Parameter Settings	PollMilliseconds 5000		
	PreparedStatement		
Description (parameter-settings)			
Comments (parameter-settings)			
Comments (parameter-settings)			
	l		
	Properties		
ОК	Cancel		

Figure 2 Properties of the Inbound eWay

# Pollmilliseconds

#### Description

Polling interval in milliseconds.

#### **Required Value**

A valid numeric value. The default is 5000.

#### PreparedStatement

#### Description

Prepared Statement used for polling against the database.

#### **Required Value**

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

# 3.1.3. Setting the Properties in the Outbound eWay Environment

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

To assure a successful database connection when using different Oracle drivers, be sure to *only* fill in the following fields:

To use the Oracle thin driver:

- DatabaseName
- PortNumber
- ServerName
- User
- Password

To use the Oracle OCI driver:

- TNSEntry
- User
- Password
- PortNumber not needed, but keep the default

#### Figure 3 eWay Environment Configuration

Properties 🛛 🗙			
Environment Configuration	* 12 18 1/- 1		
UDBC Connector settings	DatabaseName		
	DataSourceName		
	Delimiter	#	
	Description	Oracle thin driver Connection F	
	DriverProperties		
	Password		
	PortNumber	1521	
	ServerName		
	TNSEntry		
	User		
Description (JDBCConnectorSetti			
JDBC Connector settings			
:			
Comments (JDBCConnectorSetti			
· · · · · · · · · · · · · · · · · · ·			
	Properties		
ОК		Cancel	
		Cancer	

# DatabaseName

#### Description

Specifies the name of the database instance used on the Server.

#### **Required Values**

Any valid string.

# DataSourceName

#### Description

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

#### **Required Value**

Optional. In most cases, leave this box empty.

# Delimiter

#### Description

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

#### **Required Value**

The default is #.

# Description

#### Description

Enter a description for the database.

#### **Required Value**

A valid string.

# **DriverProperties**

#### Description

Use the JDBC driver that is shipped with this eWay. Often times the DataSource implementation will need to execute additional properties to assure a connection. The additional methods will need to be identified in the Driver Properties.

#### **Required Value**

Any valid delimiter.

Valid delimiters are: "<method-name-1>#<param-1>#<param-2>##......<param-n>##<method-name-2>#<param-1>#<param-2>#......<param-n>##......##".

For example: to execute the method setURL, give the method a String for the URL "setURL#<url>##".

# Password

#### Description

Specifies the password used to access the database.

#### **Required Values**

Any valid string.

### PortNumber

#### Description

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

#### **Required Values**

A valid port number. The default is 1521.

#### ServerName

#### Description

Specifies the host name of the external database server.

#### **Required Values**

Any valid string.

# **TNSEntry**

#### Description

Specifies the TNS name for the Oracle instance specified in TNSNAMES.ORA. If a TNS name is specified, then the OCI driver is used, which further requires installation of the Oracle client. If a TNS name is not specified, then the thin driver is used.

#### **Required Values**

Any valid string.

#### User

#### Description

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

#### **Required Values**

# 3.1.4. Setting the Properties in the Inbound eWay Environment

🖏 Properties		
Environment Configuration	¥12 18 1=	
Parameter Settings	DatabaseName	
	Password	
	PortNumber	1521
	ServerName	
	User	
Description (parameter-settings)		
Comments (parameter-settings)		
	Properties	
	Criopentes	
ОК		Cancel

Figure 4 Inbound eWay Environment

# DatabaseName

#### Description

Specifies the name of the database instance used on the Server.

#### **Required Values**

Any valid string.

#### Password

#### Description

Specifies the password used to access the database.

#### **Required Values**

# PortNumber

#### Description

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

#### **Required Values**

A valid port number. The default is 1433.

#### ServerName

#### Description

Specifies the host name of the external database server.

#### **Required Values**

Any valid string.

#### User

#### Description

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

#### **Required Values**

# 3.1.5. Setting the Properties in the Outbound eWay with XA Support

Properties Configuration	¥↓2↓® ⊭=	
DBC Connector settings	ClassName	oracle.jdbc.xa.client.OracleXADataSource
	Description	Oracle thin driver XA Datasource
	InitialPoolSize	2
	LoginTimeOut	0
	MaxIdleTime	0
	MaxPoolSize	10
	MaxStatements	1000
	MinPoolSize	2
	NetworkProtocol	
	PropertyCycle	0
	RoleName	
Description (JDBCConnectorSetti JDBC Connector settings		
	Properties	
ОК		Cancel

Figure 5 Outbound Oracle eWay with XA Support

# ClassName

#### Description

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

#### **Required Values**

A valid class name.

The default is oracle.jdbc.xa.client.OracleXADataSource.

# Description

#### Description

Enter a description for the database.

#### **Required Value**

# InitialPoolSize

#### Description

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

#### **Required Value**

A valid numeric value. The default is 2.

# LoginTimeOut

#### Description

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

#### **Required Value**

A valid numeric value.

# MaxIdleTime

#### Description

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

#### **Required Value**

A valid numeric value.

#### MaxPoolSize

#### Description

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

#### **Required Value**

A valid numeric value. The default is 10.

# MaxStatements

#### Description

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

#### **Required Value**

A valid numeric value. The default is 1000.

# MinPoolSize

The minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and the new connections should be created as needed.

#### **Required Value**

A valid numeric value.

# NetworkProtocol

#### Description

The network protocol used to communicate with the server.

#### **Required Values**

Any valid string. The default is 2.

# PropertyCycle

#### Description

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

#### **Required Values**

A valid numeric value.

# RoleName

#### Description

An initial SQL role name.

#### **Required Values**

# 3.1.6. Setting the Properties in the eWay XA Environment

Properties 🗴			
Environment Configuration	* 12 18 1/- 1		
UDBC Connector settings	DatabaseName		
	DataSourceName		
	Delimiter	#	
	Description	Oracle thin driver XA Datasourd	
	DriverProperties		
	Password		
	PortNumber	1521	
	ServerName		
	TNSEntry		
	User		
Description (JDBCConnectorSetti			
JDBC Connector settings			
;			
Comments (JDBCConnectorSetti			
	Properties		
ОК		Cancel	

Figure 6 Environment Properties of the eWay with XA

# DatabaseName

#### Description

Specifies the name of the database instance used on the Server.

#### **Required Values**

Any valid string.

# DataSourceName

#### Description

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

#### **Required Value**

Optional. In most cases, leave this box empty.

# Delimiter

#### Description

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

#### **Required Value**

The default is #.

# Description

#### Description

Enter a description for the database.

#### **Required Value**

A valid string.

# **DriverProperties**

#### Description

Use the JDBC driver that is shipped with this 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.

# PortNumber

#### Description

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

#### **Required Values**

A valid port number. The default is 1521.

# ServerName

#### Description

Specifies the host name of the external database server.

#### **Required Values**

Any valid string.

# **TNSEntry**

#### Description

Specifies the TNS name for the Oracle instance specified in TNSNAMES.ORA. If a TNS name is specified, then the OCI driver is used, which further requires installation of the Oracle client. If a TNS name is not specified, then the thin driver is used.

#### **Required Values**

A valid TNS name if using the OCI driver; otherwise do not enter any value.

### User

#### Description

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

#### **Required Values**

# **Chapter 4**

# Using the Oracle eWay Database Wizard

This chapter describes how to use the Oracle eWay Database Wizard to build OTD's. **This Chapter Includes:** 

- Select Wizard Type on page 28
- Connect to Database on page 29
- Select Database Objects on page 30
- Select Table/Views on page 30
- Select Procedures on page 34
- Add Prepared Statements on page 37
- Specify the OTD Name on page 38

# 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.

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

*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. 2 From the OTD Wizard Selection window, select the **Oracle Database** and click **Next**. See **Figure 7**.

New Object Type Definition Wizard 🛛 🛞		
	Select Wizard Type	
	OTD Wizard	Description
	DB2 Database	Uses a UDB database to create an OTD
	CTD	Uses a DTD to create an OTD
	Image: Second	Creates an OTD from NSSQL Database (
and the second sec	Oracle Database	Uses an Oracle database to create an OT
THE PARTY NEWS	📲 🖫 Sqlserver Database	Uses a SQLServer database to create an
V AF	📲 Sybase Database	Uses a Sybase database to create an OT
MINE AND AND	User-Defined OTD	Wizard for creating User-Defined OTDs
	TE WSDL	Wizard for creating WSDL OTD
H-111	T XSD	Uses an XSD to create an OTD
SEE BEYOND"	د	
	< Back Next >	Einish Cancel Help

#### **Figure 7** OTD Wizard Selection

#### **Connect to Database**

3 Specify the connection information for your database including your UserName and Password and click Next. See Figure 8.

Figure 8 Database Connection Info	rmation

		New Wizard - Oracle Database	*
<b>Ste</b> 1. 2. 3. 4. 5. 6. 7. 8.	Select Wizard Type Connect to Database Select Database Objects Select Trocedures Add Prepared Statements Specify the OTD Name Review Selections	Specify Database Connection Information         Please enter the Oracle database connection information below.         Connection Information         Host name:         Port ID:       1521         SID:         User name:         Password:	
		<back next=""> Einish Cancel Help</back>	

#### Select Database Objects

In the Select Database Objects window, see Figure 9, select Tables/Views for this sample. 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.

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

Figure 9 Select Database Objects

Steps	Select Database Objects
<ol> <li>Select Wizard Type</li> <li>Connect to Database</li> <li>Select Database Objects</li> <li>Select Tables/Views</li> <li>Select Procedures</li> <li>Add Prepared Statements</li> <li>Specify the OTD Name</li> <li>Review Selections</li> </ol>	Which of the following database objects would you like to include in your OTD? (Select at least one) I tables/Views Procedures Prepared Statements
SEEBEYOND	Connection established: Host: enigma:1521 SID: ore18

Select Table/Views

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

Steps	Select Tables Views and specify Column information
1. OTD Wizard Selection 2. Connect to Database	_Selected Tables/Views
2. Connect to Database     3. Select Database Objects     4. Select Tables/Views     5. Select Procedures     6. Add Prepared Statements     7. Specify the OTD Name     8. Review Selections	Name     Catalog     Schema     Type
SEEBEYOND	Description Change Change Use fully-qualified table/view names in the generated Java code.

#### 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.
- <sup>3</sup> 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.

ype the exact Table/	√iew Name or use	Wildcard characte	ers.	
ame:		DGDB		Search
<u>T</u> ables Only ○ ⊻i	iews Only 🛛 🔘 <u>B</u> oth	n 🗹 İnclude syste	m tables	
able/View Selection-				
esults: 225 Records				
Name	Catalog	Schema	Тур	e
B_EMPLOYE		DGDB	TABLE	-
B_EMPLOYEE		DGDB	TABLE	
B_EMPLOYEE1		DGDB	TABLE	
B_EMPLOYEE10		DGDB	TABLE	
B_EMPLOYEE2		DGDB	TABLE	
Select				
elected: 0 Records	Catalog	Schema	Т	vpe
DB EMPLOYEE	Catalog	DGDB	TABLE	100

#### Figure 11 Database Wizard - All Schemes

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

The table selected is added to the **Selected Tables/Views** window. See Figure 12.



Steps	Select Tables/Views a	niu specny (	Column morn	Iduuri
OTD Wizard Selection     Connect to Database	Selected Tables/Viev	vs		
3. Select Database Objects	Name	Catalog	Schema	Type
Select Tables Views	DB_EMPLOYEE		DGDB	TABLE
. Select Procedures				
. Add Prepared Statements				
. Specify the OTD Name				
. Review Selections				
. Review Selections				
. Review Selections				
. Review Selections			<u>n</u>	
. Review Selections	Add	Remove		
. Review Selections		Remove		
. Review Selections	Add	<u>R</u> emove	- 	
. Review Selections	Description DB_EMPLOYEE:	Remove		Channe
. Review Selections	Description	Remove		<u>C</u> hange
. Review Selections	Description DB_EMPLOYEE: 5 columns selected.		. 	(
SEEBEYOND"	Description DB_EMPLOYEE:		es in the genera	(

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

	Table/View (	Columns 🛛 🛞
Columns I	For: DB_EMPLOYEE	
-Column S	Selection	
	Column Name	Туре
	EMP_NO	DECIMAL
	LAST_NAME	VARCHAR
<b>I</b>	FIRST_NAME	VARCHAR
	RATE	OTHER
<b>V</b>	LAST_UPDATE	TIMESTAMP
<u>S</u> e	lect All Invert Selection	
	able column	Advanced
	······	

#### Figure 13 Table/View Columns

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

umn S	election			
	Column	SQL type	Precision / len	Scale
	EMP_NO	DECIMAL	38	0
	LAST_NAME	VARCHAR	30	0
	FIRST_NAME	VARCHAR	30	0
	RATE	OTHER	126	0
	LAST_DATE	TIMESTAMP	7	0
	ect All Inv	vert Selection	Ciear All	

8 When using Oracle packages, select **Use fully qualified table/view names in the** generated Java code. See Figure 12.

#### **Select Procedures**

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

Figure 15 Select Procedures and specify Resultset and Parameter Information

Steps           1.         Select Wizard Type	Select Procedure		Tubbulkoot unu		
Select Wizard Type     Connect to Database	Selected Proced	ettere)	Schema	Ture	-
Select Database Objects     Select Tables/Views     Select Procedures     Add Prepared Statements     Specify the OTD Name	DB_EMPLOYEE	Catalog	DGDB	PROCEDURE	
8. Review Selections	 Description DB_EMPLOYEE				
	4 parameter(s)		6	Edit Parameters.	
191	0 resultset(s)			Edit <u>R</u> esultsets.	)
SEEBEYOND	🗌 Use fully-gualifi	ed procedure n	ames in the gener	ated Java code.	

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

#### Figure 16 Add Procedures

ype the exact Procedure Name or use Wildcard characters. Iame: DGDB ▼ Search Procedure Selection Acesults:246 Records Name Catalog Schema Type COPY_DB_EMPLOYEE_TO_DB_E DGDB PROCED ► DB_AVG_RATE DGDB PROCED ► DB_EMPLOYEE_INSERT DGDB PROCED ► DB_EMPLOYEE_PROC DGDB PROCED ► DB_EMPLOYEE_PROC2 DGDB PROCED ▼ Select Name Catalog Schema Type		Add P	rocedures		
rocedure Selection tesults:246 Records  Name Catalog Schema Type COPY_DB_EMPLOYEE_TO_DB_E DGDB PROCED DB_AV0_RATE DGDB PROCED DB_EMPLOYEE_INSERT DGDB PROCED DB_EMPLOYEE_PROC DGDB PROCED DB_EMPLOYEE_PROC2 DGDB PROCED Select Name Catalog Schema Type		e Name or use Wildo		s.	
Results:246 Records       Name     Catalog     Schema     Type       COPY_DB_EMPLOYEE_TO_DB_E     DGDB     PROCED     PROCED       DB_AVG_RATE     DGDB     PROCED     PROCED       DB_EMPLOYEE_INSERT     DGDB     PROCED     PROCED       DB_EMPLOYEE_PROC     DGDB     PROCED     PROCED       DB_EMPLOYEE_PROC2     DGDB     PROCED     V       Select     Name     Catalog     Schema     Type	Name:		DGDB		Search
Results:246 Records       Name     Catalog     Schema     Type       COPY_DB_EMPLOYEE_TO_DB_E     DGDB     PROCED     PROCED       DB_AVG_RATE     DGDB     PROCED     PROCED       DB_EMPLOYEE_INSERT     DGDB     PROCED     PROCED       DB_EMPLOYEE_PROC     DGDB     PROCED     PROCED       DB_EMPLOYEE_PROC2     DGDB     PROCED     V       Select     Name     Catalog     Schema     Type	Procedure Selection-				
COPY_DB_EMPLOYEE_TO_DB_E     DGDB     PROCED       DB_AVG_RATE     DGDB     PROCED       DB_EMPLOYEE_INSERT     DGDB     PROCED       DB_EMPLOYEE_PROC     DGDB     PROCED       DB_EMPLOYEE_PROC2     DGDB     PROCED       Select     Name     Catalog     Schema	Results:246 Records				
DB_AVG_RATE DGDB PROCED ■ DB_EMPLOYEE_INSERT DGDB PROCED ■ DB_EMPLOYEE_PROC DGDB PROCED ■ DB_EMPLOYEE_PROC2 DGDB PROCED ▼ Select Name Catalog Schema Type	Name		Catalog	Schen	na Type
DB_EMPLOYEE_INSERT DGDB PROCED DB_EMPLOYEE_PROC DGDB PROCED DB_EMPLOYEE_PROC2 DGDB PROCED ▼ Select Name Catalog Schema Type	COPY_DB_EMPLOYE	E_TO_DB_E	100	DGDB	PROCED
DB_EMPLOYEE_PROC DGDB PROCED DB_EMPLOYEE_PROC2 DGDB PROCED Select Name Catalog Schema Type	DB_AVG_RATE			DGDB	PROCED
DB_EMPLOYEE_PROC2 DGDB PROCED 💌 Select Name Catalog Schema Type	DB_EMPLOYEE_INSE	ERT		DGDB	PROCED
Select Catalog Schema Type	DB_EMPLOYEE_PRO	)C		DGDB	PROCED
Name Catalog Schema Type	DB_EMPLOYEE_PRO	0C2		DGDB	PROCED 🔽
	Select				
DB_EMPLOYEE_I DGDB PROCEDURE	Name	Catalog	Sche	ma	Туре
	DB_EMPLOYEE_I		DGDB		PROCEDURE
Remove	DB_EMPLOYEE_I	Catalog		ma	
					Consol
OK Cancel					

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**.

	Procedu	ire Parameters	
Procedure Para Parameters —	ameters For: DB_EMP	LOYEE_INSERT	
Index	Name	Туре	ParamType
1	EMPNO	OTHER	IN
2	LASTNAME	VARCHAR	IN
3	FIRSTNAME	VARCHAR	. IN
4	RATE	DECIMAL	🖸 IN
		DECIMAL	
		DISTINCT	
		DOUBLE	
		FLOAT	
		INTEGER	
		JAVA OBJECT	
		LONGVARBINAE	
		LONGVARCHAR	-
		Ad	ld Remo <u>v</u> e
<u>R</u> estore		<u>o</u> k	<u>C</u> ancel

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

ResultSet Name:		By Executing Manually With Assistance
Index	Original Name Column	Name Type
muex	<u>j originaritanici oolanini</u>	
muex		
muex		
muex		

#### 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	<b>Prepared Statement</b>
-----------	---------------------------

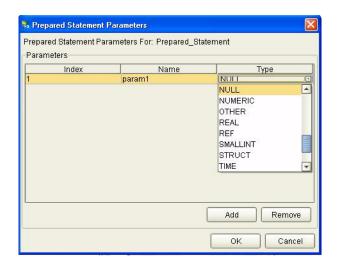
Steps	Add Prepared Statements		
<ol> <li>Select Wizard Type</li> <li>Connect to Database</li> </ol>			
<ol> <li>Select Database Objects</li> <li>Select Tables/Views</li> <li>Select Procedures</li> <li>Add Prepared Statements</li> <li>Specify the OTD Name</li> <li>Review Selections</li> </ol>	Name	SQL Statement	
	Add Edit	Remove	

<sup>2</sup> 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

	Add Prepared	Statement	8
Prepared Stateme <u>n</u> -SQL Statement-	Name:	]	
update dbempl	oyee set hire_date? v	vhere emp_no ='800'	
Open	Save As		
		<u> </u>	Cancel

- 3 On the **Add Prepared Statement** window, the name you assigned to the Prepared Statement appears. To edit the parameters, click **Edit Parameters**. You can change the datatype by clicking in the **Type** field and selecting a different type from the list.
- 4 Click **Add** if you want to add additional parameters to the Statement or highlight a row and click **Remove** to remove it. Click **OK**. See **Figure 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 but it is recommend you do not changed the Name. Doing so will cause a loss of integrity between the Resultset and the Database. Click **OK**. See **Figure 22**.

Index	Name	Туре	T
1	EMPNO	CHAR	1
2	FIRSTNME	VARCHAR	
3	MIDINIT	CHAR	
4	LASTNAME	VARCHAR	
5	WORKDEPT	CHAR	
6	PHONENO	CHAR	
7	HIREDATE	DATE	
8	JOB	CHAR	
9	EDLEVEL	SMALLINT	
10	SEX	CHAR	
11	BIRTHDATE	DATE	-
12	SALARY	DECIMAL	
12	PONILIE	DECIMAL	

#### Figure 22 ResultSet Columns

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

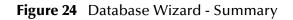
#### Specify the OTD Name

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

Steps	Specify the OTD Name
OTD Wizard Selection     Connect to Database     Select Database Objects     Select Tables/Views     Select Procedures     Add Prepared Statements     Secify the OTD Name     Review Selections	OTD Name: OraOTD
	<back next=""> Einish Cancel Help</back>

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**.



	New Wizard - Oracle Database	*
Steps	Review your Selections	
<ol> <li>OTD Wizard Selection</li> <li>Connect to Database</li> <li>Select Database Objects</li> <li>Select Tables/Views</li> <li>Select Procedures</li> <li>Add Prepared Statements</li> <li>Specify the OTD Name</li> <li>Review Selections</li> </ol>	You have successfully completed the Database Wizard. Please review your selections. Connection Information Connection type: Oracle Host name: enigma Port ID: 1521 SID: orcl8 User name: dgdb Selected Tables/Views DB_EMPLOYEE OTD Information OTD Name: OraOTD	
SEEBEYOND	To close this wizard, click Finish.	

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

# **Importing the Project**

This chapter discusses an overview of the Sample Projects provided with this eWay.

This Chapter Includes:

- Using the Sample Project in eInsight on page 40
- Working with the BPEL Operations on page 42
- Working with the Sample Project in eGate on page 52

# 5.1 eInsight Engine and eGate Components

You can deploy an eGate component as an Activity in an eInsight Business Process. Once you have associated the desired component with an Activity, the eInsight engine can invoke it using a Web Services interface. Examples of eGate components that can interface with eInsight in this way are:

- Java Messaging Service (JMS)
- Object Type Definitions (OTDs)
- An eWay
- Collaborations

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

# 5.2 Using the Sample Project in elnsight

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 **Ora\_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 Integrator Tutorial*.

#### **Importing the Sample Project**

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

Figure 25 Importing the sample project

Enterprise E	xplorer [Project Explorer]
Sample_Re	New Project
- CMa	Sort by Type
	Sort by Name
- T File	Sort by Date
O- 🐺 JCE	Import
- Orac	Refresh All from Repository
🗢 🚮 DBe	User Management
🛛 🗕 📑 Orac	Properties

2 In the **Import Manager** window, **From ZIP file** browse to the location of the sample folder and select the following .zip file **Ora\_BPEL\_Sample.zip** and click **Import**.See **Figure 26**.

	Import Manager		(
specify the ZIP file and the root to i	nport to:		
rom ZIP file:		В	rowse
Root project:	Root environ		•
Sample_Repository	Sample_Re	epository	-
Importing 0 projects	Importing 0 e	environments	
ļ.			
		Import I	Close
		[ Tubou ]	01000

Figure 26 Select the project file

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

### **The Business Process**

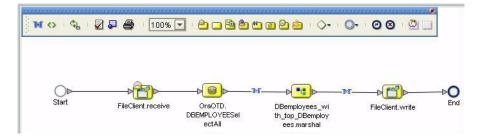
The data used for this sample project is contained within a table called DBEmployee. The table has the following columns:

Column Name	Mapping	Data Type	Data Length
EMP_NO	empno	varchar2	10
LAST_NAME	lastname	varchar2	30
FIRST_NAME	firstname	varchar2	30
SS_NUMBER	ssnumber	varchar2	20
HIRE_DATE	hiredate	varchar2	12

# Table 1Sample project dataTable 2

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

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



### Figure 27

# 5.2.1. Working with the BPEL Operations

You can associate an eInsight Business Process Activity with the Oracle eWay, both during the system design phase. 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.

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

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

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

elnsight Operation	Input	Output
SelectAll	where() clause (optional)	Returns all rows that fit the condition of the where() clause
SelectMultiple	number of rows where() clause. Optional	Returns the number of rows specified that fit the condition of the where() clause
SelectOne	where() clause. Optional	Returns the first row that fits the condition of the where() clause
Insert	definition of new item to be inserted	Returns status.
Update	where() clause	Returns status.
Delete	where() clause	Returns status.

### whereClause()

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

For example:

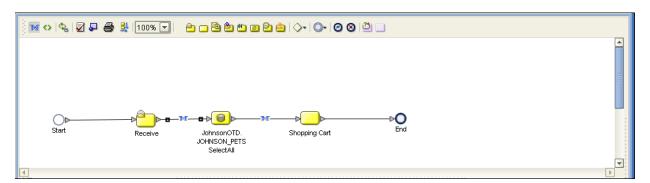
```
where
Clause such as where column2=2 AND column1=1 OR column3=3 is valid
```

### 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 all 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

Transformation Designer	
/シ + AND / == ※ != OR × NOT × ഈ 過留にSI ● × キ × UI ×	
Output All	Input All
Johnson DT D. JOHNSON_PETSSelectAll.Input Johnson DT D. JOHNSON_PETSSelectAll.Output TITEM_ID="" string1 string2 return string string2 return string2 return string2 r	Business Process Attibutes ∰ nsonOTD.JOHNSON_PETSSelectAll.Input t ↔ input d ↔ whereClause ◆ →
BusinessProcess3	

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

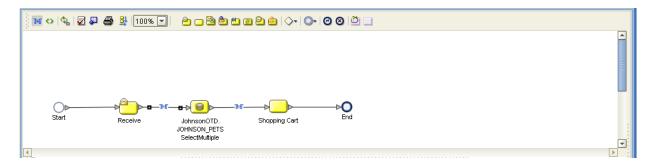
Transformation Designer	
🔆 🕂 440 🖉 🎟 🕸 🚛 OR » 1407 » 🐉 🤮 😭 CSI 🐲 » 🛧 » 640 »	
Output	Input All
🚰 Business Process Attributes	Business Process Attributes 省
or a second seco	ShoppingCart 🍡 🧑
· o- 🙀 output	cart 🖼 – 🌼 🕴
· · ← ♦ rowCount	item 🔀 – 🧔
o- 🙀 JOHNSON_PETSSelectAllTableRespons	- itemID 🔷 -
E ← ♦ rowCount	productID 🔷 —
- • PET_TYPE	petName 🔷 —
- I STOCK_AVAIL	petDescription 🔷 —
	petType 🔷 🚽
- • PET_NAME	petPrice 🔷 —
	petQuantity 🔷 🚽
- • ITEM_ID	JohnsonOTD.JOHNSON_PETSSelectAll.Input
	JohnsonOTD.JOHNSON_PETSSelectAll.Output
	JohnsonOTD.JOHNSON_PETSSelectAll.Fault
	ITEM_ID
	·
	<b>T</b>
BusinessProcess3	

### 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



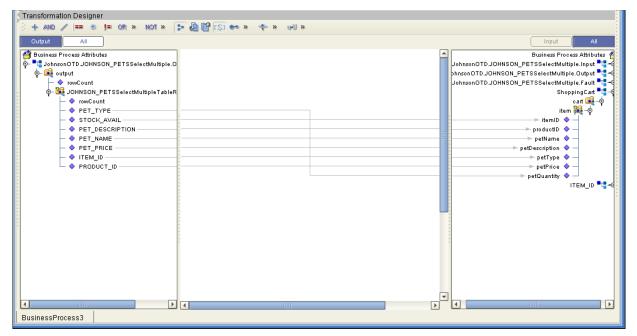
The figure below shows the definition of the number of rows and where() clause into the input for the SelectMultiple operation. You could also use an empty string or Item\_ID='123'.

Transformation Designer	
) 🕂 AND 🥒 💷 🕸 💷 OR » NOT » 🕞 🖓 😭 💽 🚧 » 🛧 » 🖟 W »	
Output All	Input All
Business Process Attributes Johnson OTD JOHNSON_PETSSelectMultiple.in output return string JOHNSON_PETSSelectMultipleTableR PET_TYPE STOCK_AVAIL PET_DESCRIPTION PET_NAME PET_PRICE ITEM_ID String-literal """ String-literal	Business Process Attributes
BusinessProcess3	

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.





### 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.

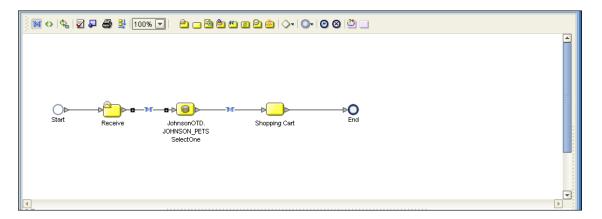


Figure 34 SelectOne Sample Business Process

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

Figure 35 SelectOne Input

Transformation Designer		
[ 🗄 🕂 AND 🥒 💷 🕷 🚛 OR » NOT » [	🝓 🗳 🔝 🕶 » 💠 » 🔎 »	
Output All		Input All
🚰 Business Process Attributes		
💁 📲 ShoppingCart	CSJ string-literal	JohnsonOTD.JOHNSON_PETSSelectOne.Input
∲- <sup>∎</sup> tem_ID	TEM ID-"	input 🚅 → whereClause ♦ →
Stilling     Stilling	string1	
💁 📲 JohnsonOTD.JOHNSON_PETSSelectOne.Output	string2	
🐵 📲 JohnsonOTD.JOHNSON_PETSSelectOne.Fault	return string 🖕 🐤 concat 🗠	
	>+ string1	
	🕼 💦 string-literal	
	return string	
BusinessProcess3		

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.

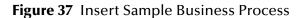
#### Figure 36 SelectOne Output

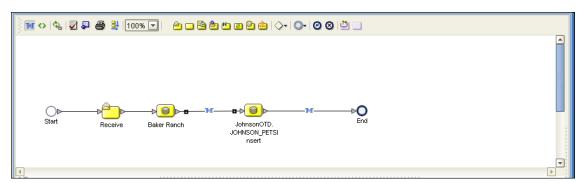
Transformation Designer		
◇+ AND / == 赤 != 0R × NOT × [お 過 留 (S) �* × キ × 山川 ×		
Output	Input All	
	Business Process Attributes ShoppingCarl item item productID petVane petVane petPrice JohnsonOTD.JOHNSON_PETSSelectOne.Fault JohnsonOTD.JOHNSON_PETSSelectONE.Fault JohnsonOTD.JOHNSON	
BusinessProcess3		

### 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.





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

### Figure 38 Insert Input

Imput       Al         Imput       Al         Business Process Attributes       JohnsonOTD JOHNSON_PETSSelectOne.Input         JohnsonOTD JOHNSON_PETSSelectOne.Fault       JohnsonOTD JOHNSON_PETSSelectOne.Fault         JohnsonOTD JOHNSON_PETSSelectOne.Fault       + PET_PRCE         JohnsonOTD JOHNSON_PETSInsert.Fault       + PET_PRCE         JohnsonOTD JOHNSON_PETSINSENCE       + PET_PRCE         JohnsonOTD JOHNSON_PETSINSENCE       + PET_PRCE         JohnsonOTD JOHNSON_PETSINSENCE<	Transformation Designer	
Business Process Attributes       Business Process Attributes         ShoppingCat       JohnsonOTD JOHNSON_PETSselectOne.Input         JohnsonOTD JOHNSON_PETSselectOne.Fault       PET_TYPE         JohnsonOTD JOHNSON_PETSisect.Output       PET_TYPE         JohnsonOTD JOHNSON_PETSisect.Output       PET_TYPE         JohnsonOTD JOHNSON_PETSisect.Output       PET_TYPE         JohnsonOTD JOHNSON_PETSisect.Output       PET_PRICE         PetType       PetType         PetBosciption       PetBosciption         PetBosciption       PetFlore         PetBosciption       PetFlore         PetOuantity       PetOuantity	🔆 + AND 🖉 == 巻  = OR » NOT » 🐉 🙆 🔮 🕼 🕬 × 💠 » 내비 »	
ShoppingCart JohnsonOTD JOHNSON_PETSSelectOne.input JohnsonOTD JOHNSON_PETSSelectOne.fault JohnsonOTD JOHNSON_PETSIselectOne.fault JohnsonOTD JOHNSON_PETSIselectOne.fault DohnsonOTD JOHNSON_PETSInset.input BakerRanch Deptetsinset.fault De	Output All	Input All
	ShoppingCat     JohnsonDTD_JOHNSON_PETSSelectOne.Input     JohnsonDTD_JOHNSON_PETSSelectOne.Fault     JohnsonDTD_JOHNSON_PETSnet.Input     JohnsonDTD_JOHNSON_PETSnet.Output     JohnsonDTD_JOHNSON_PETSInset.Output     JohnsonDTD_JOHNSON_PETSInset.Output     JohnsonDTD_JOHNSON_PETSInset.Fault     JohnsonDTD_JOHNSON_PETSINSet.	Johnson OT D JOHNSON_PETSInset.Input
BusinessProcess3		

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

Figure 39 Insert Output

Transformation Designer		
🔆 🕂 AND 🦯 == 🕸 💷 OR » NOT »	🎾 🙆 📴 🕼 🕶 » 🔸 » 🖃 »	
Output		Input All
🚰 Business Process Attributes		Business Process Attribute
JohnsonOTD.JOHNSON_PETSInsert.Output		ShoppingCart =
o- 🚅 output		JohnsonOTD.JOHNSON_PETSSelectOne.Input
🗆 🔶 status		JohnsonOTD.JOHNSON_PETSSelectOne.Output
		JohnsonOTD.JOHNSON_PETSSelectOne.Fault
		JohnsonOTD.JOHNSON_PETSInsert.Input
	· · · · · · · · · · · · · · · · · · ·	JohnsonOTD.JOHNSON_PETSInsert.Output
	·	JohnsonOTD.JOHNSON_PETSInsert.Fault
	*	BakerRanch =
	· · · · · · · · · · · · · · · · · · ·	
	*	
	*	
	· · · · · · · · · · · · · · · · · · ·	
	7 7	
	*	
	*	
	* *	
	· ·	
BusinessProcess3		

# 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.

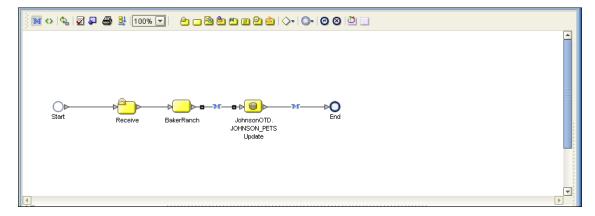
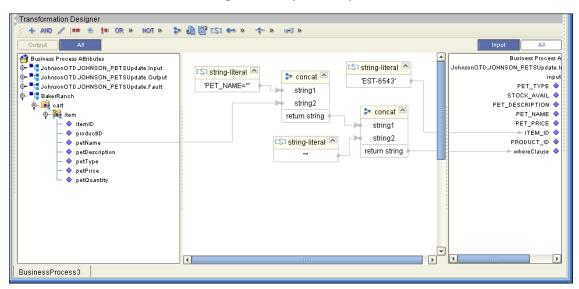


Figure 40 Update Sample Business Process

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.

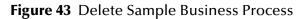
### Figure 42 Update Output

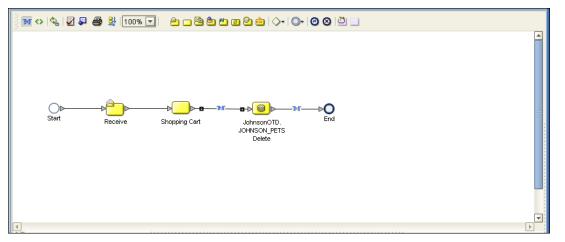
Transformation Designer	
🔆 🕂 AND 🥒 💷 🚓 💷 OR » NOT » 💲	• 🕼 🔐 🕼 » 🔶 » 🔶 »
Output All	Input All
省 Business Process Attributes	Business Process Attr
• • • • JohnsonOTD.JOHNSON_PETSUpdate.Output	ohnsonOTD.JOHNSON_PETSUpdate.Inp
o- 🚅 output	input 📭
🖾 🤶 rowCount	PET_TYPE 🔷 –
8	STOCK_AVAIL 🔷 —:
	, PET_DESCRIPTION 🔷 🚽
	PET_NAME I PET_NAME
	PET_PRICE
	ITEM_ID 🔷 🚽
	PRODUCT_ID 🔶 —
	whereClause 🔷 —
	hnsonOTD.JOHNSON_PETSUpdate.Outp
	ohnsonOTD.JOHNSON_PETSUpdate.Fav
	BakerRand
	▼
BusinessProcess3	

### Delete

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





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

### Figure 44 Delete Input

Transformation Designer		
🔆 🕂 AND 🥒 == 🏦 💷 OR » NOT »	🐉 🍓 🔮 🕼 🐲 » 💠 » 🔲 »	
Output All		Input All
Business Process Attributes Choise on DT JJOHNSON_PETSDelete.input Choise on DT JJOHNSON_PETSDelete. Gutput Choise on DT JJOHNSON_PETSDelete. Fault Choise of the state of	CSI string-literal PRODUCT_ID=" string1 string2 return string string1 string2 return string " "	Business Process Attribute JohnsonOTD.JOHNSON_PETSDelete.Input input delete whereClause
BusinessProcess3		

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

Figure 45 Delete Output

Transformation Designer		
🔆 🕂 AND 🥒 == 🏦 💷 OR » NOT »	🏞 🧶 📴 🕼 🔹 🔹 💠 🐘 🖉 🧶	
Output All		Input All
🚰 Business Process Attributes		Business Process Attributes 🗳
• • • JohnsonOTD.JOHNSON_PETSDelete.Output • • • • • • • • output		sonOTD.JOHNSON_PETSDelete.Input
v av ouput ↓ ♦ rowCount		sonOTD.JOHNSON_PETSDelete.Fault
		ShoppingCart 📲 🗝
	r	
BusinessProcess3		

# 5.3 Using the Sample Project in eGate

Follow the instructions given in **Importing the Sample Project** on page 41 importing the **Ora\_JCE\_Sample.zip** file.

# 5.3.1. Working with the Sample Project in eGate

This sample project updates the hire\_date column within the DBEmployee table by selecting the employee whose employee number is equal to 800 and then updates the record.

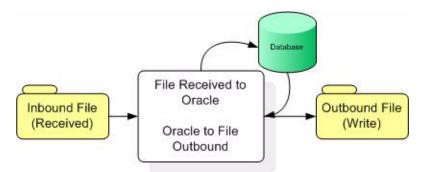
The data used for this projects is within a table called DBEmployee. The table contains the following columns with the following columns:

	la	ble 4	
Column Name	Mapping	Data Type	Data Length
EMP_NO	Empno	varchar2	10
LAST_NAME	Lastname	varchar2	30
FIRST_NAME	Firstname	varchar2	30
SS_NUMBER	SSnumber	varchar2	20
HIRE_DATE	HireDate	varchar2	12

Table 3	Sample project data
	Table 4

The sample project consists of an input file containing data that is passed into a collaboration and out to the Oracle 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 Integrator Tutorial*.

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

# 5.3.2. Configuring the eWays

To configure the Inbound File eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between the **File1** and **JCESelect1** service.
- 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 Oracle1 eWay:

- 1 On the Connectivity Map canvas, double click the eWay icon located between the **JCESelect1** service and **Oracle1** database.
- 2 On the resulting **Templates** window, select **Outbound Oracle eWay** and click **OK**.
- 3 On the Properties window, enter the appropriate configurations for the Outbound Oracle eWay and click OK. See **Working with eWay Property Sheets** on page 12. 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 the **JCESelect1** and **File2** service.
- 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 **JCESelect\_output%d.dat**. For the remaining parameters, the default settings are used.
- 4 When you have completed your selections, click **OK**.

### Creating Rules within the Collaboration

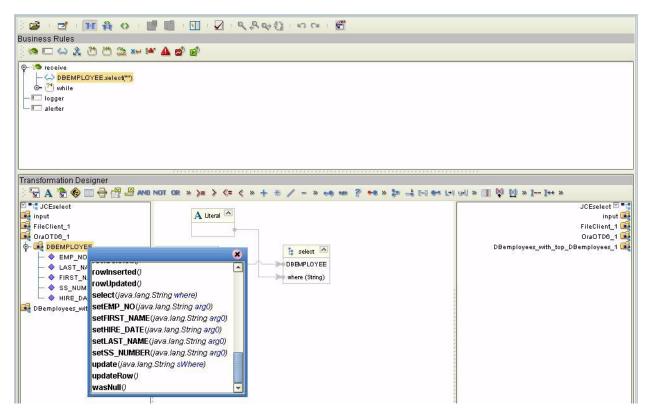
1 On the Transformation Designer window's left pane, expand the **OraOTD6** node and the **DBEMPLOYEE** sub-node by selecting them and double-clicking. See **Figure 47**.



) 😅 - 🗹 - 🕅 🍂 🔿 - 📑 📑	1	
Business Rules		
) 🗠 🗆 🍪 🌋 🖑 🖄 🏔 🖬 🛕 I	2° 🖻	
receive     DBEMPLOYEE.select"     while     logger     alerter		
Transformation Designer		1011010101010101
A SAME AND A SAME AND A SAME	R » >= > <= < » + * / - » +* ** ? **	* 🐉 🚽 [=] 🕪 [=] (#] * 🗍 🔯 🔯 * ]== ]++ *
JCEselect     input     input     FileClient_1     OraoTD6_1     OraoTD6_1     OEMPLOYEE	A Literal	JCEselect ⊠ input FileClient_1 OraOTD6_1 DBemployees_with_top_DBemployees_1

- 2 To create a new rule, highlight **DBEMPLOYEE** and right click to open the Methods sub-menu.
- 3 From the **Select a method to call** submenu, highlight the **select(java.lang.String where)** method and enter. See **Figure 48**.

### Figure 48 Selecting a method



- 4 Next you need to create a **Literal** by selecting the **Literal (A)** icon on the Transformation Designer toolbar.
- 5 On the **Create Literal** popup window, select **String** from the drop-down list and click **OK**.
- 6 On the Literal box, place your cursor inside the box and while depressing the left mouse key, drag a new connection line between the Literal box to the where(String) method node located next to the select() box. See Figure 49.

### Figure 49 Literal mapping

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Business Rules	
ු ලං 🗆 🛶 දී. ඊ. ඊ. 🏔 🚧 💣 💣	
Image: Provide the select image: Provi	
Business Rules Designer S S A S S = A S = > <= < > + 5 / - > → 5 = > > > > > > > > > > > > > > > > > >	
Input       Input         FileClient_1       Input         TraDTD6_1       Input         BEMPLOYEE       Input         Bemployees_with_top_DBemployees_1       Input	JCEselect <sup>™</sup> ■ * input 6 FileClient 1 1 OraOTD8_1 1 DBemployees_with_top_DBemployees_1
DBBMPLOYEE where (String)	

A new rule is created.

- 7 On the Business Rules canvas, highlight the <\_>DBEMPLOYEE.select("") rule. Click the while() icon located on the toolbar. A new while() condition is waiting to be created within the list.
- 8 On the **Business Rules** list, highlight the condition. While highlighted, select DBEMPLOYEE located under the OraOTD6\_1 node. Right click and select the next() method from the sub-menu. See Figure 50.

Figure 50 next()

) 🗃 I 🗹 I 🕅 🍂 🖒 I 📑 🛯	III Q   Q & Q Q   10 0   ₩	
Business Rules		
·····································	a) a)	
Business Rules Designer		1
	0R » >= > <= < » + % / - » +% m ? +% >= == %	I № Ю » I I++ »
JCEselect       input       FileClient_1       OraOTDE_1       OraOTDE_E       DBEMPLOYEE       DBemployees_with_top_DBemployees_1	DBBMPLOYEE result (boolean)	JCEselect № input FileClient_1 DraTD6_1 DBemployees_with_top_DBemployees_1

9 On the Business Rules list, highlight the New Rule located under the DBEMPLOYEE.next condition. On the left pane of the Business Rules canvas, expand the DBEMOPLOYEE sub-node located under the OraOTD6\_1 OTD node to expose the database columns. On the right side of the Business Rules canvas, expand the DBemployees\_with\_top\_DBemployees\_2 OTD node. Expand the \_sequence\_A node exposing the table's columns. Drag-and-drop the following fields from the left pane under the DB\_EMPLOYEE sub-node to the right pane under \_sequence\_A:

EMP_NO	>	EmpNo
LAST_NAME	>	Lastname
FIRST_NAME	>	Firstname
SS_NUMBER	>	SSnumber
HIRE_DATE	>	HireDate

10 On the Business Rules list, highlight Copy Hire\_Date to HireDate. On the Business Rules Designer's left pane, right-click on the DBemployees\_with\_top\_DBemployees\_1 node. From the sub-menu, select and click Select a method to call. From the sub-menu, select and double click

marshalToString(). See Figure 51.

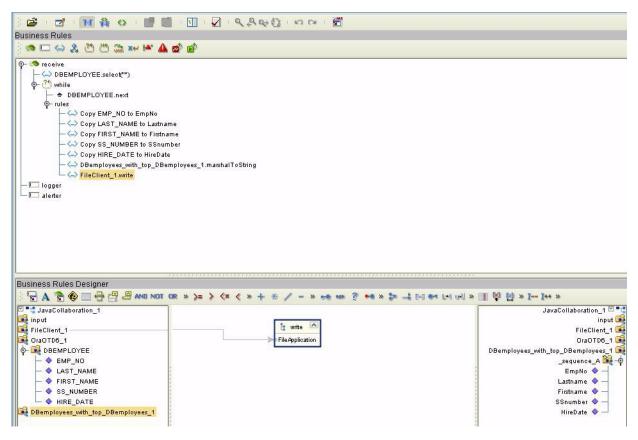
### Figure 51 marshalToString()

Imput       Imput       Imput       Imput         Imput       Imput       Imput			
Image: Second	) 😅 + 🛃 + 🍽 🍂 📀 + 📑 📑 + 🖬		
Second State	Business Rules		
Second State	े 🦔 🖂 😂 🕅 🤲 🍮 🚧 🖬 🙆 👩	5	
DEEMPLOYEE.select"     while         → DEEMPLOYEE.select"         while         → Copy EMP_NO to EmpNo         → Copy ST_NAME to Lastname         → SS_NUMABER         → FIRST_NAME         → SS_NUMABER         → HIRE_DATE         → HIRE_DATE         → DBemployees_uth top DBemployees_1         → Tesut (String)         → Tesut         → Tesut (String)			
A       Construction       AND NOT OR       > = > <       <       >        >	→ DBEMPLOYEE.seleot"")	loyees_1.marshalToString to Text	
input input FileClient_1 OraOTD6_1 OraOTD6_1 DBEmployees EMP_NO EMP_NO EMP_NO EMP_NO EMP_NO EMP_NO EMP_NO EMP_NO EMP_NO EMPLOYEE DBemployees EMP_NO EMPLOYEE DBemployees EMP_NO Employees E	Business Rules Designer	>= > <= < > + ⊕ / - > + ⊕ => ? += > ≥ =↓ [	⊣ ≪ L=1 64) » ]] 🖗 12 × I I++ »
FileClient_1       FileClient_1       FileClient_1         OraOTD6_1       DBemployees       OraOTD6_1         OraOTD6_1       DBemployees       OraOTD6_1         OraOTD6_1       DBemployees       OraOTD6_1         OraOTD6_1       DBemployees       DBemployees_with_top_DBemployees_         OF       FIRST_NAME       Employ         OF       SS_NUMBER       Lastname Interport         DBemployees_with_top_DBemployees_1       First_Name       First_Name	JCEselect		JCEselect 🗹 🍡
OraOTD6_1     Imarshallosting       OBEMPLOYEE     DBemployees       O EMP_NO     result (String)       O FIRST_NAME	🙀 input		input 🖼
IDBEMPLOYEE     DBemployees       IDDBEMPLOYEE     IDDBEMPLOYEE       IDDBEMPLOYEE     IDDBEMPLOYEE       IDDBEMPLOYEE     IDDBEMPLOYEE       IDDBEMPLOYEE     IDDBEMPLOYEE       IDDBEMPLOYEE     IDDBEMPLOYEE	🙀 FileClient_1	the marshal To String	FileClient_1 🖼
Image: Constraint of the constr	GraOTD6_1		📃 🚽 Text 🔷 🚽
→ EMP_NO     result (String)     DBemployees_with_top_DBemployees_       → LAST_NAME	OF CONTRACTOR OF	DBemployees	OraOTD6_1 🖬
		result (String)	DBemployees_with_top_DBemployees_1 🖬
	- 🔷 LAST_NAME		_sequence_A 🔀 – 👳
↓ ◆ HIRE_DATE     Firstname ◆       ♣ DBemployees_with_top_DBemployees_1     SSnumber ◆	- IRST_NAME		EmpNo 🔷 —
🙀 DBemployees_with_top_DBemployees_1 - SSnumber 🔷 -	- 🔷 SS_NUMBER		: Lastname 🔷 —
	HIRE_DATE		Firstname 🔷 —
HireDate 🧇 —	DBemployees_with_top_DBemployees_1		SSnumber 🔷 —
			HireDate 🔷 —

11 On the Business Rules list, highlight

**DBemployees\_with\_top\_DBemployees\_1.marshalToString**. On the Business Rules Designer's left pane, right-click on the **FileClient\_1** node. From the sub-menu, select and click **Select a method to call**. From the sub-menu, select and double-click **write()**. See

### Figure 52 write()



12 Click **Save** to save the Collaboration.

# 5.3.3. Creating the External Evironment

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:

On the Environment Explorer, highlight and right-click the Oracle eWay profile. Select **Properties**. Enter the configuration information required for your Outbound Oracle eWay. See **Setting the Properties in the Outbound eWay Environment** on page 17

# 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 Integrator Tutorial*.

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

# 5.4 Supported Data Types

The Oracle eWay supports the following data types:

Data Type	Description	Column Length
Number	Variable-length numeric data. Maximum precision p and/or scale s is 38.	Variable for each row. The maximum space required for a given column is 21 bytes per row.
VarChar2	Variable-length character data, with maximum length size bytes or characters.	Variable for each row, up to 4000 bytes per row. Consider the character set (single-byte or multibyte) before setting size. A maximum size must be specified.
Char	Fixed-length character data of length size bytes or characters.	Fixed for every row in the table (with trailing blanks); maximum size is 2000 bytes per row, default size is 1 byte per row. Consider the character set (single-byte or multibyte) before setting size.
Raw	Variable-length raw binary data.	Variable for each row in the table, up to 2000 bytes per row. A maximum size must be specified. Provided for backward compatibility.
Long	Variable-length character data.	Variable for each row in the table, up to $2^{32}$ - 1 bytes, or 2 gigabytes, per row. Provided for backward compatibility.
Clob		Up to $2^{32}$ - 1 bytes, or 64k.

**Table 5** Standard Data Types Supported by the Oracle eWay

Data Type	Description	Column Length
Date	Fixed-length date and time data, ranging from Jan. 1, 4712 B.C.E. to Dec. 31, 4712 C.E.	Fixed at 7 bytes for each row in the table. Default format is a string (such as DD-MON- RR) specified by the NLS_DATE_FORMAT parameter. Oracle expects a format of: "YYYY-MM- DD; hh:mm:ss.x".

# 5.5 **Converting Data Types in the Oracle eWay**

When working with data in the Oracle eWay OTD's, you may need to do a data conversion. The following tables should help:

Oracle Data Type	OTD/Java Data Type	Java Method or New Constructor to Use (Default: Java Method)	Sample Data
Int	BigDecimal	Call a New Constructor BigDecimal: java.math.BigDecimal(String)	123
Smallint	BigDecimal	Call a New Constructor BigDecimal: java.math.BigDecimal(String)	123
Number	BigDecimal	Call a New Constructor BigDecimal: java.math.BigDecimal(String)	123
Decimal*	BigDecimal	Call a New Constructor BigDecimal: java.math.BigDecimal(String)	147.78
Real	Double	<b>Double</b> : java.lang.Double.parseDouble(String)	147.78
Float	Double	<b>Double</b> : java.lang.Double.parseDouble(String)	147.78
Double	Double	<b>Double</b> : java.lang.Double.parseDouble(String)	147.78
Date	TimeStamp	<b>TimeStamp</b> : java.sql.TimeStamp.valueOf(String)	2003-08-11 11:47:39.0
Varchar2	String	Direct Assign	Any character
Char	String	Direct Assign	Any character

**Table 6** Insert and Update Operations Datatype Conversions (Text/String input data)

Oracle Data Type	OTD/Java Data Type	Java Method or New Constructor to Use (Default: Java Method)	Sample Data
Long Char	String	Direct Assign	Any character
Raw	Byte[]	String: java.lang.String.getBytes()	Any character
Long Raw	Byte[]	String: java.lang.String.getBytes()	Any character
CLOB	Clob	See Appendix	Any character

### Table 7 Select Operation Datatype Conversion (Text/String output data)

Oracle Data Type	OTD/Java Data Type	Methods To Use	Sample Data
Int	BigDecimal	<b>BigDecimal</b> : java.math.toString()	123
SmallInt	BigDecimal	<b>BigDecimal</b> : java.math.toString()	123
Number	BigDecimal	<b>BigDecimal</b> : java.math.toString()	123
Decimal	BigDecimal	<b>BigDecimal</b> : java.math.toString()	123.67
Real	Double	<b>Double</b> : java.lang.Double.toString(double)	123
Float	Double	<b>Double</b> : java.lang.Double.toString(double)	123
Double	Double	<b>Double:</b> java.lang.Double.parseDouble(String )	123
Date	TimeStamp	<b>TimeStamp</b> : java.sql.TimeStamp.valueof()	2003-08-11
			11:47:39.0
Varchar2	String	Direct Assign	Any characters
Char	String	Direct Assign	Any Characters
Raw	Byte[]	N/A	N/A
Long Raw	Byte[]	N/A	N/A
CLOB	Clob	N/A	N/A

# 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. Data Types

Oracle Tables support:

- Real
- Float
- CLOB.

For all others, to use the data types Float, Double, and CLOB build them using a data type of "Other".

### Long RAW for Prepared Statements and Stored Procedure support:

The following 2 parameters must be set prior to the Insert/Update/Delete statement.

```
setConcurrencyToReadOnly()
setScrollTypeToForwardOnly()
```

# 5.6.2. The Table

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

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

- SetConcurrencytoUpdatable
- SetConcurrencytoReadOnly
- SetScrollTypetoForwardOnly
- SetScrollTypetoScrollSensitive
- SetScrollTypetoInsensitive

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

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

# The Query Operation

To perform a query operation on a table

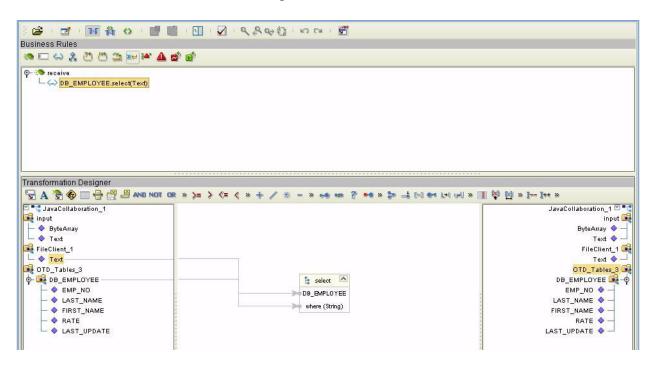
- 1 Execute the **select()** method with the "**where**" clause specified if necessary.
- 2 Loop through the ResultSet using the **next()** method.
- 3 Process the return record within a **while()** loop.

For example:

```
//Table_OTD_1.getDB_EMPLOYEE().select(Text)
    Table_OTD_1.getDB_EMPLOYEE().select( read.getText() );
```

```
//while Table_OTD_1.getDB_EMPLOYEE.next
    while (Table_OTD_1.getDB_EMPLOYEE().next()) {
    //Copy LAST_NAME to LAST_NAME
        Table_OTD_1.getDB_EMPLOYEE().setLAST_NAME(
        Table_OTD_1.getDB_EMPLOYEE().getLAST_NAME() );
}
```

Figure 53 Select()



# **The Insert Operation**

To perform an insert operation on a table

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

This example inserts an employee record.

```
//Table_OTD_1.getDB_EMPLOYEE.insert
	Table_OTD_1.getDB_EMPLOYEE().insert();
//Copy EMP_NO to EMP_NO
	insert_DB_1.getInsert_new_employee().setEmployee_no(
	java.lang.Integer.parseInt(
		employeedb_with_top_db_employee_1.getEmployee_no() ) );
//@map:Copy Employee_lname to Employee_Lname
		insert_DB_1.getInsert_new_employee().setEmployee_Lname(
		employeedb_with_top_db_employee_1.getEmployee_lname() );
//@map:Copy Employee_fname to Employee_Fname
		insert_DB_1.getInsert_new_employee().setEmployee_Fname(
		employeedb_with_top_db_employee().setEmployee_Fname(
		employeedb_with_top_db_employee_1.getEmployee_fname() );
//@map:Copy java.lang.Float.parseFloat(Rate) to Rate
```

```
insert_DB_1.getInsert_new_employee().setRate(
    java.lang.Float.parseFloat(
    employeedb_with_top_db_employee_1.getRate() ) );
//@map:Copy java.sql.Timestamp.valueOf(Update_date) to Update_date
    insert_DB_1.getInsert_new_employee().setUpdate_date(
    java.sql.Timestamp.valueOf(
    employeedb_with_top_db_employee_1.getUpdate_date() ) );
    Table_OTD_1.getDB_EMPLOYEE().insertRow();
}
```

# The Update Operation

To perform an update operation on a table

- 1 Execute the **update()** method.
- 2 Using a while loop together with **next()**, move to the row that you want to update.
- 3 Assign updating value(s) to the fields of the table OTD
- 4 Update the row by calling **updateRow()**.

In this example, we move to the third record and update the SALES\_ORDERS Table and Cust\_Name, and Cust\_phone columns.

```
//SalesOrders_with_top_SalesOrders_1.unmarshalFromString(Text)
  SalesOrders_with_top_SalesOrders_1.unmarshalFromString(
 input.getText() );
//DB_sales_orders_1.getSALES_ORDERS.update("SO_num =99")
 DB_sales_orders_1.getSALES_ORDERS().update( "SO_num ='01'" );
//while
 while (DB_sales_orders_1.getSALES_ORDERS().next()) {
//Copy SalesOrderNum to SO num
 DB_sales_orders_1.getSALES_ORDERS().setSO_num(
 SalesOrders_with_top_SalesOrders_1.getSalesOrderNum() );
//Copy CustomerName to Cust_name
  DB_sales_orders_1.getSALES_ORDERS().setCust_name(
 SalesOrders_with_top_SalesOrders_1.getCustomerName() );
//Copy CustomerPhone to Cust_phone
 DB sales orders 1.getSALES ORDERS().setCust phone(
 SalesOrders_with_top_SalesOrders_1.getCustomerPhone() );
//DB_sales_orders_1.getSALES_ORDERS.updateRow
 DB_sales_orders_1.getSALES_ORDERS().updateRow();
}
   }
```

# The Delete Operation

To perform a delete operation on a table

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

In this example DELETE an employee.

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

# 5.6.3. Using CLOBs

To use a CLOB in the Oracle eWay do the following:

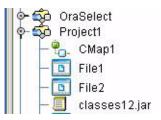
- 1 On the Enterprise Designer, select the project and right click. Select **New**. From the sub-menu select **File**.
- 2 Navigate to the classes12.jar file, <Client eDesigner>\usrdir\modules\ext\oracleadapter\classes12.jar using the Enterprise Designer's Project File Import feature. See Figure 54.

	oracleadapter	Dobasia	
🗋 classes12	2.jar ppconnapi.jar	DGbase.jar	oracp15adapter.jar StcDbLicense.jar
	odegenapi.jar	DGresource.jar	stcewaycodegen.jar
-	dutilapi.jar	DGutil.jar	stcewayutils.jar
	insightintegrationapi.		🗋 stcoraappconn.jar
	raclemodelapi.jar	🗋 oraclebuilder.jar	🗋 stcorawebservice.jar
	raclemodelimpl.jar	🛅 oraclecodegen.jar	I ·
🗋 com.stc.st	tcconfigapi.jar	🗋 oraclewizard.jar	
•]		1000	
ile <u>N</u> ame:	classes12.jar		
iles of <u>T</u> ype:	All Files		[
	. Kanada ana ang kanada		
			Select Cancel
lected Impor	t Ciloc:		
iecteu impoi	(riies.		
			Remove
			Import
			Import

Figure 54 Importing Classes12.jar

- 3 Click Select.
- 4 Click Import. See Figure 55.

#### Figure 55 Classes12.jar in a Project



5 To load the classes12.jar file into your Java Collaboration, select the Import JAR File button. On the Add/Remove Jar Files click Add. See Figure 56.

Imported .	Jar Files		 	7

Figure 56 Add/Remove Jar Files

6 On the **Select Jar File** window, select the **classes12.jar** file and click **Import**. See **Figure 57**.



	Selec	t Jar File		8
Look <u>I</u> n:	🖚 Sample_Project	•		
	je_Sample_Project ses12.jar			
<u>N</u> ame:	Jar Files			
<u>T</u> ype:	Jarries		Import C	ancel

7 On the Add/Remove Jar Files window, click Close.

8 On the Business Rules Designer, call the CLOB method. See Figure 58.

Elle Tools View Window Help	SeeBayond Enterprise Designer 5.0 - Java Collaboration Editor [JCE_Clob_]	2 G J
Enterprise Explorer [Project Explorer]     mrepo     50 DB2     50 M0     50 M0     50 M0     50 Jose     50	Butiness Fules Butiness Fules CCCCC_TEST interf - Copy task and BigDetinest - Copy task and BigDeti	
Cracle1 ← CLOB classes12jar ← SAP ALE ← SAP ALE SPEL ← SAP ALE JCE PRJ ← SA	Class Name: Clob Al Classes: Methods: Class Name: Clob Al Classes: Methods: Claboard Clab	d Ind Mr Let u JCE_Crop_IP4 CLOB_1
Project Explorer Environment Explorer	Mapper JCE_Clob_	

Figure 58 Call the CLOB Method

9 Create a local variable. See Figure 59.

Figure 59 Create a Local Variable

SeeBeyo	nd Enterprise Designer 5.0 - Java Collaboration Editor [JCE_Clob_]
Eile Tools View Window Help	80.
Enterprise Explorer [Project Explorer]	₩ A O II II O A AQQ OO E
Find Class	Convert A stat
Find Class CLOB	InsetRow .
Class Name	mit relect*curtomer_id=9876 for update*)
Class	Croate a Variable
ClassCastException ClassCircularityError	
ClassDesc	Variable Name: myClob
ClassFormatError ClassLoader	
ClientDataSupport	Type
ClassNofoundSkeption ClantProfile ClintProfile Clipboard Clipboar	
ClipboardOwner Clob	Primitive boolean     JCE_Clob_     input 3
CLOB	CLOB_1 CLOB_1
Package: oracle.sgl	PROCESSED_TEXT
Uracie sei	
	OK Cancel
	CLOE
ОК	Cancel
Project Explorer Environment Explorer * JULE_	C10b

10 In the Business Rules Designer, drag the **CLOB** to the Local Variable using the Cast method. Click **Yes** to the Incompatible DataType warning. See **Figure 60**.

	SeeBoyond Enterprise Designer 5.0 - Java Collaboration Editor [JCE_Clob_]	63
Elle Tools View Window Help		* 0
Enterprise Explorer [Project Explorer] mmepo So DB2 So DB2 So Oracle		
- % CMUE_C00 - % Delay_JEC000 - % Delay_JEC000 - % Delay_JEC000 - % Oracle - % Oracle	→ CLOB_TETT underflow → CLOB_TETT underflow	
ම පිහිරි විසෙහුගේ පේදුරි විසෙම්මතාව ම පිදුරි Test	0 * ₹ 2,626, C104_ 30 input ↓ Φ find 30 input ↓ Φ find 10 input 10 in	a _2 53 en Lei JCE_Ctes_S CLOS_TEST CLOS_TEST CLOS_TEST AMER_ID CLOS_TEST CLOS_TEST AmyClob (I
Project Explorer Environment Explorer	Maser I	

Figure 60 Drag CLOB to the Local Variable

11 Use the CLOB putString method assigning 1 to Arg(). See Figure 61.

Figure 61 CLOB putString Method

8	SeeBeyond Enterprise Designer 5.0 - Java Collaboration Editor [JCE_Clob_]	
Elle Tools View Window Help		* 0 *
Enterprise Explorer [Project Explorer]           mrrepo           Go 200           Go 200	Business Rules Business Rules Corpe Jaca and Rulp designed value COB270 to CUSTOMEE_UD Corpe Jack and LOB andy Job to PROCESSED_TEXT CODE_Iscommt CODE_Iscommt Code_Iscommt Code_Iscommt Code Cub2 TEST and Code Code TEST and Code Cod	•
ఉావధి Test	Image: Second	- 50 0 1 (0) (0) JCE_CIO_D (0) (0) (0) (0)
Project Explorer Environment Explorer ×	JCE_Clob_	

12 In the Java Collaboration Editor, the java code should look something like:

```
public void receive( com.stc.connector.appconn.file.FileTextMessage
input,cLOB.CLOBOTD CLOB_1 )
throws Throwable
{
    //@map:CLOB_1.getCLOB_TEST.insert
```

```
CLOB_1.getCLOB_TEST().insert();
        //@map:Copy java.math.BigDecimal.valueOf(9876) to CUSTOMER_ID
        CLOB 1.getCLOB TEST().setCUSTOMER ID(
java.math.BigDecimal.valueOf( 9876 ) );
        //@map:Copy oracle.sql.CLOB.empty_lob to PROCESSED_TEXT
        CLOB_1.getCLOB_TEST().setPROCESSED_TEXT(
oracle.sql.CLOB.empty_lob() );
        //@map:CLOB_TEST.insertRow
        CLOB_1.getCLOB_TEST().insertRow();
//@map:CLOB 1.getCLOB TEST.select("customer id = 9876 for update")
       CLOB_1.getCLOB_TEST().select( "customer_id = 9876 for update"
);
//If
        if (CLOB_1.getCLOB_TEST().next()) {
            //@map:oracle.sql.CLOB myClob;
            oracle.sql.CLOB myClob;
            //@map:Copy cast PROCESSED_TEXT to oracle.sql.CLOB to
myClob
            myClob = (oracle.sql.CLOB)
CLOB_1.getCLOB_TEST().getPROCESSED_TEXT();
           //@map:myClob.putString(1,Text)
            myClob.putString( 1, input.getText() );
            //@map:CLOB_TEST.updateRow
            CLOB_1.getCLOB_TEST().updateRow();
}
    }
```

# 5.6.4. 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.

- *Note:* When creating a Package Stored Procedure in the Database Wizard, you must select *Use fully qualified names*.
- *Note:* Stored Procedure Resultsets are support in Java collaborations only.

### **Executing Stored Procedures**

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

Below are the steps for executing the Stored Procedure:

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

#### For example:

```
package Storedprocedure;
public class sp_jce
public com.stc.codegen.logger.Logger logger;
public com.stc.codegen.alerter.Alerter alerter;
public void receive( com.stc.connector.appconn.file.FileTextMessage
input, com.stc.connector.appconn.file.FileApplication
FileClient_1,employeedb.Db_employee
employeedb_with_top_db_employee_1,insert_DB.Insert_DBOTD insert_DB_1
)
  throws Throwable
    {
//@map:employeedb_with_top_db_employee_1.unmarshalFromString(Text)
   employeedb_with_top_db_employee_1.unmarshalFromString(
   input.getText() );
//@map:Copy java.lang.Integer.parseInt(Employee_no) to Employee_no
   insert_DB_1.getInsert_new_employee().setEmployee_no(
   java.lang.Integer.parseInt(
   employeedb_with_top_db_employee_1.getEmployee_no() ) );
//@map:Copy Employee_lname to Employee_Lname
   insert_DB_1.getInsert_new_employee().setEmployee_Lname(
   employeedb_with_top_db_employee_1.getEmployee_lname() );
//@map:Copy Employee_fname to Employee_Fname
   insert_DB_1.getInsert_new_employee().setEmployee_Fname(
   employeedb_with_top_db_employee_1.getEmployee_fname() );
//@map:Copy java.lang.Float.parseFloat(Rate) to Rate
   insert_DB_1.getInsert_new_employee().setRate(
   java.lang.Float.parseFloat(
   employeedb_with_top_db_employee_1.getRate() ) );
//@map:Copy java.sql.Timestamp.valueOf(Update_date) to Update_date
   insert_DB_1.getInsert_new_employee().setUpdate_date(
   java.sql.Timestamp.valueOf(
   employeedb with top db employee 1.getUpdate date() ) );
//@map:Insert_new_employee.execute
   insert_DB_1.getInsert_new_employee().execute();
//@map:insert_DB_1.commit
   insert_DB_1.commit();
//@map:Copy "procedure executed" to Text
   FileClient_1.setText( "procedure executed" );
//@map:FileClient_1.write
   FileClient_1.write();
    }
}
```

# Manipulating the ResultSet and Update Count Returned by Stored Procedure

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

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

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

The **resultsAvailable()** method, added to the OTD, simplifies the whole process of determining whether any results, be it update Counts or ResultSets, are available after a stored procedure has been executed. Although JDBC provides three methods (**getMoreResults()**, **getUpdateCount()**, and **getResultSet()**) to access the results of a stored procedure call, the information returned from these methods can be quite confusing to the inexperienced Java JDBC programmer and they also differ between vendors. You can simply call **resultsAvailable()** and if Boolean true is returned, you can expect either a valid Update Count when **getUpdateCount()** is called and/or the next ResultSet has been retrieved and made available to one of the ResultSet nodes defined for the Stored Procedure OTD, when that node's **available()** method returns true.

Frequently, Update Counts information that is returned from a Stored Procedures is insignificant. You should process returned ResultSet information and avoid looping through all of the Update Counts. The following three methods control exactly what information should be returned from a stored procedure call. The **enableResultSetsOnly()** method, added to the OTD allows only ResultSets to be returned and thus every **resultsAvailable()** called only returns Boolean true if a ResultSet is available. Likewise, the **enableUpdateCountsOnly()** causes **resultsAvailable()** to return true only if an Update Count is available. The default case of **enableResultSetsAndUpdateCount()** method allows both ResultSets and Update Counts to be returned.

Collaboration usability for a Stored Procedure ResultSet

The Column data of the ResultSets can be dragged-and-dropped from their OTD nodes to the Business Rules. Below is a code snippet that can be generated by the Collaboration Editor:

```
// resultsAvailable() will be true if there's an update count and/or a
result set available.
// note, it should not be called indiscriminantly because each time
the results pointer is
// advanced via getMoreResults() call.
while (getSPIn().getSpS_multi().resultsAvailable())
```

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

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

After calling "**resultsAvailable(**)", the next result (if available) can be either a **ResultSet** or an **UpdateCount** if the default "**enableResultSetsAndUpdateCount(**)" was used.

Because of limitations imposed by some DBMSs, it is recommended that for maximum portability, all of the results in a ResultSet object should be retrieved before OUT parameters are retrieved. Therefore, you should retrieve all ResultSet(s) and update counts first followed by retrieving the OUT type parameters and return values.

The following list includes specific ResultSet behavior that you may encounter:

• The method **resultsAvailable()** implicitly calls **getMoreResults()** when it is called more than once. You should not call both methods in your java code. Doing so may

result in skipped data from one of the ResultSets when more than one ResultSet is present.

- The methods **available()** and **getResultSet()** can not be used in conjunction with multiple ResultSets being open at the same time. Attempting to open more the one ResultSet at the same time closes the previous ResultSet. The recommended working pattern is:
  - Open one Result Set, ResultSet\_1 and work with the data until you have completed your modifications and updates. Open ResultSet\_2, (ResultSet\_1 is now closed) and modify. When you have completed your work in ResultSet\_2, open any additional ResultSets or close ResultSet\_2.
- If you modify the ResultSet generated by the Execute mode of the Database Wizard, you need to assure the indexes match the stored procedure. By doing this, your ResultSet indexes are preserved.
- Generally, getMoreResults does not need to be called. It is needed if you do not want to use our enhanced methods and you want to follow the traditional JDBC calls on your own.

# 5.7 Creating or Editing an OTD from an Existing OTD

### **Editing an Existing OTD**

To edit an OTD do the following:

1 On the Enterprise Explorer, right click on the OTD. From the submenu, click Edit. See **Figure 62**.

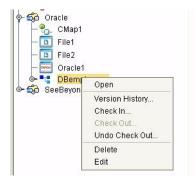




Figure 62 OTD Edit Menu Item

The Database Wizard will open allowing you to change the OTD. You may select additional Tables or edit the existing Tables.

*Caution:* If during the edit process you delete a table that is included in a Collaboration, the Collaboration will fail at run time.

You can also edit your Prepared Statements or Stored Procedures.

2 Save the new or edited OTD using **Save As** and give the OTD a new name.

### Creating a New OTD from an Existing OTD

You can also create a new OTD from an existing OTD. Open the OTD that you would like to be the basis for your new OTD. Following the steps in **Editing an Existing OTD** on page 75, make your edits and save the OTD with a new name.

# 5.8 Using XA

When using XA, do not call the commit() method.

# 5.9 Alerting and Logging

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

# Index

# A

Add Prepared Statements 37

# C

ClassName 13, 22 Connect to Database 29

# D

DatabaseName **18**, DataSourceName **18**, Delimiter **18**, Description **13**, **18**, **22**, driver class, JDBC **13**, DriverProperties **18**,

# E

Environment Properties DatabaseName 18 DataSourceName 18, 25 Delimiter 18, 26 Description 18, 26 DriverProperties 18, 26 Password 19, 26 PortNumber 19, 26 ServerName 19, 27 User 19, 27 External System Requirements 8

# I

Inbound Environment Properties Database 20 Password 20 PortNumber 21 ServerName 21 User 21 InitialPoolSize 14, 23

# J

JDBC

driver class 13, 22

# L

LoginTimeOut 14, 23

# Μ

MaxIdleTime 14, 23 MaxPoolSize 14, 23 MaxStatements 14, 23 MinPoolSize 15, 24

### Ν

NetworkProtocol 15, 24

# 0

Oracle eWay Database Wizard 28 Oracle eWay With eInsight 40 Outbound Properties ClassName 13, 22 Description 13, 22 InitialPoolSize 14, 23 LoginTimeOut 14, 23 MaxIdleTime 14, 23 MaxPoolSize 14, 23 MaxStatements 14, 23 MinPoolSize 15, 24 NetworkProtocol 15, 24 PropertyCycle 15, 24 RoleName 15, 24

# P

Password 19, 20, 26 PortNumber 19, 21, 26 Property settings, Environment DatabaseName 18 DataSourceName 18, 25 Delimiter 18, 26 Description 18, 26 DriverProperties 18, 26 Password 19, 26 PortNumber 19, 26 ServerName 19, 27 User 19, 27 Property settings, Inbound Environment Database 20 Password 20 PortNumber 21 ServerName 21

User 21 Property settings, Outbound ClassName 13, 22 Description 13, 22 InitialPoolSize 14, 23 LoginTimeOut 14, 23 MaxIdleTime 14, 23 MaxPoolSize 14, 23 MaxStatements 14, 23 MinPoolSize 15, 24 NetworkProtocol 15, 24 PropertyCycle 15, 24 PropertyCycle 15, 24

### R

RoleName 15, 24

### S

Select Database Objects Select Procedures Select Table/Views Select Wizard Type ServerName **19**, **21**, Specify the OTD Name Supported Operating Systems **6**,

### U

User 19, 21, 27

### W

WebLogic and WebSphere Application Server Support 7