## Contents

**Introducing the DB2 eWay**  
About DB2 Universal Database  
About the DB2 Universal Database eWay  
What’s New in This Release  
About This Document  
What’s in This Document  
Scope  
Intended Audience  
Text Conventions  
Related Documents  
Sun Microsystems, Inc. Web Site  
Documentation Feedback  

**Installing the DB2 eWay**  
Before You Install  
Installing the DB2 eWay  
Installing the DB2 eWay on an eGate Supported System  
Adding the eWay to an Existing Sun Java Composite Applications Platform Suite Installation  
After You Install  
Extracting the Sample Projects and Javadocs  
Installing eWay Enterprise Manager plug-ins  
Viewing Alert Codes  

**Configuring the DB2 eWay**  
Creating and Configuring the DB2 eWay  
Selecting DB2 as the External Application  
Configuring the DB2 eWay Properties  
Transaction Support Levels Between Different Versions  
Using the Properties Editor  
DB2 eWay Connectivity Map Properties  
Outbound Connectivity Map Properties  
Properties in the Outbound eWay Windows /Unix  
Properties in the Outbound eWay for z/OS and AS/400  
Inbound Connectivity Map Properties  
Properties in the Inbound eWay for Windows and Unix  

---

Sun SeeBeyond eWay™ Adapter for DB2 User’s Guide  
Sun Microsystems, Inc.
Sun SeeBeyond eWay™ Adapter for DB2 User’s Guide

Properties in the Inbound eWay for z/OS and AS/400

DB2 eWay Environment Explorer Properties

Inbound DB2 eWay Environment Properties

Outbound DB2 eWay Environment Explorer Properties

eTL Environment Properties

Alternate ID Properties

Using the DB2 OTD Wizard

Using the Database OTD Wizard

Steps to Create a New DB2 eWay OTD

Select Wizard Type

Connect to Database

Select Database Objects

Select Table/Views/Aliases

Select Procedures

Add Prepared Statements

Specify the OTD Name

Review Selections

Editing Existing OTDs

Using DB2 Operations

DB2 eWay Database Operations (BPEL)

Activity Input and Output

DB2 eWay Database Operations (JCD)

The Table

The Query (Select) Operation

The Insert Operation

The Update Operation

The Delete Operation

The Stored Procedure

Executing Stored Procedures

Manipulating the ResultSet and Update Count Returned by Stored Procedure

Prepared Statement

Batch Operations

Implementing DB2 eWay Sample Projects

About the DB2 eWay Sample Projects

Operations Used in the DB2 Sample Projects

Assigning Operations in JCD

Assigning Operations in BPEL

About the eInsight Engine and eGate Components

Steps Required to Run the Sample Projects

Running the SQL Script
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Importing a Sample Project</td>
<td>66</td>
</tr>
<tr>
<td><strong>Building and Deploying the prjDB2_BPEL Sample Project</strong></td>
<td>67</td>
</tr>
<tr>
<td>Creating a Project</td>
<td>67</td>
</tr>
<tr>
<td>Creating the OTDs</td>
<td>67</td>
</tr>
<tr>
<td>Creating the Business Process</td>
<td>69</td>
</tr>
<tr>
<td>Creating the Business Process Flow</td>
<td>69</td>
</tr>
<tr>
<td>Configuring the bpInsert Modeling Elements</td>
<td>70</td>
</tr>
<tr>
<td>Configuring the bpTableSelect Modeling Elements</td>
<td>76</td>
</tr>
<tr>
<td>Creating the Connectivity Map</td>
<td>83</td>
</tr>
<tr>
<td>Populating the Connectivity Map</td>
<td>83</td>
</tr>
<tr>
<td>Binding the eWay Components</td>
<td>84</td>
</tr>
<tr>
<td>Creating an Environment</td>
<td>85</td>
</tr>
<tr>
<td>Configuring the eWays</td>
<td>86</td>
</tr>
<tr>
<td>Configuring the eWay Properties</td>
<td>86</td>
</tr>
<tr>
<td>Steps required to configure the eWay properties:</td>
<td>86</td>
</tr>
<tr>
<td>Creating the Deployment Profile</td>
<td>88</td>
</tr>
<tr>
<td>Creating and Starting the Domain</td>
<td>89</td>
</tr>
<tr>
<td>Building and Deploying the Project</td>
<td>90</td>
</tr>
<tr>
<td>Running the Sample Project</td>
<td>90</td>
</tr>
<tr>
<td><strong>Creating the prjDB2_JCD Sample Project</strong></td>
<td>91</td>
</tr>
<tr>
<td>Creating a Project</td>
<td>91</td>
</tr>
<tr>
<td>Creating the OTDs</td>
<td>91</td>
</tr>
<tr>
<td>Creating a Connectivity Map</td>
<td>93</td>
</tr>
<tr>
<td>Populating the Connectivity Map</td>
<td>93</td>
</tr>
<tr>
<td>Creating the Collaboration Definitions (Java)</td>
<td>94</td>
</tr>
<tr>
<td>jcdDelete Collaboration</td>
<td>94</td>
</tr>
<tr>
<td>jcdInsert Collaboration</td>
<td>95</td>
</tr>
<tr>
<td>jcdPsSelect Collaboration</td>
<td>95</td>
</tr>
<tr>
<td>jcdTableSelect Collaboration</td>
<td>96</td>
</tr>
<tr>
<td>jcdUpdate Collaboration</td>
<td>96</td>
</tr>
<tr>
<td>Create the Collaboration Business Rules</td>
<td>97</td>
</tr>
<tr>
<td>Creating the jcdDelete Business Rules</td>
<td>97</td>
</tr>
<tr>
<td>Creating the jcdInsert Business Rules</td>
<td>97</td>
</tr>
<tr>
<td>Creating the jcdPsSelect Business Rules</td>
<td>99</td>
</tr>
<tr>
<td>Creating the jcdTableSelect Business Rules</td>
<td>101</td>
</tr>
<tr>
<td>Creating the jcdUpdate Business Rules</td>
<td>103</td>
</tr>
<tr>
<td>Binding the eWay Components</td>
<td>103</td>
</tr>
<tr>
<td>Creating an Environment</td>
<td>104</td>
</tr>
<tr>
<td>Configuring the eWays</td>
<td>105</td>
</tr>
<tr>
<td>Configuring the eWay Properties</td>
<td>106</td>
</tr>
<tr>
<td>Configuring the Environment Explorer Properties</td>
<td>107</td>
</tr>
<tr>
<td>Configuring the Integration Server</td>
<td>108</td>
</tr>
<tr>
<td>Creating the Deployment Profile</td>
<td>108</td>
</tr>
<tr>
<td>Creating and Starting the Domain</td>
<td>109</td>
</tr>
<tr>
<td>Building and Deploying the Project</td>
<td>110</td>
</tr>
<tr>
<td>Running the Sample</td>
<td>110</td>
</tr>
<tr>
<td><strong>Common Data Type Conversions</strong></td>
<td>111</td>
</tr>
<tr>
<td>Common Data Type Conversions</td>
<td>111</td>
</tr>
</tbody>
</table>
Index
Introducing the DB2 eWay

This document describes how to install and configure the DB2 Universal Database (UDB) eWay.

What’s in This Chapter

- “About DB2 Universal Database” on page 7
- “About the DB2 Universal Database eWay” on page 8
- “What’s New in This Release” on page 8
- “About This Document” on page 8
- “Related Documents” on page 10
- “Sun Microsystems, Inc. Web Site” on page 10
- “Documentation Feedback” on page 10

1.1 About DB2 Universal Database

A database consists of a collection of information that is organized so that it can be easily accessed, managed, and updated. DB2 Universal Database is a database that handles the development and deployment of critical solutions such as:

- On demand business
- Business intelligence
- Content management
- Enterprise Resource Planning
- Customer Relationship Management

DB2 reduces the complexity of data management by eliminating, simplifying, and automating tasks associated with maintaining an enterprise-class database. It provides a foundation of information integration technologies, including federation, replication, Web services, and XML.
1.2 About the DB2 Universal Database eWay

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

Note: The DB2 Universal Database eWay connects to DB2 via the DataDirect driver which is packaged with the eWay.

1.3 What’s New in This Release

The DB2 Universal Database eWay includes:

- **Version control:** An enhanced version control system allows you to effectively manage changes to the eWay components.

- **Multiple Drag-and-Drop Component Mapping from the Deployment Editor:** The Deployment Editor now allows you to select multiple components from the Editor’s component pane, and drop them into your Environment component.

- **Support for Runtime LDAP Configuration:** Configuration properties now support LDAP key values.

- **Connection Retry Support:** Allows you to specify the number of attempts to reconnect, and the interval between retry attempts, in the event of a connection failure.

- **Editable OTD Support:** An existing OTD can be edited and saved using the OTD Wizard. This allows you to make minor changes to an OTD without having to completely recreate the OTD from scratch. The OTD is then rebuilt, saved, and then relaunched back to the same Java Collaboration or BPEL.

- **Relaunchable OTD Support:** An OTD can be rebuilt and saved (under the same name) then relaunched back to the same Java Collaboration or BPEL. This allows you to change the metadata in an OTD without having to completely recreate the business logic from scratch.

- **Connectivity Map Generator:** Generates and links your Project’s Connectivity Map components using a Collaboration or Business Process.

- **Added support:** Support for DB2 version V5R2 and V5R3 has been added when connecting to DB2 running on an AS/400 operating system.

1.4 About This Document

This guide explains how to install, configure, and operate the Sun Java Composite Application Platform Suite (CAPS) DB2 eWay Intelligent Adapter, referred to as the DB2 eWay throughout this guide.
1.4.1 What’s in This Document

This document includes the following chapters:

- **Chapter 1 “Introducing the DB2 eWay”**: Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 “Installing the DB2 eWay”**: Describes the system requirements and provides instructions for installing the DB2 Universal Database eWay.
- **Chapter 3 “Configuring the DB2 eWay”**: Provides instructions for configuring the eWay to communicate with your legacy systems.
- **Chapter 4 “Using the DB2 OTD Wizard”**: Provides information about .sag files and using the DB2 Universal Database eWay wizard.
- **Chapter 5 “Using DB2 Operations”**: Provides database operations used to access the DB2 eWay database through activities in BPEL, or through methods called from a JCD Collaboration.
- **Chapter 6 “Implementing DB2 eWay Sample Projects”**: Provides instructions for installing and running the sample Projects.
- **Appendix A “Common Data Type Conversions”**: Provides conversions between the DB2 eWay and OTD/Java datatypes and the value or size of the data element that can be used.

1.4.2 Scope

This document describes the process of installing, configuring, and running the DB2 Universal Database eWay.

This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the DB2 Universal Database eWay Javadoc files from the Enterprise Manager.

1.4.3 Intended Audience

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

1.4.4 Text Conventions

The following conventions are observed throughout this document.
Table 1  Text Conventions

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

1.5  Related Documents

The following Sun documents provide additional information about the product suite:

- *eGate Integrator User’s Guide*
- *Sun Java Composite Application Platform Suite Installation Guide*

1.6  Sun Microsystems, Inc. Web Site

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

**http://www.sun.com**

1.7  Documentation Feedback

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

**CAPS_docsfeedback@sun.com**
Chapter 2

Installing the DB2 eWay

This chapter describes how to install the DB2 Universal Database eWay.

What’s in This Chapter

- “Before You Install” on page 11
- “Installing the DB2 eWay” on page 11
- “After You Install” on page 13
- “Extracting the Sample Projects and Javadocs” on page 13
- “Installing eWay Enterprise Manager plug-ins” on page 13

2.1 Before You Install

Open and review the Readme.txt file for the DB2 Universal Database eWay for any additional information or requirements, prior to installation. The Readme.txt file is located on the installation CD-ROM.

2.2 Installing the DB2 eWay

The Enterprise Manager, a web-based application, is used to select and upload eWays and add-on files during the installation process. The following section describes how to install the components required for this eWay.

Refer to the readme for the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

Note: When the Repository is running on a UNIX operating system, the eWays are loaded from the Enterprise Manager running on a Windows platform connected to the Repository server using Internet Explorer.
2.2.1 Installing the DB2 eWay on an eGate Supported System

After you have installed Core Products, do the following:

1. From the Composite Application Platform Suite Installer, click on the Click to install additional products link (on the Administration tab).
2. Expand the eWay option.
3. From Select Sun Java Composite Applications Platform Suite Products to Install, select the products for your Sun Java Composite Applications Platform Suite and include the following:
   - FileeWay (the File eWay is used by most sample Projects)
   - DB2eWay
4. Once you have selected all of your products, click Next in the top-right or bottom-right corner of the Select Sun Java Composite Applications Platform Suite Products to Install box.
5. From the Selecting Files to Install box, locate and select your first product’s SAR file. Once you have selected the SAR file, click Next. Follow this procedure for each of your products. The Installing Files window appears after the last SAR file has been selected.
6. From the Installing Files window, review the product list. If it is correct, Click Install Products. The Enterprise Manager starts the installation.
7. When your product’s installation is completed, click on the prompt, “When installation completes, click here to continue.”

To upload the Sun SeeBeyond eWay™ Adapter for DB2 User’s Guide, Help file, Javadoc, Readme, and sample Projects, do the following:

A. Expand the Documentation option.
B. Select DB2eWayDocs.
C. Click Next in the top-right or bottom-right corner of the Select Sun Java Composite Applications Platform Suite Products to Install box.

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

It is possible to add the eWay to an existing Sun Java Composite Applications Platform Suite installation.

Steps required to add an eWay to an Existing CAPS installation include:

1. Complete steps 1 through 6 on Installing the DB2 eWay on an eGate Supported System on page 12.
2. Open the Enterprise Designer and select Update Center from the Tools menu. The Update Center Wizard appears.
3. For Step 1 of the wizard, simply click Next.
For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.

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

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

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

---

### 2.3 After You Install

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.

---

### 2.4 Extracting the Sample Projects and Javadocs

The DB2 eWay includes sample Projects and Javadocs. The sample Projects are designed to provide you with a basic understanding of how certain database operations are performed using the eWay, while Javadocs provide a list of classes and methods exposed in the eWay.

**Steps to extract the Sample Projects and Javadocs include:**

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

2. Click **Download Sample** to open the **DB2_eWay_Sample.zip** file.

3. Use WinZip to extract the sample files to a desired location. Note that the **DB2_eWay_Sample.zip** file contains two additional ZIP files for each sample Project.

4. Click Download Javadocs to open the **DB2_eWay_Javadoc.zip** file.

5. Use WinZip to extract the Javadocs files to a desired location.

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

Refer to **Importing a Sample Project** on page 66 for instructions on importing the sample Project into your repository via the Enterprise Designer.

---

### 2.5 Installing eWay Enterprise Manager plug-ins

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

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

The eWay Enterprise Manager plug-ins are available from the List of Components to Download under the CAPS Suite Installer’s DOWNLOADS tab.

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

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

To add plug-ins from the Enterprise Manager

1. From the Enterprise Manager’s Explorer toolbar, click configuration.
2. Click the Web Applications Manager tab, go to the Auto-Install from Repository sub-tab, and connect to your Repository.
3. Select the application plug-ins you require, and click Install. The application plug-ins are installed and deployed.

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

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

Viewing Alert Codes

You can view and delete alerts using the Enterprise Manager. An alert is triggered when a specified condition occurs in a Project component. The purpose of the alert is to warn the administrator or user that a condition has occurred.

To View the eWay Alert Codes

1. Add the eWay Enterprise Manager plug-in for this eWay.
2. From the Enterprise Manager’s Explorer toolbar, click configuration.
3. Click the Web Applications Manager tab and go to the Manage Alert Codes sub-tab.
4. Browse for and select the Alert Properties File for the application plug-in that you added. The Alert Properties Files are located in the alertcodes folder of your CAPS installation directory.
5. Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of available alert codes is displayed in Table 2.

### Table 2  Alert Codes for the DB2 eWay

<table>
<thead>
<tr>
<th>Alert Code Description</th>
<th>Description Details</th>
<th>User Actions</th>
</tr>
</thead>
</table>
| **DBCOMMON-CONNECT-FAILED000001=Failed to connect to database {0} on host {1}. Reason: The Pooled connection could not be allocated: [{2}]** | Occurs during the initial database connection. | - Database is down; start your database.  
- External configuration information is invalid. You may need to verify the following:  
  - Server name  
  - Database name  
  - User  
  - Password  
  - Port |
| **DBCOMMON-CONNECT-FAILED000002=Operation failed because of a database connection error. Reason: [{0}]** | Occurs while retrieving a connection from the database or connection pool. | - Verify that the database has not terminated with unexpected errors. |
| **DBCOMMON-CONNECT-FAILED000005=Connection handle not usable. Reason: [{0}]** | The connection in the pool is stale and unusable. | - Probably a database restart occurred causing the connection to be stale, retry the operation after the database is up. |
| **DBCOMMON-XARESOURCE-FAILED000001=Unable to get XAResource for the database. Reason: [{0}]** | Could not obtain XAResource for the connection. | - Check if the database supports XA and has been configured for Distributed Transactions. |
| **DBCOMMON-XACONNECT-FAILED000001=Failed to connect to database {0} on host {1}. The XA connection could not be allocated: Reason [{2}]** | Occurs during the initial database connection. | - Check if the database is configured for XA and if the database is running.  
- External configuration information is invalid. You may need to verify the following:  
  - Server name  
  - Database name  
  - User  
  - Password  
  - Port |
<p>| <strong>DBCOMMON-XSTART-FAILED000001=Unable to perform XAStart for the connection. Reason: [{0}]</strong> | A connection error has occurred which caused XAStart to fail. | - Check if the database is running, and there are no network issues. |
| <strong>DBCOMMON-XAEND-FAILED000001=XAEnd failed. Reason: [{0}]</strong> | Error occurred during commit on XA connection. | - Look for the detailed error mentioned in the alert for the appropriate action. |</p>
<table>
<thead>
<tr>
<th>Alert Code\Description</th>
<th>Description Details</th>
<th>User Actions</th>
</tr>
</thead>
</table>
| DBCOMMON-CANNOT-GET-ISOISATION-LEVEL=Unable to get isolationLevel for the transaction. Reason: [{0}] | Could not read transaction isolation information of the connection. | - Transaction isolation is one of the following constants:
  - Connection.TRANSACTION_READ_UNCOMMITTED
  - Connection.TRANSACTION_READ_COMMITTED
  - Connection.TRANSACTION_REPEATABLE_READ
  - Connection.TRANSACTION_SERIALIZABLE
  - Connection.TRANSACTION_NONE

Note: Confirm with the vendor that the getIsolation() method of the connection is implemented correctly. |

An alert code is a warning that an error has occurred. It is not a diagnostic. The user actions noted above are just some possible corrective measures you may take. Refer to the log files for more information. For information on Managing and Monitoring alert codes and logs, see the Sun SeeBeyond eGate Integrator System Administration Guide.
Chapter 3

Configuring the DB2 eWay

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

What’s in This Chapter

- “Creating and Configuring the DB2 eWay” on page 17
- “DB2 eWay Connectivity Map Properties” on page 22
- “DB2 eWay Environment Explorer Properties” on page 24

3.1 Creating and Configuring the DB2 eWay

All eWays contain a set of parameters with properties that are unique to that eWay type. The DB2 eWay properties are modified from these locations:

- **Connectivity Map**: These parameters most commonly apply to a specific component eWay, and may vary from other eWays (of the same type) in the Project.

- **Environment Explorer**: These parameters are commonly global, applying to all eWays (of the same type) in the Project. The saved properties are shared by all eWays in the DB2 External System window.

- **Collaboration or Business Process**: DB2 eWay properties may also be set from your Collaboration or Business process, in which case the settings will override the corresponding properties in the eWay’s configuration file. Any properties that are not overridden retain their configured default settings.

3.1.1 Selecting DB2 as the External Application

To create a DB2 eWay you must first create a DB2 External Application in your Connectivity Map. DB2 eWays are located between a DB2 External Application and a Service. Services are containers for Collaborations, Business Processes, eTL processes, and so forth.

**To create the DB2 External Application**

1. From the Connectivity Map toolbar, click the **External Applications** icon.

2. Select the **DB2 External Application** from the menu (see Figure 1). The selected DB2 External Application icon now appears on the Connectivity Map toolbar.
Drag the new **DB2 External Application** from the toolbar onto the Connectivity Map canvas. This icon now represents an external DB2 system.

From the Connectivity Map, you can associate (bind) the External Application to the Service to establish an eWay (see Figure 2).

When DB2 is selected as the External Application, it automatically applies the default DB2 eWay properties, provided by the OTD, to the eWay that connects it with the Service. These properties can then be or modified for your specific system using the **Properties Editor**.

### 3.1.2 Configuring the DB2 eWay Properties

A Project’s eWay properties can be modified after the eWay has been established in the Connectivity Map and the Environment has been created.

**Configuring the DB2 eWay (Connectivity Map) Properties and setting the type of transaction support**

1. On the Enterprise Designer’s Connectivity Map, double-click the DB2 eWay icon. The eWay Connections window appears.
2. Select a transaction support level from the list and click **OK**.
The eWay Connection choices include:

- **Outbound DB2 eWay (for Windows/Unix & z/OS and AS/400):** Also referred to as LocalTransaction, this support level is opposite to NoTransaction, and this means that the transaction, when aborted, will roll back all changes made since the beginning of the transaction.

- **Outbound DB2 XA-eWay:** Also referred to as XATransaction, this support level allows two-phase commit. This means that the transaction, when aborted, will roll back all changes when one of the updates fails. The update could occur in the database eWay or other eWays that support XA. Additionally, the Collaboration can contain only the database eWay, or a combination of database eWay and other eWays that support XA.

- **Outbound DB2 non-Transactional eWay (for Windows/Unix & z/OS and AS/400):** Also referred to as NoTransaction, this support level indicates that the Collaboration does not support transactions. This means that when a transaction aborts, there is no ability to roll back any changes to the previous update.

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

**Configuring the DB2 eWay (Environment Explorer) Properties**

1. From the Environment Explorer tree, right-click the DB2 External System. Select Properties from the shortcut menu. The Properties Editor opens with the DB2 eWay Environment properties.

2. Make any necessary changes to the Environment property values, and click OK to save the settings.

**3.1.3 Transaction Support Levels Between Different Versions**

The types of transaction support levels used in Java CAPS 5.1 may be different from the support levels used in Java CAPS 5.1.1. Projects that are imported from a Java CAPS 5.1.0 version can potentially display different results, depending on whether the 5.1.0 Java Collaboration Definition (JCD) included multiple (insert/update/delete) operations. This only affects non-XA transactions. If you are using an XA transaction, then you can skip this section.
Example:

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

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

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

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

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

C. Rebuild the Project.

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

Figure 4  Transaction Support Levels
Under the scenario noted above, if you want 5.1.1 behavior for a LocalTransaction, then set your eWay connection to be Outbound DB2 non-Transactionable eWay (NoTransaction).

### 3.1.4 Using the Properties Editor

Modifications to the eWay configuration properties are made from the DB2 eWay Properties Editor.

A description of each property is displayed in the Description pane when it is selected. This provides a brief explanation of the required settings or options.

The Comments pane provides an area to record notes and information regarding the currently selected property. These comments are saved when you close the editor.

#### Modifying the Default eWay Configuration Properties

1. From the Connectivity Map or the Environment Explorer, open the Properties Editor to the DB2 eWay default properties.
2. From the upper-right pane of the Properties Editor, select a subdirectory of the configuration directory. The parameters contained in that subdirectory are now displayed in the Properties pane of the Properties Editor. For example, if you click on the connector subdirectory, the editable connector parameters are displayed in the right pane (see Figure 5).

![Figure 5 Properties Editor -- DB2 eWay Properties](image)

3. Click on any property field to make it editable. For example, click on the class property to edit the class value. If a property value is true/false or multiple choice, the field displays a submenu of property options.
4. Click on the ellipsis (…) in the properties field to open a separate configuration dialog box. This is helpful for large values that cannot be fully displayed in the
parameter’s property field. Enter the property value in the dialog box and click OK. The value is now displayed in the property field.

5 After modifying the configuration properties, click OK to close the Properties Editor and save your changes.

3.2 DB2 eWay Connectivity Map Properties

The DB2 eWay configuration parameters, accessed from the Connectivity Map, are organized into the following sections:
- **Outbound Connectivity Map Properties** on page 22
- **Inbound Connectivity Map Properties** on page 23

3.2.1 Outbound Connectivity Map Properties

The Outbound configuration parameters, accessed from the Connectivity Map, are organized into the following sections:
- **Properties in the Outbound eWay Windows /Unix** on page 22
- **Properties in the Outbound eWay for z/OS and AS/400** on page 22

Properties in the Outbound eWay Windows /Unix

The **JDBC Connector Settings** section of the DB2 Connectivity Map properties contains the top-level parameters displayed in Table 3.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description for the database.</td>
<td>A valid string. The default is <strong>DB2 Connection Pool Datasource</strong>.</td>
</tr>
<tr>
<td>ClassName</td>
<td>Displays the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.</td>
<td>A valid class name. The default is <strong>com.SeeBeyond.db2.jdbcx.db2.DB2DataSource</strong></td>
</tr>
</tbody>
</table>

*Note: Do not change this value.*

Properties in the Outbound eWay for z/OS and AS/400

The **JDBC Connector Settings** section of the DB2 Connectivity Map properties contains the top-level parameters displayed in Table 4.
### Table 4  Outbound z/OS and AS/400 Connectivity Map JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description for the database.</td>
<td>A valid string. The default is DB2 Datasource For AS400/zOS.</td>
</tr>
<tr>
<td>ClassName</td>
<td>Displays the Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface.</td>
<td>A valid class name. The default is com.SeeBeyond.db2.jdbcx.db2.DB2DataSource. Note: Do not change this value.</td>
</tr>
</tbody>
</table>

#### 3.2.2 Inbound Connectivity Map Properties

The Inbound configuration parameters, accessed from the Connectivity Map, are organized into the following sections:

- Properties in the Inbound eWay for Windows and Unix on page 23
- Properties in the Inbound eWay for z/OS and AS/400 on page 23

### Properties in the Inbound eWay for Windows and Unix

The Parameter Settings section of the DB2 Connectivity Map properties contains the top-level parameters displayed in Table 5.

#### Table 5  Inbound eWay Connectivity Map Parameter Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pollmilliseconds</td>
<td>Polling interval in milliseconds.</td>
<td>A valid numeric value. The default is 5000.</td>
</tr>
<tr>
<td>PreparedStatement</td>
<td>The Prepared Statement used for polling against the database.</td>
<td>The Prepared Statement must be the same Statement you created using the Database OTD Wizard. Only a SELECT Statement is allowed. Additionally, no place holders should be used. This is a SQL statement that cannot contain any input data (i.e. you cannot use &quot;?&quot; in the Prepared Query).</td>
</tr>
</tbody>
</table>

#### Properties in the Inbound eWay for z/OS and AS/400

The Parameter Settings section of the DB2 Connectivity Map properties contains the top-level parameters displayed in Table 6.
3.3 DB2 eWay Environment Explorer Properties

The DB2 eWay configuration parameters, accessed from the Environment Explorer tree, are organized into the following sections:

- **Inbound DB2 eWay Environment Properties** on page 25
- **Outbound DB2 eWay Environment Explorer Properties** on page 26
- **eTL Environment Properties** on page 33

### Table 6 Inbound eWay Connectivity Map Parameter Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pollmilliseconds</td>
<td>Polling interval in milliseconds.</td>
<td>A valid numeric value. The default is 5000.</td>
</tr>
<tr>
<td>PreparedStatement</td>
<td>The Prepared Statement used for polling against the database.</td>
<td>The Prepared Statement must be the same Statement you created using the Database OTD Wizard. Only a SELECT Statement is allowed. Additionally, no place holders should be used. This is a SQL statement that cannot contain any input data (i.e. you cannot use &quot;?&quot; in the Prepared Query).</td>
</tr>
</tbody>
</table>
3.3.1 **Inbound DB2 eWay Environment Properties**

The Inbound DB2 eWay properties, accessed from the Environment Explorer tree, are organized into the following sections:

- **Inbound Environment Explorer Properties for Windows/Unix** on page 25
- **Inbound Environment Explorer Properties for z/OS and AS/400** on page 26

**Inbound Environment Explorer Properties for Windows/Unix**

The **Parameter Settings** section of the Inbound Windows/Unix Environment contains the top-level parameters displayed in Table 7.

**Table 7** Parameter Settings for the Inbound Environment Explorer

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description for the database.</td>
<td>A valid string. The default is <strong>DB2 DriverManager</strong>.</td>
</tr>
<tr>
<td>ServerName</td>
<td>Specifies the host name of the external database server.</td>
<td>Any valid string.</td>
</tr>
<tr>
<td>PortNumber</td>
<td>Specifies the I/O port number on which the server is listening for connection requests.</td>
<td>A valid port number. The default is 50000.</td>
</tr>
</tbody>
</table>
### 3.3.2 Outbound DB2 eWay Environment Explorer Properties

The Outbound DB2 eWay properties, accessed from the Environment Explorer tree, are organized into the following sections:

- **Outbound Environment Explorer Properties for Windows/Unix** on page 27
- **Outbound Environment Explorer Properties for z/OS and AS/400** on page 30
Outbound Environment Explorer Properties for Windows/Unix

The Outbound DB2 eWay properties for Windows/Unix, accessed from the Environment Explorer tree, are organized into the following sections:

- JDBC Connector Settings for Windows/Unix on page 27
- Connection Retry Settings for Windows/Unix on page 29

JDBC Connector Settings for Windows/Unix

The JDBC Connector Settings section of the Outbound Windows/Unix Environment contains the top-level parameters displayed in Table 9.

Table 9  Outbound eWay Windows/Unix Environment - JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description for the database.</td>
<td>A valid string. The default value is <strong>DB2 Connection Pool Datasource</strong>.</td>
</tr>
<tr>
<td>ServerName</td>
<td>Specifies the host name of the external database server.</td>
<td>Any valid string.</td>
</tr>
<tr>
<td>PortNumber</td>
<td>Specifies the I/O port number on which the server is listening for connection requests.</td>
<td>A valid port number. The default is 50000.</td>
</tr>
<tr>
<td>DatabaseName</td>
<td>Specifies the name of the database instance used on the Server.</td>
<td>Any valid string.</td>
</tr>
<tr>
<td>User</td>
<td>Specifies the user name that the eWay uses to connect to the database.</td>
<td>Any valid string.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the password used to access the database.</td>
<td>Any valid string.</td>
</tr>
</tbody>
</table>
### Table 9  Outbound eWay Windows/Unix Environment - JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriverProperties</td>
<td>Use the JDBC driver that is shipped with this eWay. The DataSource implementation may need to execute additional methods to assure a successful run. The additional methods will need to be identified in the Driver Properties.</td>
<td>The delimiter set by the user. For more information, see the Delimiter property below. Valid delimiters are: <code>&lt;method-name-1&gt;#&lt;param-1&gt;#&lt;param-2&gt;........&lt;param-n&gt;##&lt;method-name-2&gt;#&lt;param-1&gt;#&lt;param-2&gt;........&lt;param-n&gt;##......##</code>. For example: to execute the method setSpyAttributes, give the method a String for the URL <code>&quot;setSpyAttribute#&lt;url&gt;##&quot;</code>. Note: The setSpyAttributes (for Data Direct drivers) that are contained in the following examples (between the last set of double octothorps [##] within each example), are used for debugging purposes and need not be used on every occasion. Optional—if you are using Spy Log: <code>&quot;setURL#jdbc:Seebeyond:DB2://&lt;server&gt;:4100;DatabaseName=&lt;database&gt;##setSpyAttributes#log=(file)c:/temp/spy.log;logTName=yes##&quot;</code></td>
</tr>
<tr>
<td>Delimiter</td>
<td>This is the delimiter character to be used in the DriverProperties prompt.</td>
<td>The default is #. See the DriverProperties property above for more information on how the default value is used.</td>
</tr>
</tbody>
</table>
Table 9  Outbound eWay Windows/Unix Environment - JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinPoolSize</td>
<td>Specifies the minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and the new connections should be created as needed. If the pool size is too small, you may experience longer connection times due to the existing number of physical connections. A connection that stays in the pool allows transactions to use it via a logical connection which is faster.</td>
<td>A valid numeric value. The default is 0.</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Specifies the maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size depends on the transaction volume and response time. If the pool size is too big, you may end up with too many connections with the database.</td>
<td>A valid numeric value. The default is 10.</td>
</tr>
<tr>
<td>MaxIdleTime</td>
<td>The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.</td>
<td>A valid numeric value. The default is 0.</td>
</tr>
</tbody>
</table>

Connection Retry Settings for Windows/Unix

The Connection Retry Settings section of the Outbound DB2 Environment contains the top-level parameters displayed in Table 10.

Table 10  Outbound eWay Windows/Unix Environment - Connection Retry Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionRetries</td>
<td>Specifies the number of retries to establish a connection with the DB2 database upon a failure to acquire one.</td>
<td>An integer indicating the number of attempts allowed to establish a connection. The configured default is 0.</td>
</tr>
</tbody>
</table>
Table 10  Outbound eWay Windows/Unix Environment - Connection Retry Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionRetry Interval</td>
<td>Specifies the milliseconds of pause before each attempt to access the database. This setting is used in conjunction with the 'Connection Retries' setting. For example: In the event that the eWay cannot connect to the Database, the eWay will try to reconnect to the database in 5 second intervals, a total of 10 times, when the Connection Retries property is set at 10 and the Connection Retry Interval property is 5000.</td>
<td>An integer indicating the configured length of the time (in milliseconds) before each reattempt to access the destination file. The configured default is 1000 (1 second).</td>
</tr>
</tbody>
</table>

Outbound Environment Explorer Properties for z/OS and AS/400

The Outbound z/OS and AS/400 properties, accessed from the Environment Explorer tree, are organized into the following sections:

- JDBC Connector Settings for z/OS and AS/400 on page 30
- Connection Retry Settings for z/OS and AS/400 on page 32

JDBC Connector Settings for z/OS and AS/400

The JDBC Connector Settings section of the Outbound z/OS and AS/400 Environment contains the top-level parameters displayed in Table 11.

Table 11  Outbound eWay z/OS and AS/400 Environment - JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter a description for the database.</td>
<td>A valid string. The default value is <strong>DB2 Datasource For AS400/zOS</strong>.</td>
</tr>
<tr>
<td>ServerName</td>
<td>Specifies the host name of the external database server.</td>
<td>Any valid string.</td>
</tr>
<tr>
<td>LocationName</td>
<td>Specifies the Location Name for the DB2 database that is being used on AS400/zOS. You can issue a “Select current service from sysibm.sysdummy1” to the database to find out its value.</td>
<td>A valid port number. The default is 446.</td>
</tr>
<tr>
<td>PackageCollection</td>
<td>Specifies the PackageCollection for the DB2 database that is being used on AS400/zOS. Use the same one you used in the Wizard Connection Panel. For more information regarding Packages refer to <strong>Packages</strong> on page 39.</td>
<td>Any valid String.</td>
</tr>
<tr>
<td>PackageCollection</td>
<td>Specify the PackageCollection for the DB2 database that is being used on AS400/zOS.</td>
<td>Any valid String.</td>
</tr>
</tbody>
</table>
### Table 11  Outbound eWay z/OS and AS/400 Environment - JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Specifies the user name that the eWay uses to connect to the database.</td>
<td>Any valid string.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the password used to access the database.</td>
<td>Any valid string.</td>
</tr>
</tbody>
</table>
| DriverProperties     | Use the JDBC driver that is shipped with this eWay. Often times the DataSource implementation needs to execute additional methods to assure a connection. You must identify the additional methods in the Driver Properties. | The delimiter set by the user. For more information, see the Delimiter property below. Valid delimiters are:  
  
  ""<method-name-1>#<param-1>#<param-2>….<param-n>##<method-name-2>#<param-1>#<param-2>….<param-n>##……##"".  
  
  For example: to execute the method setSpyAttributes, give the method a String for the URL  
  "setSpyAttribute#<url>##".  
  
  **Note:** The setSpyAttributes (for Data Direct drivers) that are contained in the following examples (between the last set of double octothorps [##] within each example), are used for debugging purposes and need not be used on every occasion.  
  
Optional—if you are using Spy Log:  
  
  "setURLjdbc:SeeBeyond:DB2://<server>:4100;DatabaseName=<database>##setSpyAttributes##log=(file)c:/temp/spy.log;logTName=yes##"
Table 11  Outbound eWay z/OS and AS/400 Environment - JDBC Connector Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delimiter</td>
<td>This is the delimiter character to be used in the DriverProperties prompt.</td>
<td>A valid delimiter. The default is #.</td>
</tr>
<tr>
<td>MinPoolSize</td>
<td>Specifies the minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and the new connections should be created as needed. If the pool size is too small, you may experience longer connection times due to the existing number of physical connections. A connection that stays in the pool allows transactions to use it via a logical connection which is faster.</td>
<td>A valid numeric value. The default is 0.</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Specifies the maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum. The pool size depends on the transaction volume and response time. If the pool size is too big, you may end up with too many connections with the database.</td>
<td>A valid numeric value. The default is 10.</td>
</tr>
<tr>
<td>MaxIdleTime</td>
<td>The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.</td>
<td>A valid numeric value. The default is 0.</td>
</tr>
</tbody>
</table>

Connection Retry Settings for z/OS and AS/400

The Connection Retry Settings section of the Outbound z/OS and AS/400 Environment contains the top-level parameters displayed in Table 12.

Table 12  Outbound eWay z/OS and AS/400 Environment - Connection Retry Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionRetries</td>
<td>Specifies the number of retries to establish a connection with the DB2 database upon a failure to acquire one.</td>
<td>An integer indicating the number of attempts allowed to establish a connection. The configured default is 0.</td>
</tr>
</tbody>
</table>
### Table 12  Outbound eWay z/OS and AS/400 Environment - Connection Retry Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionRetry Interval</td>
<td>Specifies the milliseconds of pause before each reattempt to access the database. This setting is used in conjunction with the 'Connection Retries' setting. For example: In the event that the eWay cannot connect to the Database, the eWay will try to reconnect to the database in 5 second intervals, a total of 10 times, when the Connection Retries property is set at 10 and the Connection Retry Interval property is 5000.</td>
<td>An integer indicating the configured length of the time (in milliseconds) before each reattempt to access the destination file. The configured default is 1000 (1 second).</td>
</tr>
</tbody>
</table>

### 3.3.3 eTL Environment Properties

This section contains all the configuration properties needed for eTL collaborations. The eTL environment properties, accessed from the Environment Explorer tree, are organized in the following section:

- **Alternate ID Properties** on page 33

**Note:** The configuration properties specified in this section will not affect the functionality of DB2 eWay.

### Alternate ID Properties

This section contains all the configuration properties that allow the user to override database catalogs and/or schema names used in the sql statements generated by eTL collaborations. The **Alternate ID** section contains the top-level parameters displayed in Table 13.

### Table 13  eTL Environment - Alternate ID Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OverwriteCatalog Name</td>
<td>Indicates whether or not to overwrite the database catalog name in the sql statements generated by eTL collaboration.</td>
<td>A True or False value. If you select “True”, the content of the “CatalogNameMap” property will be used. If “False” is selected, the catalog name will be overwritten. The default is True.</td>
</tr>
<tr>
<td>CatalogNameMap</td>
<td>Specifies the key-value pair(s) for catalog name overwrites. The correct format is: <code>&lt;oldCatalogName1&gt;=&lt;newCatalogName1&gt;;&lt;oldCatalogName2&gt;=&lt;newCatalogName2&gt;;...</code></td>
<td>A semi-colon separated list that contains valid name overwrites.</td>
</tr>
</tbody>
</table>
### Table 13  eTL Environment - Alternate ID Settings

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OverwriteSchemaName</td>
<td>Indicates whether or not to overwrite the database schema name in the sql statements generated by eTL collaboration.</td>
<td>A True or False value. If you select “True”, the content of next configuration property “SchemaNameMap” will be used. If “False” is selected, the schema name will be overwritten. The default is True.</td>
</tr>
<tr>
<td>SchemaNameMap</td>
<td>Specifies the key-value pair(s) for schema name overwrites. The correct format is: <code>&lt;oldSchemaName1&gt;=&lt;newSchemaName1&gt;;&lt;oldSchemaName2&gt;=&lt;newSchemaName2&gt;...</code></td>
<td>A semi-colon separated list that contains valid name overwrites.</td>
</tr>
</tbody>
</table>
Using the DB2 OTD Wizard

This chapter describes how to use the DB2 Universal Database eWay Wizard to build OTD’s.

What’s in This Chapter

- “Using the Database OTD Wizard” on page 36
- “Steps to Create a New DB2 eWay OTD” on page 37
- “Editing Existing OTDs” on page 50

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 Java methods, refer to your JDBC developer’s reference.

Note: Database OTD’s are not messagable. For more information on messagable OTD’s, see the eGate Integrator User’s Guide.
4.2 Steps to Create a New DB2 eWay OTD

The following steps are required to create a new OTD for the DB2 Intelligent Adapter eWay.

- **Select Wizard Type** on page 37
- **Connect to Database** on page 37
- **Select Database Objects** on page 39
- **Select Table/Views/Aliases** on page 40
- **Select Procedures** on page 43
- **Add Prepared Statements** on page 46
- **Specify the OTD Name** on page 49
- **Review Selections** on page 49

4.2.1 Select Wizard Type

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

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

**Figure 7** OTD Wizard Selection

4.2.2 Connect to Database

1. **Select the Connection type** using the drop-down list (see Figure 8). The rest of the Connection Information fields displayed will depend on your selection.
Specify the applicable connection information (depending on the Connection type) for your database including:

- **Host Name** - The server where DB2 resides.
- **Port** - The port number of DB2.
- **Location** (AS/400 and z/OS specific) - The name of the DB2 subsystem. To find the location of the DB2 subsystem, use the Database Query Tool to issue the following query: `select current server from sysibm.sysdummy1`.
- **Database** (Windows/Unix specific) - The name of the database instance.
- **Collection** (AS/400 and z/OS specific) - The name that identifies a group of packages. For more information on Packages, refer to Packages on page 39.
- **User Name** - The user name that the eWay uses to connect to the database.
- **Password** - The password used to access the database.
- **Optional Parameters** field (only for AS/400 and z/OS) will display in the OTD Wizard if the `DB2_ConnectionInfo.txt` file is present in the `edesigner\usrdir\modules\ext\db2adapter` directory. This field allows additional connection parameters for the DB2 OTD wizard. Please contact SeeBeyond for more information on Optional Parameters.

**Note:** This parameter is not specific to an OTD. It’s only specific to the session where Enterprise Designer runs. When you edit the OTD, the current value that shows up is the value that was entered previously in the Wizard. It may or may not be the same value you used for creating the original OTD.
Parameter examples include:

- The `showSelectableTables=false` parameter can be set to false (default is true) to see additional tables listed under a user, such as the behavior in a previous eWay which uses an older version of the driver.
- The `AlternateID=user1` parameter allows you to set the object owner to be `user1` otherwise the user in the User Name field will be used.
- Multiple parameters can be used and must be separated by a semi-colon (;

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

### Packages

This eWay uses a DataDirect driver (previously known as Merant) to execute SQL calls in DB2. The DataDirect driver requires packages to be created in the DB2 System. Packages do not contain specific SQL statements like static SQL packages but rather dynamic sections, used like cursors to help facilitate the driver’s executing of dynamic SQL queries and returning results.

Creating packages on the server, also known as binding packages, needs only be done once. The first user of the OTD Wizard must have bind permission to create the packages. Without bind authority the user receives an error message when the driver attempts to bind the packages and they will be unable to issue any SQL call. Packages are created automatically, under the Collection ID, when the user fills in the Wizard entries. If the Collection ID is left as blank, it will generate the packages under NULLID.

The driver creates SQL packages on the database including: DDJC330A, DDJC330B, DDJC330C, etc. When connecting, the driver queries a system table to determine whether the default packages exist on the system. If none exist, the driver creates them.

**Note:** SQL applications that execute dynamic SQL against DB2 need to have packages bound on the server. In the case of some IBM native tools this may not be obvious because the packages are already installed on the database by default.

### 4.2.3 Select Database Objects

1. When selecting Database Objects, you can select any combination of **Tables**, **Views**, **Procedures**, or **Prepared Statements** you would like to include in the .otd file. Click Next to continue (see Figure 9).
2. Click Next to continue. The Select Tables/Views window appears.

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

### 4.2.4 Select Table/Views/Aliases

1. In the Select Tables/Views/Aliases window, click Add (see Figure 10).

**Figure 9** Select Database Object

**Figure 10** Select Tables/Views
2 In the **Add Tables** window, select the type of criteria to be used for your search, consisting of table data, view only data, or both. You can include system tables in your search by selecting the checkbox.

3 From the **Table/View** Name drop down list, select the location of your database table and click **Search** (see Figure 11). You can search for Table/View Names by entering a table name. The use of wildcard characters of ‘?’, and ‘*’ as part of your Table/View name search allow for greater search capabilities. For example, “AB?CD” or “AB*CD”.

4 Select the table of choice and click **OK**. The table selected is added to the Selected window (see Figure 11).

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

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

6 If you clicked **Change** on the Selected **Tables/Views** window, you can select or deselect your table columns on the **Table/View Columns** window. You can also change the data type for each table by highlighting the data type and selecting a different one from the drop down (see Figure 12).
7 Click Advanced to change the data type, precision/length, or scale. In general, do not change the precision/length or the scale. Once you have finished your table choices, click OK (see Figure 13).

**Figure 13** Tables/Views Columns - Advanced
When using Prepared Statement packages, select **Use fully qualified table/view names in the generated Java code.**

### 4.2.5 Select Procedures

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

**Figure 14** Select Procedures and specify Resultset and Parameter Information

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

3. In the resulting Procedure Selection list box, select a Procedure. Click **OK**.
4 On the **Select Procedures and specify Resultset and Parameter Information** window click **Edit Parameters** to make any changes to the selected Procedure (see Figure 16).

**Figure 15** Add Procedures

4 On the **Select Procedures and specify Resultset and Parameter Information** window click **Edit Parameters** to make any changes to the selected Procedure (see Figure 16).

**Figure 16** Procedure Parameters

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

6 To select how you would like the OTD to generate the nodes for the Resultset click **Edit Resultsets**.

7 Click **Add** to add the type of Resultset node you would like to generate.
The DBWizard provides three different ways to generate the ResultSet nodes of a Stored Procedure. They are the "By Executing", "Manually", and "With Assistance" modes.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>By Executing Mode</td>
<td>&quot;By Executing&quot; 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 &quot;By Executing&quot; mode does not return all ResultSets, one should use the other modes to generate the ResultSet nodes.</td>
</tr>
<tr>
<td>With Assistance Mode</td>
<td>&quot;With Assistance&quot; 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 &quot;Assist&quot; mode, highlight the execute statement up to and including the table name(s) before executing the query.</td>
</tr>
</tbody>
</table>
4.2.6 Add Prepared Statements

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

Prepared statements can be used to perform insert, update, delete and query operations. A prepared statement uses a question mark (?) as a place holder for input. For example: insert into EMP_TAB (Age, Name, Dept No) values (?, ?, ?)

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

Steps Required to Add Prepared Statements Include:

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

You can process a resultset by looping through the `next()` method. For more information, see The Query (Select) Operation on page 51.

1 On the Add Prepared Statements window, click Add.
2. Enter the name of a Prepared Statement or create a SQL statement by clicking in the SQL Statement window. When finished creating the statement, click **Save As** giving the statement a name. This name appears as a node in the OTD. Click **OK** (see Figure 19).

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.

**Note:** When doing a Prepared Statement with two or more tables, where multiple tables have the same column name, you must put the table name qualifier in the Prepared Statement to build the OTD.

4. Click **Add** if you want to add additional parameters to the Statement or highlight a row and click **Remove** to remove it. Click **OK** (see Figure 20).
5. To edit the ResultSet Columns, click Edit ResultSet Columns. Both the Name and Type are editable but it is recommend you do not change the Name. Doing so will cause a loss of integrity between the ResultSet and the Database. Click OK (see Figure 21).

**Note:** The OTD Wizard fails to create OTDs with complex prepared statements that use the same column name in different tables. This problem is resolved by modifying the SQL statement to use column name aliases.

6. On the Add Prepared Statements window, click OK.
4.2.7 Specify the OTD Name

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

Steps Required to Specify the OTD Name:

1. Enter a name for the OTD. The OTD contains the selected tables and the package name of the generated classes (see Figure 22).

   **Figure 22** Naming the OTD

   ![Figure 22](image)

2. Click Next.

4.2.8 Review Selections

Review the selections made for the new OTD.

Steps Required to Review Your OTD Selections:

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

2. If you are satisfied with the OTD information, click Finish to begin generating the OTD (see Figure 23).
4.3 Editing Existing OTDs

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

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

To Edit an Existing OTD

When a minor change is needed for an existing OTD, there is no need to rebuild it from scratch; instead, you can edit the OTD. To edit an OTD, complete the following steps:

1. In the Enterprise Explorer, right-click on the OTD. From the submenu, click Edit (see Figure 24). The Database Connection Information Wizard opens.
2 Connect to the DB2 database by entering the applicable information in the wizard. Once the connection is established, the Database Wizard opens, allowing you to make modifications to the OTD.

3 Once you have completed editing the OTD, click the Finish button to save the changes.

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

When editing an OTD, you can connect to another instance of the database under the following conditions:

- The same type of DB2 database must be used. Because of incompatibility of certain features in the databases, switching between DB2 databases that run on z/OS, AS/400 and Windows/UNIX is not supported.
- The same version of the database should be used unless the newer version is compatible with the older version.
- Tables in the database must be defined with the same definition.
- The stored procedures must be identical.
- For tables/stored procedures built with ‘qualified-name’, the schema name for the tables/stored procedures must be identical in both database instances.
Chapter 5

Using DB2 Operations

The database operations used in the DB2 eWay are used to access the DB2 database. Database operations are either accessed through Activities in BPEL, or through methods called from a JCD Collaboration.

What’s in This Chapter

- DB2 eWay Database Operations (BPEL) on page 52
- DB2 eWay Database Operations (JCD) on page 54

5.1 DB2 eWay Database Operations (BPEL)

The DB2 eWay uses a number operations to query the DB2 database. Within a BPEL business process, the DB2 eWay uses BPEL Activities to perform basic outbound database operations, including:

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

In addition to these outbound operations, the DB2 eWay also employs the inbound Activity ReceiveOne within a Prepared Statement OTD.

5.1.1 Activity Input and Output

The Sun SeeBeyond Enterprise Designer – Business Rules Designer includes Input and Output columns to map and transform data between Activities displayed on the Business Process Canvas.

Figure 25 displays the business rules between the FileClient.write and otdDB2.Db_employeeDelete Activities. In this example, the whereClause appears on the Input side.
Figure 25  Input and Output Between Activities

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

<table>
<thead>
<tr>
<th>eInsight Operation</th>
<th>Activity Input</th>
<th>Activity Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectAll</td>
<td>where() clause (optional)</td>
<td>Returns all rows that fit the condition of the where() clause.</td>
</tr>
<tr>
<td>SelectMultiple</td>
<td>number of rows where() clause (optional)</td>
<td>Returns the number of rows specified that fit the condition of the where() clause, and the number of rows to be returned.</td>
</tr>
<tr>
<td>SelectOne</td>
<td>where() clause (optional)</td>
<td>Returns the first row that fits the condition of the where() clause.</td>
</tr>
<tr>
<td>Insert</td>
<td>definition of new item to be inserted</td>
<td>Returns status.</td>
</tr>
<tr>
<td>Update</td>
<td>where() clause</td>
<td>Returns status.</td>
</tr>
<tr>
<td>Delete</td>
<td>where() clause</td>
<td>Returns status.</td>
</tr>
</tbody>
</table>
5.2 **DB2 eWay Database Operations (JCD)**

The same database operations are also used in the JCD, but appear as methods to call from the Collaboration.

Tables, Views, and Stored Procedures are manipulated through OTDs. Methods to call include:

- `insert()`
- `insertRow()`
- `update(String sWhere)`
- `updateRow()`
- `delete(String sWhere)`
- `deleteRow()`
- `select(String where)`

**Note:** Refer to the Javadoc for a full description of methods included in the DB2 eWay.

5.2.1 **The Table**

A table OTD represents a database table. It consists of fields and methods. Fields correspond to the columns of a table while methods are the operations that you can apply to the OTD. This allows you to perform query, update, insert, and delete SQL operations in a table. The ability to update via a resultset is called “Updatable Resultset”, which is a feature supported by this eWay.

By default, the Table OTD has UpdatableConcurrency and ScrollTypeForwardOnly. Normally you do not have to change the default setting.

The type of result returned by the `select()` method can be specified using:

- SetConcurrenytoUpdatable
- SetConcurrenytoReadOnly
- SetScrollTypetoForwardOnly
- SetScrollTypetoScrollSensitive
- SetScrollTypetoInsensitive

**The Query (Select) Operation**

To perform a query operation on a table

1. Execute the `select()` method with the “where” clause specified if necessary.

   **Note:** The content of the `input.getText()` file may contain null, meaning it will not have a “where” clause or it can contain a “where” clause such as `empno > 50`.

2. Loop through the ResultSet using the `next()` method.
3 Process the return record within a `while()` loop.

For example:

```java
class jcdTableSelect {
    public void receive(com.stc.connector.appconn.file.FileTextMessage input, dtd.otdOutputDTD1325973702.DB_Employee otdOutputDTD_DB_Employee_1, otdDB2.OtdDB2OTD otdDB2_1, com.stc.connector.appconn.file.FileApplication FileClient_1) throws Throwable {
        FileClient_1.setText("Selectiong records from db_employee table via Table Select........");
        FileClient_1.write();
        otdDB2_1.getDb_employee().select(input.getText());
        while (otdDB2_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo(typeConverter.shortToString(otdDB2_1.getDb_employee().getEMP_NO(), "#", false, "") );
            otdOutputDTD_DB_Employee_1.setLastname(otdDB2_1.getDb_employee().getLAST_NAME());
            otdOutputDTD_DB_Employee_1.setFirstname(otdDB2_1.getDb_employee().getFIRST_NAME());
            otdOutputDTD_DB_Employee_1.setRate(otdDB2_1.getDb_employee().getRATE().toString());
            otdOutputDTD_DB_Employee_1.setLastDate(typeConverter.dateToString(otdDB2_1.getDb_employee().getLAST_UPDATE(), "yyyy-MM-dd hh:mm:ss", false, "") );
            FileClient_1.setText(otdOutputDTD_DB_Employee_1.marshalToString());
            FileClient_1.write();
        }
        FileClient_1.setText("Table Select Done.");
        FileClient_1.write();
    }
}
```

The Insert Operation

To perform an insert operation on a table

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

This example inserts an employee record.
package prjDB2_JCDjcdALL;

public class jcdInsert
{
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
        com.stc.connector.appconn.file.FileTextMessage input,
        otdDB2.OtdDB2OTD otdDB2_1, dtd.otdInputDTD_1206505729.DB_Employee otdInputDTD_DB_Employee_1,
        com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Inserting records in to db_employee table....... " );
        FileClient_1.write();
        otdInputDTD_DB_Employee_1.unmarshalFromString( input.getText() );
        otdDB2_1.getDb_employee().insert();
        for (int i1 = 0; i1 < otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
            otdDB2_1.getDb_employee().setEMP_NO( typeConverter.stringToShort( otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#", false, 0 ) );
            otdDB2_1.getDb_employee().setLAST_NAME( otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
            otdDB2_1.getDb_employee().setFIRST_NAME( otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
            otdDB2_1.getDb_employee().setRATE( new java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getRate() ) );
            otdDB2_1.getDb_employee().setLAST_UPDATE( typeConverter.stringToTimestamp( otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-MM-dd hh:mm:ss", false, "" ) );
            otdDB2_1.getDb_employee().insertRow();
        }
        FileClient_1.setText( "Insert Done." );
        FileClient_1.write();
    }
}

The Update Operation

To perform an update operation on a table

1. Execute the update() method.

   Note: The content of the input.getText() file may contain null, meaning it will not have a "where" clause or it can contain a "where" clause such as empno > 50.

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()`.

```java
package prjDB2_JCDjcdALL;

public class jcdUpdate
{
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
            com.stc.connector.appconn.file.FileTextMessage input,
            otdDB2.OtdDB2OTD otdDB2_1,
            com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Updating the Rate and Last_update fields .. " );
        FileClient_1.write;
        otdDB2_1.getDb_employee().update( input.getText() );
        while (otdDB2_1.getDb_employee().next()) {
            otdDB2_1.getDb_employee().set_LAST_NAME( "Krishna" );
            otdDB2_1.getDb_employee().set_FIRST_NAME( "Kishore" );
            otdDB2_1.getDb_employee().updateRow();
        }
        FileClient_1.setText( "Update Done." );
        FileClient_1.write();
    }
}
```

The Delete Operation

To perform a delete operation on a table:

1. Execute the `delete()` method.

**Note:** The content of the `input.getText()` file may contain null, meaning it will not have a "where" clause or it can contain a "where" clause such as `empno > 50`.

In this example DELETE an employee.

```java
package prjDB2_JCDjcdALL;

public class jcdDelete
{
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
            com.stc.connector.appconn.file.FileTextMessage input,
            otdDB2.OtdDB2OTD otdDB2_1,
            com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
    }
```
5.2.2 The Stored Procedure

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

Executing Stored Procedures

The OTD represents the Stored Procedure “LookUpGlobal” with two parameters, an inbound parameter (INLOCALID) 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:

```java
package Storedprocedure;

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

    public void receive(
        com.stc.connector.appconn.file.FileTextMessage input,
        com.stc.connector.appconn.file.FileApplication FileClient_1,
        employeedb.Db_employee employeedb_with_top_db_employee_1,
        insert_DB.Insert_DBOTD insert_DB_1)
    throws Throwable
    {
        employeedb_with_top_db_employee_1.unmarshalFromString(input.getText());
        insert_DB_1.getInsert_new_employee().setEmployee_no(
            java.lang.Integer.parseInt(employeedb_with_top_db_employee_1.getEmployee_no())
        );
```
insert_DB_1.getInsert_new_employee().setEmployee_Lname(employeedb_with_top_db_employee_1.getEmployee_lname());
insert_DB_1.getInsert_new_employee().setEmployee_Fname(employeedb_with_top_db_employee_1.getEmployee_fname());
insert_DB_1.getInsert_new_employee().setRate(java.lang.Float.parseFloat(employeedb_with_top_db_employee_1.getRate()));
insert_DB_1.getInsert_new_employee().setUpdate_date(java.sql.Timestamp.valueOf(employeedb_with_top_db_employee_1.getUpdate_date()));
insert_DB_1.getInsert_new_employee().execute();
insert_DB_1.commit();
FileClient_1.setText( "procedure executed" );
FileClient_1.write();
}

Manipulating the ResultSet and Update Count Returned by Stored Procedure

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

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

DB2 stored procedures do not return records as ResultSets, instead, the records are returned through output reference cursor parameters. Reference Cursor parameters are essentially ResultSets.

The resultsAvailable() method, added to the PreparedStatementAgent class, simplifies the whole process of determining whether any results, be it Update Counts or ResultSets, are available after a stored procedure has been executed. Although JDBC provides three methods (getMoreResults(), getUpdateCount(), and getResultSet()) to access the results of a stored procedure call, the information returned from these methods can be quite confusing to the inexperienced Java JDBC programmer and they also differ between vendors. You can simply call resultsAvailable() and if Boolean true is returned, you can expect either a valid Update Count when getUpdateCount() is called and/or the next ResultSet has been retrieved and made available to one of the ResultSet nodes defined for the Stored Procedure OTD, when that node’s available() method returns true.
Frequently, Update Counts information that is returned from a Stored Procedures is insignificant. You should process returned ResultSet information and avoid looping through all of the Update Counts. The following three methods control exactly what information should be returned from a stored procedure call. The 
\texttt{enableResultSetsOnly()} method, added to the PreparedStatement Agent class allows only ResultSets to be returned and thus every \texttt{resultsAvailable()} called only returns Boolean true if a ResultSet is available. Likewise, the \texttt{enableUpdateCountsOnly()} causes \texttt{resultsAvailable()} to return true only if an Update Count is available. The default case of \texttt{enableResultsetsAndUpdateCount()} method allows both ResultSets and Update Counts to be returned.

\textbf{Collaboration usability for a stored procedure ResultSet}

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

```java
while (getSPIn().getSpS_multi().resultsAvailable())
{
    if (getSPIn().getSpS_multi().getUpdateCount() > 0)
    {
        System.err.println("Updated "+ getSPIn().getSpS_multi().getUpdateCount()+" rows");
    }

    if (getSPIn().getSpS_multi().getNormRS().available())
    {
        while (getSPIn().getSpS_multi().getNormRS().next())
        {
            System.err.println("Customer Id   ="+getSPIn().getSpS_multi().getNormRS().getCustomerId());
            System.err.println("Customer Name ="+getSPIn().getSpS_multi().getNormRS().getCustomerName());
            System.err.println();
        }
        System.err.println("===");
    }
    else if (getSPIn().getSpS_multi().getDbEmployee().available())
    {
        while (getSPIn().getSpS_multi().getDbEmployee().next())
        {
            System.err.println("EMPNO    ="+getSPIn().getSpS_multi().getDbEmployee().getEMPNO());
            System.err.println("ENAME    ="+getSPIn().getSpS_multi().getDbEmployee().getENAME());
            System.err.println("JOB      ="+getSPIn().getSpS_multi().getDbEmployee().getJOB());
            System.err.println("MGR      ="+getSPIn().getSpS_multi().getDbEmployee().getMGR());
            System.err.println("HIREDATE ="+getSPIn().getSpS_multi().getDbEmployee().getHIREDATE());
            System.err.println("SAL      ="+getSPIn().getSpS_multi().getDbEmployee().getSAL());
            System.err.println("COMM     ="+getSPIn().getSpS_multi().getDbEmployee().getCOMM());
            System.err.println("DEPTNO   ="+getSPIn().getSpS_multi().getDbEmployee().getDEPTNO());
            System.err.println();
        }
        System.err.println("===");
    }
```
Note: **resultsAvailable()** and **available()** cannot be indiscriminately called because each time they move ResultSet pointers to the appropriate locations.

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

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

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

- The method **resultsAvailable()** implicitly calls **getMoreResults()** when it is called more than once. You should not call both methods in your java code. Doing so may result in skipped data from one of the ResultSets when more than one ResultSet is present.
- The methods **available()** and **getResultSet()** can not be used in conjunction with multiple ResultSets being open at the same time. Attempting to open more the one ResultSet at the same time closes the previous ResultSet. The recommended working pattern is:
  - Open one Result Set (ResultSet_1) and work with the data until you have completed your modifications and updates. Open ResultSet_2, (ResultSet_1 is now closed) and modify. When you have completed your work in ResultSet_2, open any additional ResultSets or close ResultSet_2.
- If you modify the ResultSet generated by the Execute mode of the Database Wizard, you need to assure the indexes match the stored procedure. By doing this, your ResultSet indexes are preserved.
- Generally, getMoreResults does not need to be called. It is needed if you do not want to use our enhanced methods and you want to follow the traditional JDBC calls on your own.

The DBWizard Assistant expects the column names to be in English when creating a ResultSet.

### Prepared Statement

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

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

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

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

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

Batch Operations

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

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

getPrepStatement().getPreparedStatementTest().setAge(45);
getPrepStatement().getPreparedStatementTest().setName('Harrison Ford');
getPrepStatement().getPreparedStatementTest().setDeptNo(7);
getPrepStatement().getPreparedStatementTest().addBatch();
getPrepStatement().getPreparedStatementTest().executeBatch();
Chapter 6

Implementing DB2 eWay Sample Projects

This chapter provides an introduction to the DB2 eWay components, and information on how these components are created and implemented in a Sun Java Composite Application Platform Suite Project.

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

What’s in This Chapter

- “About the DB2 eWay Sample Projects” on page 63
- “Steps Required to Run the Sample Projects” on page 65
- “Running the SQL Script” on page 66
- “Importing a Sample Project” on page 66
- “Building and Deploying the prjDB2_BPEL Sample Project” on page 67
- “Creating the prjDB2_JCD Sample Project” on page 91

6.1 About the DB2 eWay Sample Projects

The DB2 eWay DB2_eWay_Sample.zip file contains two sample Projects that provide basic instruction on using DB2 operations in the Java Collaboration Definition (JCD), or the Business Process Execution Language (BPEL) Projects for DB2 running under Windows or Unix.

- prjDB2_JCD: demonstrates how to select, insert, update, and delete data from a DB2 database using JCDs.
- prjDB2_BPEL: demonstrates how to select, insert, update, and delete data from a DB2 database using a BPEL business process.

Both the prjDB2_JCD and prjDB2_BPEL sample Projects demonstrate how to:

- Select employee records from the db_employee table using a prepared statement
- Insert employee records data into the db_employee table.
- Update an employee record in the db_employee table.
- Delete an employee record in the db_employee table.
In addition to sample Projects, the **DB2_eWay_Sample.zip** file also includes six sample input trigger files and ten sample output files (five per sample).

**Sample input files include:**
- TriggerDelete.in.
- TriggerInsert.in. (for JCE projects only)
- TriggerBpInsert.in. (for BPEL projects only)
- TriggerPsSelect.in.
- TriggerTableSelect.in.
- TriggerUpdate.in.

**Sample output JCD files include:**
- JCD_Delete_output0.dat
- JCD_Insert_output0.dat
- JCD_PsSelect_output0.dat
- JCD_TableSelect_output0.dat
- JCD_Update_output0.dat

**Sample output BPEL files include:**
- BPEL_Delete_output0.dat
- BPEL_Insert_output0.dat
- BPEL_PsSelect_output0.dat
- BPEL_TableSelect_output0.dat
- BPEL_Update_output0.dat

### 6.1.1 Operations Used in the DB2 Sample Projects

The following database operations are used in both BPEL and JCD sample Projects:
- Insert
- Update
- Delete
- Select (SelectAll as a BPEL Activity)

**Assigning Operations in JCD**

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

1. Create a Collaboration that contains a database OTD created from the DB2 database.
2. Right-click the OTD listed in your Collaboration and then select **Select Method to Call** from the shortcut menu.
Assigning Operations in BPEL

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

1. Select the desired **receive** or **write** operation under the eWay in the Enterprise Explorer.

2. Drag the operation onto the eInsight Business Process canvas.

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

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

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

### 6.1.2 About the eInsight Engine and eGate Components

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

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

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

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

### 6.2 Steps Required to Run the Sample Projects

The following steps are required to run the sample projects that are contained in the **DB2eWayDocs.sar** file.

1. Run the SQL script.
   
   This creates the tables and records required by the sample Project.

2. Import the sample Projects.

3. Build, deploy, and run the sample Projects.
You must do the following before you can run an imported sample Project:

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

Check the output.

### 6.3 Running the SQL Script

The data used for both the JCD and BPEL sample Projects are contained within a table called **db_employee**. You create this table by using a DB2 tool to run the SQL statement **DB2_sample_script.sql**, that is included in the sample Project.

Following is the SQL statement designed for the sample Projects.

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

The sample Projects provided with the DB2 eWay use input files to pass predefined data or conditions into the Collaboration or BPEL business process, which then transforms the database contents, and delivers the result set.

### 6.4 Importing a Sample Project

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

1. Extract the samples from the Enterprise Manager to a local file.
   
   Sample files are uploaded with the eWay’s documentation SAR file, and then downloaded from the Enterprise Manager’s Documentation tab. The **DB2_eWay_Sample.zip** file contains the various sample Project ZIP files.

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

2. From the Enterprise Designer’s Project Explorer pane, right-click the Repository and select **Import Project** from the shortcut menu. The **Import Manager** appears.
Browse to the directory that contains the sample Project ZIP file. Select the sample file and click **Import**.

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

---

### 6.5 Building and Deploying the prjDB2_BPEL Sample Project

The following provides step-by-step instructions for manually creating the prjDB2_BPEL sample Project.

Steps required to create the sample project include:

- **Creating a Project** on page 67
- **Creating the OTDs** on page 67
- **Creating the Business Process** on page 69
- **Creating the Connectivity Map** on page 83
- **Creating an Environment** on page 85
- **Configuring the eWays** on page 86
- **Creating the Deployment Profile** on page 88
- **Creating and Starting the Domain** on page 89
- **Building and Deploying the Project** on page 90

#### 6.5.1 Creating a Project

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

1. Start the Enterprise Designer.
2. From the Project Explorer tree, right-click the Repository and select **New Project**. A new Project (**Project1**) appears on the Project Explorer tree.
3. Click twice on **Project1** and rename the Project (for this sample, **prjDB2_BPEL**).

#### 6.5.2 Creating the OTDs

The sample Project requires three OTDs to interact with the DB2 eWay. These OTDs include:

- DB2 Database OTD
- Inbound DTD OTD
- Outbound DTD OTD
Steps required to create a DB2 Database OTD include:

1. Right-click your new Project in the Enterprise Designer’s Project Explorer, and select **New > Object Type Definition**.
   
The New Object Type Definition Wizard window appears.

2. Select the **DB2 Database OTD Wizard** from the list of OTD Wizards and click **Next**.

3. Enter the connection information for the DB2 database. Connection fields include:
   - Host name
   - Port
   - Database
   - User name
   - Password

4. Click **Next**, and select the types of database object you want to include in the sample Project. For this example, select the following:
   - Tables/Views/Aliases
   - Prepared Statements

5. Click **Add** to select tables from the DB2 database. The **Add Tables** window appears.

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

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

8. Click **Add**, the Add Prepared Statement window appears. Enter the following:
   - Prepared Statement Name: Select_ps
   - SQL Statement:
     
     ```sql
     select * from db_employee where emp_no > ? order by emp_no
     ```

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

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

10. Enter an OTD name. In this example, we use **otdDB2**.

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

Steps required to create inbound and outbound DTD OTDs include:

1. Right-click your new Project in the Enterprise Designer’s Project Explorer, and select **New > Object Type Definition**.
   
The New Object Type Definition Wizard window appears.

2. Select **DTD** from the list of OTD Wizards and click **Next**.
3 Browse to and then select a DTD file. For our example, select one of the following DTD files from the sample Project, and then click Next.
   - otdInputDTD.dtd
   - otdOutputDTD.dtd
4 The file you select appears in the Select Document Elements window. Click Next.
5 Click Finish to complete the DTD based OTD. Repeat this process again to create the second DTD file.

6.5.3 Creating the Business Process

Steps required to create the Business Process include:
- Creating the business process flow
- Configuring the modeling elements

Creating the Business Process Flow

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

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

<table>
<thead>
<tr>
<th>Business Process</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>bpInsert</td>
<td>FileClient.Receive</td>
</tr>
<tr>
<td></td>
<td>FileClient.Write</td>
</tr>
<tr>
<td></td>
<td>FileClient.Write</td>
</tr>
<tr>
<td></td>
<td>otdDB2.DB_EMPLOYEEInsert (inside a Scope)</td>
</tr>
<tr>
<td></td>
<td>otdInputDTD_DBemployees.unmarshal</td>
</tr>
<tr>
<td>bpUpdate</td>
<td>FileClient.receive</td>
</tr>
<tr>
<td></td>
<td>FileClient.write</td>
</tr>
<tr>
<td></td>
<td>otdDB2.DB_EMPLOYEEUpdate</td>
</tr>
<tr>
<td></td>
<td>FileClient.write</td>
</tr>
</tbody>
</table>
Configuring the bpInsert Modeling Elements

Business Rules, created between the Business Process Activities, allow you to configure the relationships between the input and output Attributes of the Activities using the Business Process Designer’s Business Rule Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Insert operation. See Figure 26 for an illustration of how all the modeling elements appear when connected.

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

Figure 26  bpInsert Business Process
Steps required to configure the bpInsert business process:

Configure the following business rules in the bpInsert business process.

1. Configure the business rule between `FileClient.receive` and `FileClient.write` as seen in Figure 27.

   **Figure 27** bpInsert Business Rule # 1

2. Configure the business rule between `FileClient.write` Activity and `otdInputDTD_DBemployees.unmarshal` Activity as seen in Figure 28.

   **Figure 28** bpInsert Business Rule # 2

3. Configure the business rule between `otdInputDTD_DBemployees.unmarshal` and the `Insert` (Scope element).
4 Configure the business rule in the While statement that connects to the `otdDB2.DB_EMPLOYEEInsert` Activity.

Figure 29  bpInsert Business Rule # 3

Figure 30  bpInsert Business Rule # 4

5 Configure the business rule in the While statement that connects from the `otdDB2.DB_EMPLOYEEInsert` Activity.
Configure the business rule from the Insert (Scope element) to the FileClient.write Activity.

### Figure 32  bpInsert Business Rule # 6

**Configuring the bpUpdate Modeling Elements**

The bpUpdate business process describes how to update a record in the DB2 database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Update operation. Figure 33 illustrates how all the modeling elements appear when connected.

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

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

**Figure 33** bpUpdate Business Process

Steps required to configure the bpUpdate business process:

Configure the following business rules in the bpUpdate business process:

1. Configure the business rule between `FileClient.receive` and `FileClient.write` as seen in Figure 34.

**Figure 34** bpUpdate Business Rule # 1

2. Configure the business rule between the `FileClient.write` Activity and `otdDB2.DB_EMPLOYEEUpdate` Activity as seen in Figure 35.

**Figure 35** bpUpdate Business Rule # 2
3 Configure the business rule between `otdDB2.DB_EMPLOYEEUpdate` and the `FileClient.write` activity.

**Figure 36**  bpUpdate Business Rule # 3

**Configuring the bpDelete Modeling Elements**

The bpDelete business process describes how to delete a record in the DB2 database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the Delete operation. See Figure 37 for an illustration of how all the modeling elements appear when connected.

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

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

**Figure 37**  bpDelete Business Process

Steps required to configure the bpDelete business process:

1 Configure the business rule between `FileClient.receive` and `FileCleint.write` as seen in Figure 38.
Implementing DB2 eWay Sample Projects
Building and Deploying the prjDB2_BPEL Sample Project

Chapter 6

Section 6.5

2. Configure the business rule between the `FileClient.write` Activity and `otdB2.DB_EMPLOYEEDelete` Activity as seen in Figure 39.

3. Configure the business rule between the `otdB2.DB_EMPLOYEEDelete` Activity and the `FileClient.write` Activity as seen in Figure 40.

Configuring the `bpTableSelect` Modeling Elements

The `bpTableSelect` business process is describes how to select all records the DB2 database using the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 41 for an illustration of how all the modeling elements appear when connected.
**Note:** The where clause in the business rule reads the trigger value as a placeholder for input. This permits you to modify the query to select a specific record. Also note that all records are selected from the database when the TriggerTableSelect.in file is empty.

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

**Figure 41** bpTableSelect Business Process

Steps required to configure the bpTableSelect business process:

1. Configure the business rule between `FileClient.receive` and `FileClient.write` as seen in Figure 42.

**Figure 42** bpTableSelect Business Rule # 1

2. Configure the business rule between the `FileClient.write` Activity and `otdDB2.DB_EMPLOYEESelectAll` Activity as seen in Figure 43.

**Figure 43** bpTableSelect Business Rule # 2
3 Configure the business rule between the `otdDB2.DB_EMPLOYEESelectAll` Activity and the `otdInputDTD_DBemployees.marshal` Activity as seen in Figure 44.

**Figure 44** bpSelectTable Business Rule #3

4 Configure the business rule between the `otdInputDTD_DBemployees.marshal` Activity and the `FileClient.write` Activity as seen in Figure 45.

**Figure 45** bpTableSelect Business Rule #4

5 Configure the business rule between the `FileClient.write` Activity and the `FileClient.write` Activity as seen in Figure 46.

**Figure 46** bpTableSelect Business Rule #5
Configuring the bpPsSelect Modeling Elements

The bpPsSelect business process describes how to use a Prepared Statement query to select all records in the DB2 database via the Business Process Designer.

Once you have connected the modeling elements together, begin adding the business processes necessary to facilitate the SelectAll operation. See Figure 47 for an illustration of how all the modeling elements appear when connected.

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

![bpPsSelect Business Process](image)

**Figure 47** bpPsSelect Business Process

Steps required to configure the bpPsSelect business process:

1. Configure the business rule between `FileClient.receive` and `FileClient.write` as seen in Figure 42.

   ![bpSelectTable Business Rule # 1](image)

   **Figure 48** bpSelectTable Business Rule # 1

2. Configure the business rule between `FileClient.write` and `otdB2.Select_psPSSelectAll` as seen in Figure 49.

   ![Business Rule Designer](image)
Figure 49  bpSelectTable Business Rule # 2

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

Figure 50  Activities within Case 1 Scope

4  Configure the business rule between the start of the Scope element in Case 1 and the otdInputDTD_DBemployees.marshal activity, as seen in Figure 51.
5 Configure the business rule between `otdInputDTD_DBemployees.marshal` and `FileCleint.write` in the Scope element, as seen in Figure 52.

6 Configure Case 2 of the Decision branching activity. This requires adding business rules between the `otdInputDTD_DBemployees.marshal` and the `FileClient.write` activities within the Scope element.
Configure the business rule between the start of the Scope element in **Case 2** and the `FileClient.Write` activity, as seen in Figure 54.

Figure 54  Case 2 Scope Business Rule # 1

8 Configure the business rule between the `Decision.end` Element and the `FileClient.write` Activity, as seen in Figure 55.
6.5.4 Creating the Connectivity Map

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

1. From the Project Explorer tree, right-click the new prjDB2_BPEL Project and select New > Connectivity Map from the shortcut menu.

2. The New Connectivity Map appears, and a node for the Connectivity Map is added under the Project on the Project Explorer tree labeled CMap1.

Create four additional Connectivity Maps—CMap2, CMap3, CMap4, and CMap5— and rename them as follows:

- cmDelete
- cmInsert
- cmPsSelect
- cmTableSelect
- cmUpdate

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the prjDB2_BPEL sample Project requires the following components:
Any eWay added to the Connectivity Map is associated with an External Application. To establish a connection to DB2, first select DB2 as an External Application to use in your Connectivity Map.

**To Select a DB2 External Application**

1. Click the **External Application** icon on the Connectivity Map toolbar.
2. Select the External Applications necessary to create your Project (for this sample, DB2 and File). Icons representing the selected External Applications are added to the Connectivity Map toolbar.
3. Rename the following components and then save changes to the Repository:
   - File1 to FileClientIN
   - File2 to FileClientOUT
   - DB21 to eaDB2OUT

**To Select a DB2 Business Process**

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

**Binding the eWay Components**

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

**Steps required to bind eWay components together:**

1. Open one of the Connectivity Maps and double-click a Business Process, for example the **bpDelete** Business Process in the cmDelete Connectivity Map. The **bpDelete** Binding dialog box appears.
2. From the **bpDelete** Binding dialog box, map **FileSender** (under Implemented Services) to the **FileClientIN** (File) External Application. To do this, click on **FileSender** in the **bpDelete** Binding dialog box, and drag the cursor to the **FileClientIN** External Application in the Connectivity Map. A link is now visible between **FileClientIN** and **bpDelete**.
3. From the **bpDelete** Binding dialog box, map **DB2_otdDB2** (under Invoked Services) to the **eaDB2OUT** External Application.
4. From the **bpDelete** Binding dialog box, map **FileReceiver** to the **FileClientOUT** External Application, as seen in Figure 56.
6.5.5 Creating an Environment

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

1. From the Enterprise Designer’s Enterprise Explorer, click the Environment Explorer tab.
2. Right-click the Repository and select New Environment. A new Environment is added to the Environment Explorer tree.
3. Rename the new Environment to envDB2Proj.
4. Right-click envDB2Proj and select New DB2 External System. Name the External System esDB2. Click OK. esDB2 is added to the Environment Editor.
5. Right-click envDB2Proj and select New File External System. Name the External System esFileClient. Click OK. esFileClient is added to the Environment Editor.
6. Right-click envDB2Proj and select New Logical Host. The LogicalHost1 box is added to the Environment, and LogicalHost1 is added to the Environment Editor tree.
7. Right-click LogicalHost1 and select New Sun SeeBeyond Integration Server. A new Integration Server (IntegrationSvr1) is added to the Environment Explorer tree under LogicalHost1 (see Figure 57).

Minimize the bpDelete Binding dialog box by clicking the chevrons in the upper-right corner.

Save your current changes to the Repository, and then repeat this process for each of the other Connectivity Maps.
6.5.6 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the prjDB2_BPEL sample Project use three eWays that are represented as a nodes between the External Applications and the Business Process, as seen in Figure 58.

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

Figure 58  eWays in the cmDelete Connectivity Map

Configuring the eWay Properties

Steps required to configure the eWay properties:

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

<table>
<thead>
<tr>
<th>Connectivity Map</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmDelete</td>
<td>Input file name</td>
<td>TriggerDelete.in</td>
</tr>
<tr>
<td>cmInsert</td>
<td>Input file name</td>
<td>TriggerBplInsert.in</td>
</tr>
<tr>
<td>cmPsSelect</td>
<td>Input file name</td>
<td>TriggerPsSelect.in</td>
</tr>
<tr>
<td>cmTableSelect</td>
<td>Input file name</td>
<td>TriggerTableSelect.in</td>
</tr>
<tr>
<td>cmUpdate</td>
<td>Input file name</td>
<td>TriggerUpdate.in</td>
</tr>
</tbody>
</table>

2  Double-click the FileClientOUT eWay on each of the Connectivity Maps and modify the properties for your system, as seen in Table 16. Click OK to close the Properties Editor.

Table 16  FileClientOUT eWay Property Settings

<table>
<thead>
<tr>
<th>Connectivity Map</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmDelete</td>
<td>Input file name</td>
<td>BPEL_Delete_output%d.dat</td>
</tr>
<tr>
<td>cmInsert</td>
<td>Input file name</td>
<td>BPEL_Insert_output%d.dat</td>
</tr>
<tr>
<td>cmPsSelect</td>
<td>Input file name</td>
<td>BPEL_PsSelect_output%d.dat</td>
</tr>
<tr>
<td>cmTableSelect</td>
<td>Input file name</td>
<td>BPEL_TableSelect_output%d.dat.in</td>
</tr>
<tr>
<td>cmUpdate</td>
<td>Input file name</td>
<td>BPEL_Update_output%d.dat</td>
</tr>
</tbody>
</table>

Steps to Configure the Environment Explorer Properties

1  From the Environment Explorer tree, right-click the DB2 External System (esDB2 in this sample), and select Properties. The Properties Editor opens to the DB2 eWay Environment configuration.

2  Modify the DB2 eWay Environment configuration properties for your system, as seen in Table 17, and click OK.

Table 17  DB2 eWay Environment Properties

<table>
<thead>
<tr>
<th>Section</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration &gt; Outbound DB2 eWay for Windows/Unix &gt; JDBC Connector settings</td>
<td>ServerName</td>
<td>Enter the host name of the database server being used.</td>
</tr>
<tr>
<td></td>
<td>DatabaseName</td>
<td>Enter the name of the particular database that is being used on the server.</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>Enter the user account name for the database.</td>
</tr>
<tr>
<td></td>
<td>Password</td>
<td>Enter the user account password for the database.</td>
</tr>
</tbody>
</table>
3 From the **Environment Explorer** tree, right-click the File External System (**esFileClient** in this sample), and select **Properties**. The Properties Editor opens to the DB2 eWay Environment configuration.

4 Modify the File eWay Environment configuration properties for your system, as seen in Table 18, and click **OK**.

**Table 18**  File eWay Environment Properties

<table>
<thead>
<tr>
<th>Section</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration &gt; Inbound File eWay &gt;</td>
<td>Directory</td>
<td>Enter the directory that contains the input files (trigger files included in the sample Project). Trigger files include:</td>
</tr>
<tr>
<td>Parameter Settings</td>
<td></td>
<td>▪ TriggerBpInsert.in.~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TriggerDelete.in.~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TriggerPsSelect.in.~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TriggerTableSelect.in.~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ TriggerUpdate.in.~in</td>
</tr>
</tbody>
</table>

| Configuration > Outbound File eWay >        | Directory     | Enter the directory where output files are written. In this sample Project, the output files include:                                        |
| Parameter Settings                           |               | ▪ BPEL_Delete_output0.dat                                                                  |
|                                              |               | ▪ BPEL_Insert_output0.dat                                                                  |
|                                              |               | ▪ BPEL_PsSelect_output0.dat                                                                |
|                                              |               | ▪ BPEL_TableSelect_output0.dat                                                             |
|                                              |               | ▪ BPEL_Update_output0.dat                                                                  |

### 6.5.7 Creating the Deployment Profile

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

1. From the Enterprise Explorer’s Project Explorer, right-click the **prjDB2_BPEL** Project and select **New > Deployment Profile**.

2. Enter a name for the Deployment Profile (for this sample **dpDB2_BPEL**). Select **envDB2Proj** as the Environment and click **OK**.

3. From the Deployment Editor toolbar, click the **Automap** icon. The Project’s components are automatically mapped to their system windows, as seen in Figure 59.
6.5.8 Creating and Starting the Domain

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

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

**Steps required to create and start the domain:**

1. Navigate to your `<caps51>`/logicalhost directory (where `<caps51>` is the location of your Sun Java Composite Application Platform Suite installation.
2. Double-click the domainmgr.bat file. The Domain Manager appears.
3. If you have already created a domain, select your domain in the Domain Manager and click the Start an Existing Domain button. Once your domain is started, a green check mark indicates that the domain is running.
4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click Yes. The Create Domain dialog box appears.
5. Make any necessary changes to the Create Domain dialog box and click Create. The new domain is added to the Domain Manager. Select the domain and click the Start an Existing Domain button. Once your domain is started, a green check mark indicates that the domain is running.

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

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

Build the Project

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

Deploy the Project

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

6.5.10 Running the Sample Project

Perform the following steps to run your deployed sample Project:

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

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

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

   You can modify the following input files to view different output.
   - TriggerTableSelect.in
   - TriggerDelete.in
   - TriggerUpdate.in

   Having no content in these files causes the operation to read all records.

2. Verify the output data by viewing the sample output files. See About the DB2 eWay Sample Projects on page 63 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.
6.6 Creating the prjDB2_JCD Sample Project

The following provides step-by-step instructions for manually creating the prjDB2_JCD sample Project.

Steps required to create the sample project include:

- Creating a Project on page 91
- Creating the OTDs on page 91
- Creating a Connectivity Map on page 93
- Creating the Collaboration Definitions (Java) on page 94
- Binding the eWay Components on page 103
- Creating an Environment on page 104
- Configuring the eWays on page 105
- Creating and Starting the Domain on page 108
- Building and Deploying the Project on page 109
- Running the Sample on page 109

6.6.1 Creating a Project

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

1. Start the Enterprise Designer.
2. From the Project Explorer tree, right-click the Repository and select New Project. A new Project (Project1) appears on the Project Explorer tree.
3. Click twice on Project1 and rename the Project (for this sample, prjDB2_JCD).

6.6.2 Creating the OTDs

The sample Project requires three OTDs to interact with the DB2 eWay. These OTDs include:

- DB2 Database OTD
- Inbound DTD OTD
- Outbound DTD OTD

Steps required to create a DB2 Database OTD:

1. Right-click your new Project in the Enterprise Designer’s Project Explorer, and select New > Object Type Definition.
   The New Object Type Definition Wizard window appears.
2. Select the DB2 Database OTD Wizard from the list of OTD Wizards and click Next.
3. Enter the connection information for the DB2 database. Connection fields include:
Click **Next**, and select the types of database object you want to include in the sample Project. For our example, select the following:

- Tables/Views/Aliases
- Prepared Statements

Click **Add** to select tables from the DB2 database. The **Add Tables** window appears.

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

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

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

- Prepared Statement Name: Select_ps
- SQL Statement:

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

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

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

Enter an OTD name. In this example, use **otdDB2**.

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

**Steps required to create inbound and outbound DTD OTDs include:**

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

   The New Object Type Definition Wizard window appears.

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

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

   - otdInputDTD.dtd
   - otdOutputDTD.dtd

4 The file you select appears in the Select Document Elements window. Click **Next**.
5 Click **Finish** to complete the DTD based OTD. Repeat this process again to create the second DTD file.

### 6.6.3 Creating a Connectivity Map

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

**Steps required to create a new Connectivity Map:**

1. From the Project Explorer tree, right-click the new `prjDB2_JCD` Project and select **New > Connectivity Map** from the shortcut menu.

2. The New Connectivity Map appears and a node for the Connectivity Map is added under the Project, on the Project Explorer tree labeled `CMap1`.

Create four additional Connectivity Maps—`CMap2`, `CMap3`, `CMap4`, and `CMap5`—and rename them as follows:

- `cmDelete`
- `cmInsert`
- `cmPsSelect`
- `cmTableSelect`
- `cmUpdate`

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas.

### Populating the Connectivity Map

Add the Project components to the Connectivity Map by dragging the icons from the toolbar to the canvas.

Each Connectivity Map in the `prjDB2_JCD` sample Project requires the following components:

- File External Application (2)
- DB2 External Application
- Service

Any eWay added to the Connectivity Map is associated with an External Application. To establish a connection to DB2, first select DB2 as an External Application to use in your Connectivity Map.

**Steps required to select a DB2 External Application:**

1. Click the **External Application** icon on the Connectivity Map toolbar.

2. Select the External Applications necessary to create your Project (for this sample, DB2 and File). Icons representing the selected External Applications are added to the Connectivity Map toolbar.

3. Rename the following components and then save changes to the Repository:
4 Rename each Connectivity Map Service to match the intended operation, as for example:
   - jcdDelete
   - jcdInsert
   - jcdPsSelect
   - jcdTableSelect
   - jcdUpdate

6.6.4 Creating the Collaboration Definitions (Java)

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

JCDs required for the prjDB2_JCD sample include:
   - jcdDelete
   - jcdInsert
   - jcdPsSelect
   - jcdTableSelect
   - jcdUpdate

jcdDelete Collaboration

Steps required to create the jcdDelete Collaboration:

1 From the Project Explorer, right-click the sample Project and select New > Collaboration Definition (Java) from the shortcut menu. The Collaboration Definition Wizard (Java) appears.

2 Enter a Collaboration Definition name (for this sample jcdDelete) and click Next.

3 For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click Sun SeeBeyond > eWays > File > FileClient > receive. The File Name field now displays receive. Click Next.

4 For Step 3 of the wizard, from the Select OTDs selection window, double-click prjDB2_JCD > otdALL > otdDB2. The otdDB2 OTD is added to the Selected OTDs field.
Click the Up One Level button to return to the Repository. Double-click Sun SeeBeyond > eWays > File > FileClient. The Selected OTDs field now lists the FileClient OTD.

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

**jcdInsert Collaboration**

Steps required to create the jcdInsert Collaboration:

1. From the Project Explorer, right-click the sample Project and select New > Collaboration Definition (Java) from the shortcut menu. The Collaboration Definition Wizard (Java) appears.

2. Enter a Collaboration Definition name (for this sample jcdInsert) and click Next.

3. For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click Sun SeeBeyond > eWays > File > FileClient > receive. The File Name field now displays receive. Click Next.

4. For Step 3 of the wizard, from the Select OTDs selection window, double-click prjDB2_JCD > otdALL > otdDB2. The otdDB2 OTD is added to the Selected OTDs field.

5. In the same window, double-click otdInputDTD_DBemployees. The otdInputDTD_DBemployees OTD is added to the Selected OTDs field.

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

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

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

**jcdPsSelect Collaboration**

Steps required to create the jcdPsSelect Collaboration:

1. From the Project Explorer, right-click the sample Project and select New > Collaboration Definition (Java) from the shortcut menu. The Collaboration Definition Wizard (Java) appears.

2. Enter a Collaboration Definition name (for this sample jcdPsSelect) and click Next.

3. For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click Sun SeeBeyond > eWays > File > FileClient > receive. The File Name field now displays receive. Click Next.

4. For Step 3 of the wizard, from the Select OTDs selection window, double-click prjDB2_JCD > otdALL > otdDB2. The otdDB2 OTD is added to the Selected OTDs field.
In the same window, double-click `otdOutputDTD_DBemployee`. The `otdOutputDTD_DBemployee` OTD is added to the Selected OTDs field.

Note that the `otdOutputDTD_DBemployee` OTD is created from the `otdOutputDTD.dtd` that is included in the Sample Project.

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

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

### jcdTableSelect Collaboration

**Steps required to create the jcdTableSelect Collaboration:**

1. From the Project Explorer, right-click the sample Project and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

2. Enter a Collaboration Definition name (for this sample **jcdTableSelect**) and click **Next**.

3. For Step 2 or the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond > eWays > File > FileClient > receive**. The **File Name** field now displays **receive**. Click **Next**.

4. For Step 3 of the wizard, from the Select OTDs selection window, double-click **prjDB2_JCD > otdALL > otdDB2**. The **otdDB2** OTD is added to the **Selected OTDs** field.

5. In the same window, double-click `otdOutputDTD_DBemployee`. The `otdOutputDTD_DBemployee` OTD is added to the **Selected OTDs** field.

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

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

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

### jcdUpdate Collaboration

**Steps required to create the jcdUpdate Collaboration:**

1. From the Project Explorer, right-click the sample Project and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.

2. Enter a Collaboration Definition name (for this sample **jcdUpdate**) and click **Next**.

Sun SeeBeyond eWay™ Adapter for DB2 User’s Guide

Sun Microsystems, Inc.
For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click Sun SeeBeyond > eWays > File > FileClient > receive. The File Name field now displays receive. Click Next.

For Step 3 of the wizard, from the Select OTDs selection window, double-click prjDB2_JCD > otdALL > otdDB2. The otdDB2 OTD is added to the Selected OTDs field.

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

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

### Create the Collaboration Business Rules

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

#### Creating the jcdDelete Business Rules

The jcdDelete Collaboration implements the Input Web Service Operation to read the TriggerDelete.in file and then delete the record \( \text{emp\_no} = 500 \). The Collaboration also writes a message to JCD_Delete_output0.dat to confirm a deleted record.

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

The jcdDelete Collaboration contains the Business Rules displayed in Figure 60.

**Figure 60** jcdDelete Business Rules

Creating the jcdInsert Business Rules

The jcdInsert Collaboration implements the Input Web Service Operation to read the TriggerInsert.in file. It then unmarshals data from the input data into the otdInputDTD_DBEmployees OTD, calls the otdDB2 OTD, and inserts records into the
database via a For Loop. The Collaboration also writes a message to
JCD_Insert_output0.dat to confirm an inserted record.

The jcdInsert Collaboration contains the Business Rules displayed in Figure 61.

Figure 61 jcdInsert Business Rules

Sample code from the jcdInsert Includes:

```java
package prjDB2_JCDjcdALL;
public class jcdInsert {
    public com.stc.codegen.logger.Logger logger;
    public com.stc.codegen.alerter.Alerter alerter;
    public com.stc.codegen.util.CollaborationContext collabContext;
    public com.stc.codegen.util.TypeConverter typeConverter;

    public void receive(
        com.stc.connector.appconn.file.FileTextMessage input,
        otdDB2.OtdDB2OTD otdDB2_1, dtd.otdInputDTD_1206505729.DB_Employee otdInputDTD_DB_Employee_1,
        com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        \
        \ Writes out a message stating records are being inserted.
        FileClient_1.setText( "Inserting records in to db_employee table...." );
        FileClient_1.write();

        \
        Unmarshals data from the input XML data into the otdInputDTD_DBEmployee OTD.
        otdInputDTD_DB_Employee_1.unmarshalFromString(input.getText());

        \
        Calls the otdDB2 OTD, and inserts multiple records into the database via a For Loop. The first insert() method opens the table
```
result set for insert operations, while the insertRow() method inserts records into the table result set.

```java
otdDB2_1.getDb_employee().insert();
for (int i1 = 0; i1 <
  otdInputDTD_DB_Employee_1.countX_sequence_A(); i1 += 1) {
  otdDB2_1.getDb_employee().setEMP_NO(
    typeConverter.stringToShort(
      otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getEmpNo(), "#",
      false, 0 ) );
  otdDB2_1.getDb_employee().setLAST_NAME(
    otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastname() );
  otdDB2_1.getDb_employee().setFIRST_NAME(
    otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getFirstname() );
  otdDB2_1.getDb_employee().setRATE( new java.math.BigDecimal( otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getRate() ) );
  otdDB2_1.getDb_employee().setLAST_UPDATE(
    typeConverter.stringToTimestamp(
      otdInputDTD_DB_Employee_1.getX_sequence_A( i1 ).getLastDate(), "yyyy-MM-dd hh:mm:ss", false, "" ) );
  otdDB2_1.getDb_employee().insertRow();
}
```

```
\\ Writes a message to confirm an inserted records.
}
FileClient_1.setText( "Insert Done." );
FileClient_1.write();
```

Creating the jcdPsSelect Business Rules

The jcdPsSelect Collaboration implements the Input Web Service Operation to read the TriggerPsSelect.in file. It then copies the database resultset (as noted in the prepared statement query) into the otdOutputDTD_DBEmployee OTD and selects all available records from the database. The Collaboration also writes a message to JCD_PsSelect_output0.dat to confirm when records are selected, or when no records are available.

The jcdPsSelect Collaboration contains the Business Rules displayed in Figure 62.
Sample code from the jcdPsSelect Includes:

```java
package prjDB2_JCDjcdALL;

public class jcdPsSelect {

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

    public void receive(
        com.stc.connector.appconn.file.FileTextMessage input,
        otdDB2.OtdDB2OTD otdDB2_1,
        dtd.otdOutputDTD1325973702.DB_Employee otdOutputDTD_DB_Employee_1,
        com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {

        \<< Writes out a message stating records are being selected \>
        FileClient_1.setText( "Selecting records from db_employee table via Prepared Statement select...." );

        \<< Copies the database resultset into the otdOutputDTD_DB_Employee OTD and selects all available records from the database. The executeQuery() method executes the prepared statement query, while the resultsAvailable() method ensures all rows are retrieved in the while loop. \>>
        FileClient_1.write();
        otdDB2_1.getSelect_ps().setEMPNO(
            typeConverter.stringToShort( "0", "#", false, 0 ) );
        otdDB2_1.getSelect_ps().executeQuery();
        if ( otdDB2_1.getSelect_ps().resultsAvailable() ) {
            while ( otdDB2_1.getSelect_ps().get$Select_psResults().next() ) {
                otdOutputDTD_DB_Employee_1.setEmpNo( 
                    typeConverter.intToString( 
```
Creating the `jcdTableSelect` Business Rules

The `jcdTableSelect` Collaboration implements the Input Web Service Operation to read the `TriggerTableSelect.in` file. It then copies the database resultset into the `otdOutputDTD_DBEmployee` OTD and selects all available records from the database that meet the criteria `emp_no = 100`. The Collaboration also writes a message to `JCD_TableSelect_output0.dat` to confirm when records are selected, or when no records are available.

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

The `jcdTableSelect` Collaboration contains the Business Rules displayed in Figure 63.
Sample code from the jcdTableSelect Includes:

```java
package prjDB2_JCDjcdALL;
public class jcdTableSelect {
    public void receive( com.stc.connector.appconn.file.FileTextMessage input, dtd.otdOutputDTD1325973702.DB_Employee
        otdOutputDTD_DB_Employee_1, otdDB2.OtdDB2OTD otdDB2_1,
        com.stc.connector.appconn.file.FileApplication FileClient_1 )
        throws Throwable
    {
        FileClient_1.setText( "Selecting records from db_employee table via Table Select........" );
        FileClient_1.write();

        otddDB2_1.getDb_employee().select( input.getText() );
        while (otddDB2_1.getDb_employee().next()) {
            otdOutputDTD_DB_Employee_1.setEmpNo( typeConverter.shortToString( otddDB2_1.getDb_employee().getEMP_NO(), "#", false, "" ) );
            otdOutputDTD_DB_Employee_1.setLastname( otddDB2_1.getDb_employee().getLAST_NAME() );
            otdOutputDTD_DB_Employee_1.setFirstname( otddDB2_1.getDb_employee().getFIRST_NAME() );
            otdOutputDTD_DB_Employee_1.setRate( otddDB2_1.getDb_employee().getRATE().toString() );
            otdOutputDTD_DB_Employee_1.setLastDate( typeConverter.dateToString( otddDB2_1.getDb_employee().getLAST_DATE() ) );
        }
    }
}
```
Creating the jcdUpdate Business Rules

The jcdUpdate Collaboration implements the Input Web Service Operation to read the TriggerUpdate.in file and then update the record \texttt{emp_no = 300}. The Collaboration also writes a message to JCD_Update_output0.dat to confirm an updated record.

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

The jcdUpdate Collaboration contains the Business Rules displayed in Figure 64.

\textbf{Figure 64} jcdUpdate

6.6.6 Binding the eWay Components

The final step in creating a Connectivity Map is binding the eWay components together.
Steps required to bind eWay components together:

1. Double-click a Connectivity Map—in this example `cmDelete`—in the Project Explorer tree. The `cmDelete` Connectivity Map appears in the Enterprise Designers canvas.

2. Drag and drop the `jcdDelete` Collaboration from the Project Explorer to the `jcdDelete` Service. The Service icon “gears” change from red to green.

3. Double-click the `jcdDelete` Service. The `jcdDelete` Binding dialog box appears.

4. Map the input `FileClient` (under Implemented Services) to the `FileClientIN` (File) External Application. To do this, click on `FileSender` in the `jcdDelete` Binding dialog box, and drag the cursor to the `FileClientIN` External Application in the Connectivity Map. A link is now visible between `FileClientIN` and `jcdDelete`.

5. From the `jcdDelete` Binding dialog box, map `otdDB2` (under Invoked Services) to the `eaDB2OUT` External Application.

6. From the `jcdDelete` Binding dialog box, map `FileClient_1` to the `FileClientOUT` External Application, as seen in Figure 56.

7. Minimize the `jcdDelete` Binding dialog box by clicking the chevrons in the upper-right corner.

8. Save your current changes to the Repository, and then repeat this process for each of the other Connectivity Maps.

**Figure 65** Connectivity Map - Associating (Binding) the Project’s Components

---

### 6.6.7 Creating an Environment

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

Steps required to create an Environment:

1. From the Enterprise Designer’s Enterprise Explorer, click the **Environment Explorer** tab.
2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

3 Rename the new Environment to **envDB2Proj**.

4 Right-click **envDB2Proj** and select **New DB2 External System**. Name the External System **esDB2**. Click OK. **esDB2** is added to the Environment Editor.

5 Right-click **envDB2Proj** and select **New File External System**. Name the External System **esFileClient**. Click OK. **esFileClient** is added to the Environment Editor.

6 Right-click **envDB2Proj** and select **New Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.

7 Right-click **LogicalHost1** and select **New Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see Figure 57).

8 Save your current changes to the Repository.

### 6.6.8 Configuring the eWays

eWays facilitate communication and movement of data between the external applications and the eGate system. Each Connectivity Map in the **prjDB2_BPEL** sample Project uses three eWays that are represented as nodes between the External Applications and the Business Process, as seen in Figure 58.

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

Steps required to configure the eWay properties:

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

Table 19  FileClientIN eWay Property Settings

<table>
<thead>
<tr>
<th>Connectivity Map</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmDelete</td>
<td>Input file name</td>
<td>TriggerDelete.in</td>
</tr>
<tr>
<td>cmInsert</td>
<td>Input file name</td>
<td>TriggerInsert.in</td>
</tr>
<tr>
<td>cmPsSelect</td>
<td>Input file name</td>
<td>TriggerPsSelect.in</td>
</tr>
<tr>
<td>cmTableSelect</td>
<td>Input file name</td>
<td>TriggerTableSelect.in</td>
</tr>
<tr>
<td>cmUpdate</td>
<td>Input file name</td>
<td>TriggerUpdate.in</td>
</tr>
</tbody>
</table>

2. Double-click the FileClientOUT eWay on each of the Connectivity Maps and modify the properties for your system, as seen in Table 20. Click OK to close the Properties Editor.

Table 20  FileClientOUT eWay Property Settings

<table>
<thead>
<tr>
<th>Connectivity Map</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmDelete</td>
<td>Input file name</td>
<td>JCD_Delete_output%d.dat</td>
</tr>
<tr>
<td>cmInsert</td>
<td>Input file name</td>
<td>JCD_Insert_output%d.dat</td>
</tr>
<tr>
<td>cmPsSelect</td>
<td>Input file name</td>
<td>JCD_PsSelect_output%d.dat</td>
</tr>
<tr>
<td>cmTableSelect</td>
<td>Input file name</td>
<td>JCD_TableSelect_output%d.dat.in</td>
</tr>
<tr>
<td>cmUpdate</td>
<td>Input file name</td>
<td>JCD_Update_output%d.dat</td>
</tr>
</tbody>
</table>

Figure 67  eWays in the cmDelete Connectivity Map
Configuring the Environment Explorer Properties

Steps required to configure the Environment Explorer properties:

1. From the Environment Explorer tree, right-click the File External System (esDB2 in this sample), and select Properties. The Properties Editor opens to the DB2 eWay Environment configuration.

2. Modify the DB2 eWay Environment configuration properties for your system, as seen in Table 21, and click OK.

Table 21  DB2 eWay Environment Properties

<table>
<thead>
<tr>
<th>Section</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration &gt; Outbound DB2 eWay for Windows/Unix &gt; JDBC Connector settings</td>
<td>ServerName</td>
<td>Enter the name of the database server being used.</td>
</tr>
<tr>
<td></td>
<td>DatabaseName</td>
<td>Enter the name of the particular database that is being used on the server.</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>Enter the user account name for the database.</td>
</tr>
<tr>
<td></td>
<td>Password</td>
<td>Enter the user account password for the database.</td>
</tr>
</tbody>
</table>

3. From the Environment Explorer tree, right-click the File External System (esFileClient in this sample), and select Properties. The Properties Editor opens to the DB2 eWay Environment configuration.

4. Modify the File eWay Environment configuration properties for your system, as seen in Table 22, and click OK.

Table 22  File eWay Environment Properties

<table>
<thead>
<tr>
<th>Section</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration &gt; Inbound File eWay &gt; Parameter Settings</td>
<td>Directory</td>
<td>Enter the directory that contains the input files (trigger files included in the sample Project).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Trigger files include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TriggerDelete.in~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TriggerInsert.in~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TriggerPsSelect.in~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TriggerTableSelect.in~in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TriggerUpdate.in~in</td>
</tr>
</tbody>
</table>
6.6.9 Configuring the Integration Server

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

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

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

3. Click the ellipsis. The Password Settings dialog box appears. Enter STC as the Specific Value and as the Confirm Password, and click OK.

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

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

6.6.10 Creating the Deployment Profile

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

Steps required to create the Deployment Profile:

1. From the Enterprise Explorer’s Project Explorer, right-click the prjDB2_JCD Project and select New > Deployment Profile.

2. Enter a name for the Deployment Profile (for this sample dpDB2_JCD). Select envDB2Proj as the Environment and click OK.

3. From the Deployment Editor toolbar, click the Automap icon. The Project’s components are automatically mapped to their system windows, as seen in Figure 59.

<table>
<thead>
<tr>
<th>Section</th>
<th>Property Name</th>
<th>Required Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration &gt;</td>
<td>Directory</td>
<td>Enter the directory where output files are written. In this sample Project, the output files include:</td>
</tr>
</tbody>
</table>
| Outbound File eWay |                    |   - JCD_Delete_output0.dat  
| > Parameter Settings |                    |   - JCD_Insert_output0.dat  
|                  |                    |   - JCD_PsSelect_output0.dat  
|                  |                    |   - JCD_TableSelect_output0.dat  
|                  |                    |   - JCD_Update_output0.dat  |
6.6.11 Creating and Starting the Domain

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

Steps required to create and start the domain:

1. Navigate to your `<caps51>` directory (where `<caps51>` is the location of your Sun Java Composite Application Platform Suite installation).

2. Double-click the `domainmgr.bat` file. The Domain Manager appears.

3. If you have already created a domain, select your domain in the Domain Manager and click the Start an Existing Domain button. Once your domain is started, a green check mark indicates that the domain is running.

4. If there are no existing domains, a dialog box indicates that you can create a domain now. Click Yes. The Create Domain dialog box appears.

5. Make any necessary changes to the Create Domain dialog box and click Create. The new domain is added to the Domain Manager. Select the domain and click the Start an Existing Domain button. Once your domain is started, a green check mark indicates that the domain is running.

For more information about creating and managing domains see the eGate Integrator System Administration Guide.
6.6.12 Building and Deploying the Project

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

**Build the Project**

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

**Deploy the Project**

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

6.6.13 Running the Sample

Additional steps are required to run the deployed sample Project.

**Steps required to run the sample Project:**

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

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

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

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

- TriggerTableSelect.in
- TriggerDelete.in
- TriggerUpdate.in

Having no content in these files causes the operation to read all records.

Verify the output data by viewing the sample output files. See About the DB2 eWay Sample Projects on page 63 for more details on the types of output files used in this sample Project. The output files may change depending on the number of times you execute the sample Project, the input file, and also the content of your database table.
Appendix A

Common Data Type Conversions

This section lists conversions between the DB2 and OTD/Java datatypes, the types of methods to use for each datatype, and the value or size of the data element that can be used.

What’s in this Appendix:

- Common Data Type Conversions on page 111

A.1 Common Data Type Conversions

Table 1 lists common datatype conversions used for the DB2.

<table>
<thead>
<tr>
<th>DB2 Data Type</th>
<th>OTD/Java Data Type</th>
<th>JCD Class Browser Fields</th>
<th>Sample Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>BigInt</td>
<td>Long</td>
<td>Long:</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.lang.Long.parseLong(String)</td>
<td></td>
</tr>
<tr>
<td>Int</td>
<td>Int</td>
<td>Integer:</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.lang.Integer.parseInt(String)</td>
<td></td>
</tr>
<tr>
<td>tinyInt</td>
<td>Byte</td>
<td>Byte:</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.lang.Byte.parseByte(String)</td>
<td></td>
</tr>
<tr>
<td>SmallInt</td>
<td>Short</td>
<td>Short:</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.lang.Short.parseShort(String)</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>BigDecimal</td>
<td>BigDecimal:</td>
<td>145.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.math.BigDecimal(String)</td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>BigDecimal</td>
<td>BigDecimal:</td>
<td>145.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.math.BigDecimal(String)</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>Boolean</td>
<td>Boolean:</td>
<td>true or false</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.lang.Boolean.getBoolean(String)</td>
<td></td>
</tr>
<tr>
<td>Real</td>
<td>Float</td>
<td>Float:</td>
<td>3468.494</td>
</tr>
<tr>
<td></td>
<td></td>
<td>java.lang.Float.parseFloat(String)</td>
<td></td>
</tr>
<tr>
<td>DB2 Data Type</td>
<td>OTD/Java Data Type</td>
<td>JCD Class Browser Fields</td>
<td>Sample Data</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------</td>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
<td>java.lang.Double.parseDouble(String)</td>
<td>3468.494</td>
</tr>
<tr>
<td>Money</td>
<td>BigDecimal</td>
<td>Call a NewConstructor BigDecimal: java.math.BigDecimal(String)</td>
<td>2456.95</td>
</tr>
<tr>
<td>Smallmoney</td>
<td>BigDecimal</td>
<td>Call a NewConstructor BigDecimal: java.math.BigDecimal(String)</td>
<td>2456.95</td>
</tr>
<tr>
<td>Smalldatetime</td>
<td>TimeStamp</td>
<td>java.sql.TimeStamp.valueOf(String)</td>
<td>2003-09-28 11:35:00</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Binary</td>
<td>N/A (Used by the Database Internally)</td>
<td>N/A</td>
</tr>
<tr>
<td>DateTime</td>
<td>TimeStamp</td>
<td>java.sql.TimeStamp.valueOf(String)</td>
<td>2003-09-28 11:35:42</td>
</tr>
<tr>
<td>Varchar</td>
<td>String</td>
<td>Direct Assign</td>
<td>Any Characters</td>
</tr>
<tr>
<td>Char</td>
<td>String</td>
<td>Direct Assign</td>
<td>Any Characters</td>
</tr>
<tr>
<td>Text</td>
<td>String</td>
<td>Direct Assign</td>
<td>Any Characters</td>
</tr>
<tr>
<td>Binary(1)</td>
<td>Byte[]</td>
<td>java.lang.String.getBytes()</td>
<td>0 or 1</td>
</tr>
</tbody>
</table>
Index

A
Activity Input and Output52
Add Prepared Statements46
Automap88, 108

B
binding
dialog box85, 104
BPEL operations52
Building an eWay Project63

C
Collaboration
editor94
Connection Retry Support8
Connectivity Map
   Outbound properties22
Connectivity Map Generator8
conventions, text9

D
Database Operations
   BPEL52
   JCD54
database OTD wizard
   steps to create37
Delete Operation57
Deployment Profile
   Automap88, 108

E
Editable OTD Support8
eWays
   creating18
Executing Stored Procedures58
External Application
   creating17

I
Inbound Properties
   LocationName26, 30
Insert Operation55
installation11
Installing
   Repository on UNIX11
   sample Projects and Javadocs13

J
Javadocs, installing13
JCD operations54

L
LDAP Configuration8
LocationName26, 30

M
Multiple Drag-and-Drop8

O
Operations
   BPEL52
   Delete57
   Insert55
   JCD54
   Query (Select)54
   Update56
Outbound Connectivity Map Properties22

P
prepared statement
batch operations62
executing61
Project
importing66
properties
   Connectivity Map properties
      modifying18
   Environment properties
      modifying19
   MaxPoolSize29
   MinPoolSize29
      modifying21
   Connectivity Map properties18
Properties Editor21
Property settings, Inbound
   LocationName26, 30
Index

Q
Query (Select) Operation54

R
Relaunchable OTD Support8
ResultSet
  collaboration usability for a stored procedure60
ResultSet methods
  available59
  enableResultSetandUpdateCounts59
  enableResultSetOnly59
  enableUpdateCountsOnly59
  getUpdateCount59
  next59
  resultsAvailable59

S
sample projects, installing13
SQL54
SQL operations, table54
Stored Procedure
  manipulating ResultSet and update count59
Stored Procedures58
  Executing58

T
Table
  SQL operations54
  text conventions9

U
update count59
Update Operation56

V
Version control8