SUN SEEBEYOND

eWAY™ ADAPTER FOR WEBSPHERE MQ USER'S GUIDE

Release 5.1.3

Part Number: 820-0980

Version 20070420111012

Contents

Chapter 1

Introduction

This document describes how to install, configure, and implement the Sun SeeBeyond eWay™ Adapter for WebSphere MQ, in a typical eGate environment.

This chapter provides a brief overview of operations and components, general features, and system requirements of the Sun SeeBeyond eWay™ Adapter for WebSphere MQ.

What's in This Chapter

1.1 About IBM's WebSphere MQ

WebSphere MQ (formerly MQSeries™) from IBM™ is a client-server message broker supporting an open API (application programming interface), available on a variety of operating systems including AIX™, Solaris™, HP-UX™, and Windows™. WebSphere MQ is "middleware" that provides commercial messaging and queuing services. Messaging enables programs to communicate with each other via messages rather than direct connection. Messages are placed in queues for temporary storage, freeing up programs to continue to work independently. This process also allows communication across a network of dissimilar components, processors, operating systems, and protocols.

1.2 About the WebSphere MQ eWay Adapter

The Sun SeeBeyond eWay™ Adapter for WebSphere MQ (referred to as the WebSphere MQ eWay throughout this document) allows the eGate system to exchange data with IBM's WebSphere MQ. The eGate Integrator, using the WebSphere MQ eWay, uses business logic within a Collaboration or Business Process to perform operations for data identification, manipulation, and transformation. Messages are tailored to meet the communication requirements of specific applications or protocols. Queues or Topics provide non-volatile storage for data within the eGate system allowing applications to run independently of one another at different speeds and times.

The WebSphere MQ eWay transparently integrates existing systems with IBM's WebSphere MQ. This document explains how to install and configure the WebSphere MQ eWay.

1.3 What's New in This Release

The Sun SeeBeyond eWay™ Adapter for WebSphere MQ includes the following changes and new features:

New for Version 5.1.3

- New Functionality: The use of the SSL feature with the WebSphere MQ eWay provides a secure communications channel for data exchanges.

New for Version 5.1.2

- Supports XA Transactions for Client mode (outbound).

New for Version 5.1.1

- Supports automatic deployment of EAR files to WebLogic Application Server version 9.1.

New for Version 5.1.0

- Version Control: An enhanced version control system allows you to effectively manage changes to the eWay components.
- Manual Connection Management: Establishing a connection can now be performed automatically (configured as a property) or manually (using OTD methods from the Java Collaboration).
- 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: eWay configuration properties now support LDAP key values.
- MDB Pool Size Support: Provides greater flow control (throttling) by specifying the maximum and minimum MDB pool size.
- 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.
- Bulkget and Bulkput for BPEL: Allows you to get/put multiple messages at a time.
- MQ Open Options exposed in the Connectivity Map: Allows you to set open option outside the Java Collaboration Definition.
- Added support for WebSphere MQ version 6.0.

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

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

1.4 What's in This Document

This guide includes the following chapters:

- Chapter 1 **"Introduction"** provides an overview of the WebSphere MQ eWay.
- Chapter 2 **"Installing the WebSphere MQ eWay"** describes how to install the WebSphere MQ eWay and lists the supported operating systems and system requirements.
- Chapter 4 **"Configuring the WebSphere eWay"** describes the process of configuring the WebSphere MQ eWay to run in your environment.
- Chapter 5 **"Using the WebSphere MQ eWay With eInsight"** describes the features and functionality of the WebSphere MQ eWay using the eInsight Business Process Manager and the eInsight Web Services interface.
- Chapter 6 **"Using the WebSphere MQ eWay with Java Collaborations"** describes the features and functionality of the WebSphere MQ eWay using the eGate Integrator and the Collaboration Editor (Java).

1.4.1 WebSphere MQ eWay Javadoc

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

1.4.2 Scope of the Document

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

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

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	▪ Click **OK**. ▪ On the **File** menu, click **Exit**. ▪ Select the **eGate.sar** file.
`Monospaced`	Command line arguments, code samples; variables are shown in ***bold italic***	`java -jar` ***`filename`***`.jar`
Blue bold	Hypertext links within document	See **Text Conventions** on page 10
Blue underlined	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.5 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

1.6 Documentation Feedback

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

CAPS_docsfeedback@sun.com

Chapter 2

Installing the WebSphere MQ eWay

This chapter explains how to install the WebSphere MQ eWay and lists the supported operating systems and system requirements for installation.

What's in This Chapter

2.1 WebSphere MQ eWay System Requirements

The WebSphere MQ eWay Readme contains the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements

The WebSphere MQ eWay Readme is uploaded with the eWay's documentation file (MQSerieseWayDocs.sar) and can be accessed from the Documentation tab of the Sun Java Integrator Suite Installer. Refer to the eWay Readme for the latest requirements before installing or running the WebSphere MQ eWay.

2.2 System Requirements

The WebSphere MQ eWay has the following system requirements:

- On 64-bit platforms, the WebSphere MQ eWay requires that the installed LogicalHost and the MQ library are both either the 32-bit versions or both the 64-bit versions.
- The system requirements for the WebSphere MQ eWay are the same as those for eGate Integrator. For more information, refer to the *Sun Java Composite Application Platform Suite Installation Guide*. It is also helpful to review the **Readme** for

additional requirements prior to installation. The **Readme** is accessed from the Documentation tab of the Enterprise Manager Installer.

- For **AIX operating systems**, the environmental variable **LDR_CNTRL** for JVM may need to be adjusted in order to accommodate WebSphere MQ shared memory. Java uses 8 segments by default (this is the maximum value allowed; each segment is 256 MB). For example, the following setting changes the number of segments to 3:

  ```
  setenv LDR_CNTRL MAXDATA=0x30000000
  ```

  This line can only be run from the C-Shell.

  A **2102 error, MQRC_RESOURCE_PROBLEM: Insufficient system resources available, might indicate that your MAXDATA needs to be modified using the solution provided above.**

- For **HP-UX 11** operating systems, HP-UX Java binding support is only available for systems running the POSIX draft 10 threaded version of WebSphere MQ. The HP-UX Developers kit for Java 1.1.7, Release C.01.17.01 or above is also required.

- Although the WebSphere MQ eWay, the Repository, and Logical Hosts run on the platforms listed under **Supported Operating Systems**, the Enterprise Designer requires the Windows operating system. The Enterprise Manager can run on any platform that supports Internet Explorer 6.0.

The WebSphere MQ eWay's **Readme.txt** file contains the latest requirements and information for the WebSphere MQ eWay. The WebSphere MQ eWay Readme is uploaded with the eWay's documentation file (**MQSerieseWayDocs.sar**) and accessed from the Documentation tab of the Enterprise Manager.

2.2.1 External System Requirements

The WebSphere MQ eWay requires the following installed on the Logical Host:

- IBM WebSphere MQ V6.0 or WebSphere MQ V5.3.
- The **com.ibm.mq.jar** file specific to the operating system on which WebSphere MQ is deployed. See **"Copying the com.ibm.mq.jar File to the eWay" on page 17**.

Install the following after installing IBM MQSeries V5.2:

- IBM MQSeries classes for Java 5.2.0.
- Classes for Java Message Service 5.2.0.0

2.3 JMS Services and the WebSphere MQ eWay

The WebSphere MQ eWay does not support a JMS interface. The eWay's OTD and MQ connectivity are implemented using the WebSphere MQ base Java classes, not the WebSphere MQ classes for Java Message Service (JMS). The WebSphere MQ eWay has no intrinsic awareness or support for JMS. To use the eWay and JMS services, refer to the *Sun SeeBeyond eGate Integrator User's Guide* for information regarding JMS services provided by the Sun Java Composite Application Platform Suite for connectivity options.

2.4 Installing the WebSphere MQ eWay

The Sun Java Composite Application Platform Suite Installer, 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.

Note: *When the Repository is running on a UNIX operating system, the eWays are loaded from the Enterprise Manager running on a Windows computer connected to the Repository server using Internet Explorer.*

2.4.1 Installing the eWay on a JavaCAPS Supported System

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

1. From the Sun Java Composite Application Platform Suite Installer's **ADMINISTRATION** tab, click on **license** in the Sun Java Composite Application Platform Suite Products Installed table.
2. From **Select Sun Java Composite Application Platform Suite Products to Install**, select the products for your Sun Java Composite Application Platform Suite and include the following:
   - **FileeWay** (the File eWay is used in the sample Projects)
   - **MQSerieseWay**

   To upload the WebSphere MQ eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, select the following:
   - **MQSerieseWayDocs**
3. Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.
4. 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.
5. From the **Installing Files** window, review the product list. If it is correct, Click **Install Products**.
6. When your product's installation is completed, click on the prompt, "**When installation completes, click here to continue**."
7. Continue installing the eGate Integrator as instructed in the *Sun Java Composite Application Platform Suite Installation Guide.*

Adding the eWay to an Existing Suite Installation

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

1. Complete steps 1 through 6 above.
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**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**
7. When prompted, restart the IDE (Integrated Development Environment) to complete the installation.

2.4.2 Installing eWay Enterprise Manager plug-ins

The **Sun SeeBeyond Enterprise Manager** is a Web-based interface that allows you to monitor and manage your Java Integration Suite applications. The Enterprise Manager requires an eWay specific "plug-in" for each of your installed eWays. 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 Installation Guide* describes how to install the Sun SeeBeyond Enterprise Manager. The *Sun SeeBeyond eGate™ Integrator System Administration Guide* describes how to monitor servers, Services, logs, and alerts using the Sun SeeBeyond Enterprise Manager and the command-line client.

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

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

1. From the Enterprise Manager:
   A. From the **Enterprise Manager**'s Explorer toolbar, click the **Configuration** icon.
   B. Click the **Web Applications Manager** tab, go to the **Auto-Install from Repository** tab, and connect to your Repository.
   C. Select the application plug-ins you require, and click **Install**. The application plug-ins are installed and deployed.
2. From the **Sun Java Composite Application Platform Suite Installer:**
   A. From the **Sun Java Composite Application Platform Suite Installer's Download tab**, select the Plug-Ins you require and save them to a temporary directory.

B Log onto the **Sun SeeBeyond Enterprise Manager**. From the **Enterprise Manager**'s Explorer toolbar, click the **Configuration** icon.

C Click the **Web Applications Manager** tab and go to the **Manage Applications** tab.

D 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.

WebSphere MQ eWay 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 the **Configuration** icon.
3. Click the **Web Applications Manager** tab and go to the **Manage Alert Codes** tab. Your installed alert codes are displayed under the **Results** section. If your eWay alert codes are not available displayed under **Results**, do the following

   A From the **Install New Alert Codes** section, browse to and select the eWay alert properties file for the application plug-in that you added. The alert properties files are located in the **alertcodes** folder of your Sun Java Composite Application Platform Suite installation directory.

   B Click **Deploy**. The available alert codes for your application are displayed under **Results**. A listing of available this eWay's alert codes is displayed in Table 2.

Table 2 WebSphere MQ eWay Alert Codes

Alert Code	Description	User Action
WEBSPHEREMQEWAY_CONNECT	This is an informational alert that is posted when the eWay successfully connects to a queue manager in client connections mode.	This alert does not indicate any malfunction. No user actions are necessary.
WEBSPHEREMQEWAY_CONNECT_BINDINGS	This is an informational alert that is posted when the eWay successfully connects to a queue manager in bindings connections mode.	This alert does not indicate any malfunction. No user actions are necessary.
WEBSPHEREMQEWAY_CONNECT_FAILED_BINDINGS	This alert indicates that the eWay failed to connect to a queue manager in bindings mode.	▪ Verify that your eWay configuration has the correct connection parameters. ▪ Verify that your network connectivity is valid. ▪ Verify that your targeted WebSphere MQ application is running and configured correctly.

Alert Code	Description	User Action
WEBSPHEREMQEWAY_CONNECT_FAILED_CLIENT	This alert indicates that the eWay failed to connect to a queue manager in client mode.	▪ Verify that your eWay configuration has the correct connection parameters. ▪ Verify that your network connectivity is valid. ▪ Verify that your targeted WebSphere MQ application is running and configured correctly.
WEBSPHEREMQEWAY_DISCONNECT	This alert indicates that the eWay has successfully disconnected with the targeted queue manager for one of the following reasons: ▪ The eWay is shutting down as initiated by the user or by a Collaboration. ▪ The eWay is being shut down by another JavaCAP Suite subcomponent as the result of a Collaboration error.	This alert does not (necessarily) indicate any malfunction. Refer to the log file for more information.
WEBSPHEREMQEWAY_RECEIVE_FAILED	This alert is posted if the eWay operates in receive (inbound polling) mode, and fails to retrieve a message from a MQ queue when a message is expected, This is not a case in which the eWay fails to fetch a message from an empty queue.	▪ Verify your network connection. ▪ Verify that the eWay parameters associated with message retrieval are correct. ▪ The eWay has received a truncated message, and the eWay is configured to NOT accept truncated messages. ▪ Check your targeted MQ queue or its queue manager for errors.
WEBSPHEREMQEWAY_RECEIVE_TRUNC	This alert indicates that the eWay, operating in receive mode, has successfully retrieved a message from the queue, but the message is truncated.	▪ If the eWay is configured beforehand to accept truncated messages, this alert is a warning regarding the severity of message truncation. ▪ If the eWay is configured to NOT accept truncated messages, and a truncated message is received, this alert is NOT posted; rather, the preceding alert is the one that is posted.
WEBSPHEREMQEWAY_RECONNECT_FAILED	This alert is no longer posted by the product. This alert code is reserved for future enhancement.	This alert is no longer posted by the product.

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.*

2.4.3 WebSphere MQ eWay Required JAR Files

The WebSphere MQ eWay requires the JAR file: **com.ibm.mq.jar**, a system specific file, for normal operation. For XA transactions in Client mode, or for deploying EAR files to a WebLogic Application Server, the eWay also requires the **com.ibm.mqetclient.jar** file.

Copying the com.ibm.mq.jar File to the eWay

The WebSphere MQ eWay requires the JAR file: **com.ibm.mq.jar**. This JAR file must be copied from the MQ server you are running against. Depending on the mode of operation, the WebSphere MQ library utilized by the eWay may require operating system-specific modules.

If a Project is deployed to a different MQ server running on a different operating system, the JAR file on the Logical Host must be replaced with the JAR file from that specific MQ server (so that the JAR file remains operating system specific for the current MQ Server).

To install the correct JAR file, do the following:

1. From the WebSphere MQ Server, copy the **com.ibm.mq.jar** file.
2. Paste the **com.ibm.mq.jar** file to the following directory:

   ```
   <JavaCAPS51>\logicalhost\is\lib
   ```

   where <*JavaCAPS51*> is the directory in which Sun Java Composite Application Platform Suite is installed.

Copying the com.ibm.mqetclient.jar File to the eWay

The WebSphere MQ eWay requires the JAR file: **com.ibm.mqetclient.jar**, for XA transactions in Client mode and for deploying EAR files to a WebLogic Application Server. The **com.ibm.mqetclient.jar** file is installed as part of the WebSphere MQ Extended Transaction client installation. For directions on adding this file for XA transactions in Client mode, see **“Adding XA Required JAR Files to the Integration Server Classpath” on page 17**. For directions on adding this file for deploying EAR files to a WebLogic Application Server, see **“Deploying EAR Files to Application Servers” on page 18**.

Adding XA Required JAR Files to the Integration Server Classpath

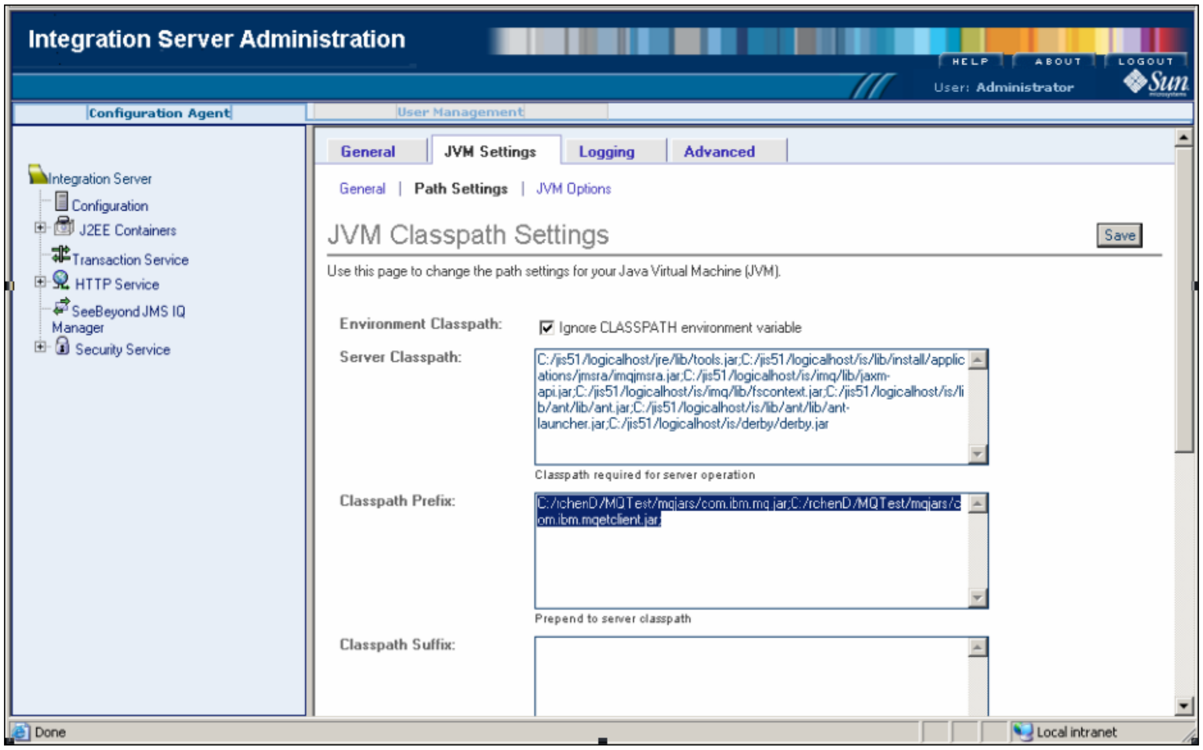
To run XA transactions in Client mode, the **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** must be added to the **Sun SeeBeyond Integration Server** classpath. To add the JAR files to the IS classpath, do the following:

1. Note the location on your Logical Host, where you placed the **com.ibm.mq.jar** and **com.ibm.mqetclient.jar.**

2. Open the Integration Server Administration tool. For directions on how to access the Integration Server Administration tool, see the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

3. From the IS Administration tool's **Classpath Prefix** field, add the absolute path for both of the required JAR files (see Figure 1).

Figure 1 Integration Server Administration Tool - Classpath Prefix



JVM Classpath Settings

Use this page to change the path settings for your Java Virtual Machine (JVM).

Environment Classpath: [✓] Ignore CLASSPATH environment variable

Server Classpath:

```
C:/jis51/logicalhost/jre/lib/tools.jar;C:/jis51/logicalhost/is/lib/install/applic
ations/jmsra/imqjmsra.jar;C:/jis51/logicalhost/is/imq/lib/jaxm-
api.jar;C:/jis51/logicalhost/is/imq/lib/fscontext.jar;C:/jis51/logicalhost/is/li
b/ant/lib/ant.jar;C:/jis51/logicalhost/is/lib/ant/lib/ant-
launcher.jar;C:/jis51/logicalhost/is/derby/derby.jar
```

Classpath required for server operation

Classpath Prefix:

```
C:/rchenD/MQTest/mqjars/com.ibm.mq.jar;C:/rchenD/MQTest/mqjars/c
om.ibm.mqetclient.jar;
```

Prepend to server classpath

Classpath Suffix:

4. Save and close the IS Administration tool.

2.4.4 Deploying EAR Files to Application Servers

The Sun Java CAPS Enterprise Designer can be configured to automatically deploy an EAR file to the Sun Java System Application Server. To configure the Enterprise Designer for deployment, follow the directions for deploying applications to the Sun Java System Application Server, provided in the *Sun SeeBeyond eGate Integrator System Administration Guide*. Because automatic deployment is not supported directly from Enterprise Designer for the Weblogic Application Server or the WebSphere Application Server, additional instructions are provided below.

WebLogic Application Servers

The WebSphere MQ eWay supports automatic deployment of EAR files to WebLogic Application Server (version 9.1). For step by step directions, see the *Sun SeeBeyond eGate Integrator System Administration Guide*. This deployment also requires that you add the system specific JAR files: **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** to the WebLogic Application Server classpath.

The system specific **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** files, copied from the MQ server you are running against, must be added to the WebLogic Application Server's classpath. You can do this by copying the **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** files to the WebLogic server in the following directory:

```
C:\bea\weblogic91\<ApplicationServer>\domains\wl_server\lib
```

where `<ApplicationServer> is the name of your Application Server.`

1. Or, you can add the location of these JAR files on the WebLogic server, to the classpath.Build the EAR file, which is generated in the Enterprise Designer.
2. Use your WebLogic Admin console to deploy the EAR file.

Refer to your application server's documentation for requirements regarding working directories.

WebSphere Application Servers

The WebSphere MQ eWay supports automatic deployment of EAR files to WebSphere Application Server (version 9.1). For step by step directions, see the *Sun SeeBeyond eGate Integrator System Administration Guide*. This deployment also requires that you add the system specific JAR files: **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** to the WebSphere Application Server classpath.

The system specific **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** files, copied from the MQ server you are running against, must be added to the WebSphere Application Server's classpath. You can do this by copying the **com.ibm.mq.jar** and **com.ibm.mqetclient.jar** files to the WebSphere server in the following directory:

```
C:ProgramFiles\IBM\WebSphere\AppServer\lib\WMQ\java\lib
```

where `<ApplicationServer> is the name of your Application Server.`

1. Build the EAR file, which is generated in the Enterprise Designer.
2. Use either the WebShpere Admin console or run the command line scripts to deploy the EAR file.

Refer to your application server's documentation for requirements regarding working directories. Refer to the *Java CAPS Installation Guide* for more information on deploying the WebSphere Application Server and other known issues.

2.4.5 After Installation

Once the eWay is installed and configured it must then be incorporated into a Project before it can perform its intended functions. See the *Sun SeeBeyond eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

2.5 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0 Projects to Sun Java Composite Application Platform Suite, version 5.1.3. Only Projects developed on ICAN version 5.0.2 and above can be migrated successfully to the Sun Java Composite Application Platform Suite. To migrate your ICAN 5.0 Projects, do the following:

Export the Project

1. Before you export your Projects, save your current ICAN 5.0 Projects to your Repository.
2. From the Project Explorer, right-click your Project and select **Export** from the shortcut menu. The Export Manager appears.
3. Select the Project that you want to export in the left pane of the Export Manager and move it to the Selected Projects field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Projects.
4. In the same manner, select the Environment that you want to export in the left pane of the Export Manager and move it to the Selected Environments field by clicking the **Add to Select Items** (arrow) button, or click **All** to include all of your Environments.
5. Browse to select a destination for your Project ZIP file and enter a name for your Project in the **ZIP file** field.
6. Click **Export** to create the Project ZIP file in the selected destination.

Install Sun Java Composite Application Platform Suite

7. Install the Sun Java Composite Application Platform Suite, including all eWays, libraries, and other components used by your ICAN 5.0 Projects.
8. Start the Sun SeeBeyond Enterprise Designer.

Import the Project

9. From the Enterprise Designer's Project Explorer tree, right-click the Repository and select **Import Project** from the shortcut menu. The Import Manager appears.
10. Browse to and select your exported Project file.
11. Click **Import**. A warning message, **"Missing APIs from Target Repository**," may appear at this time. This occurs because various product APIs were installed on the ICAN 5.0 Repository when the Project was created, that are not installed on the Sun Java Composite Application Platform Suite Repository. These APIs may or may not apply to your Projects. You can ignore this message if you have already installed all of the components that correspond to your Projects. Click **Continue** to resume the Project import.
12. Close the Import Manager after the Project is successfully imported.

Deploy the Project

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

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

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

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

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

14 If your imported Project includes File eWays, these must be reconfigured in your Environment prior to deploying the Project. To reconfigure your File eWays, do the following:

A The Environment File External System properties can now accommodate both inbound and outbound eWays. If your previous Environment includes both inbound and outbound File External Systems, delete one of these (for example, the outbound File External System).

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

C The Directory property has been relocated from the Connectivity Map Properties to the Environment Properties. Set the inbound and outbound Directory values, and click **OK**.

15 Deploy your Projects.

Note: *Only projects developed on ICAN 5.0.2 and above can be imported and migrated successfully into the Java Integration Suite.*

Chapter 3

Operating SSL

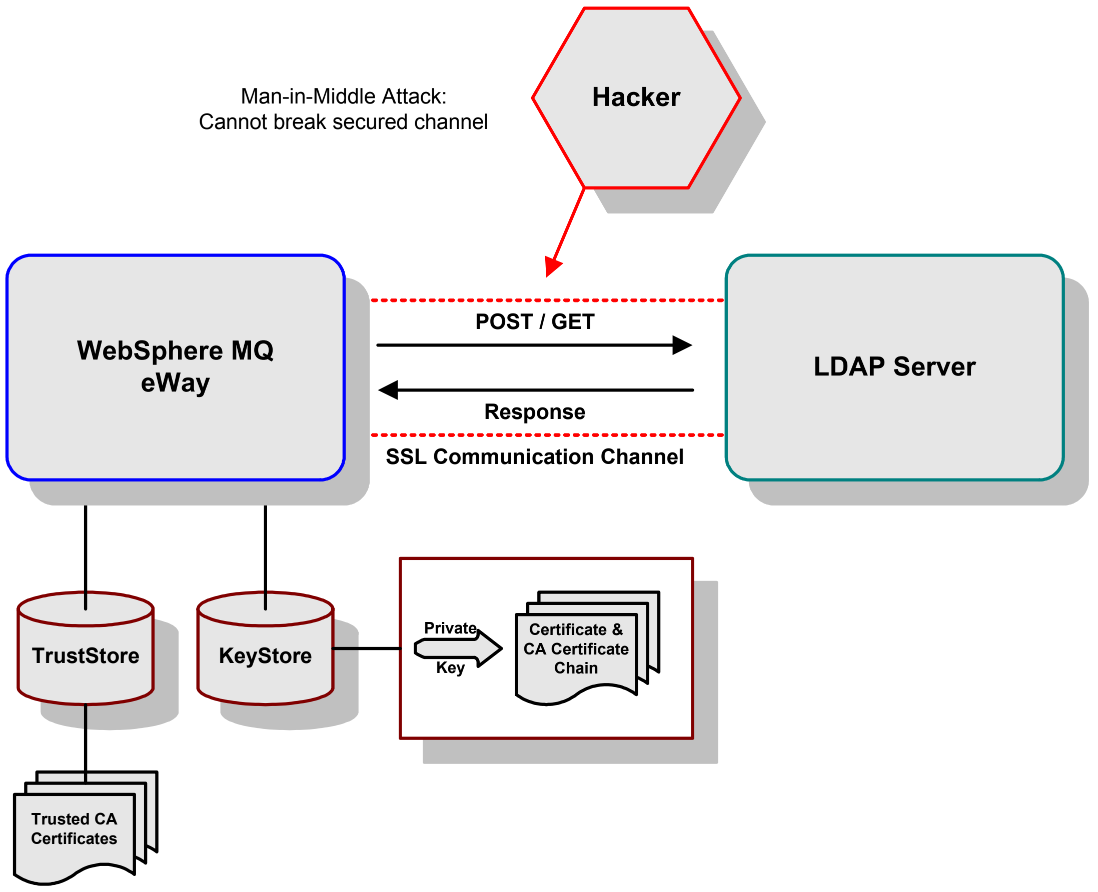
This chapter explains the operation of the Secure Sockets Layer (SSL) feature available with the HTTP(S) eWay.

What's in This Chapter

3.1 Overview

The use of SSL with WebSpehre MQ, enables WebSphere MQ data exchanges that are secure from unauthorized interception from "hackers" or other entities. The eWay's SSL feature provides a secure communications channel for the data exchanges (see Figure 2).

Figure 2 General SSL Operation: WebSphere MQ



This SSL feature is supported through the use of JSSE version 1.0.3.

Currently, the JSSE reference implementation is used. JSSE is a provider-based architecture, meaning that there is a set of standard interfaces for cryptographic algorithms, hashing algorithms, secured-socket-layered URL stream handlers, and so on.

Because the user is interacting with JSSE through these interfaces, the different components can be mixed and matched as long as the implementation is programmed under the published interfaces. However, some implementations may not support a particular algorithm.

The JSSE 1.0.3 application programming interface (API) is capable of supporting SSL versions 2.0 and 3.0 and Transport Layer Security (TLS) version 1.0. These security protocols encapsulate a normal bidirectional stream socket and the JSSE 1.0.3 API adds transparent support for authentication, encryption, and integrity protection. The JSSE reference implementation implements SSL version 3.0 and TLS 1.0.

For more information, visit the Sun Java Web site at the following URL:

http://java.sun.com

Note: *See the JSSE documentation provided by Sun Microsystems for further details.*

3.2 KeyStores and TrustStores

As depicted in Figure 2, JSSE makes use of files called *KeyStores* and *TrustStores*. The KeyStore is used by the eWay for client authentication, while the TrustStore is used to authenticate a server in SSL authentication.

- A *KeyStore* consists of a database containing a private key and an associated certificate, or an associated certificate chain. The certificate chain consists of the client certificate and one or more certification authority (CA) certificates.
- A *TrustStore* contains only the certificates trusted by the client (a "trust" store). These certificates are CA root certificates, that is, self-signed certificates. The installation of the Logical Host includes a TrustStore file named **cacerts.jks** in the location:

  ```
  <c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config
  ```

  where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain. This file is recommended as the TrustStore for the HTTPS eWay.

Both KeyStores and TrustStores are managed by means of a utility called **keytool**, which is a part of the Java SDK installation.

3.2.1 Generating a KeyStore and TrustStore

This section explains steps on how to create both a KeyStore and a TrustStore (or import a certificate into an existing TrustStore such as the default Logical Host TrustStore in the location:

```
<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\cacert
s.jks
```

where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain. The primary tool provided, is IBM Websphere MQ which is used to configure/create key stores, but **openssl** is also used as a reference for generating **pkcs12** KeyStores.

For more information on **openssl**, and available downloads, visit the following Web site:

http://www.openssl.org.

3.2.2 KeyStores

This section explains how to use KeyStores.

Creating a KeyStore in JKS Format

This section explains how to create a KeyStore using the JKS format as the database format for both the private key, and the associated certificate or certificate chain. By default, as specified in the java.security file, **keytool** uses JKS as the format of the key and certificate databases (KeyStore and TrustStores). A CA must sign the certificate signing request (CSR). The CA is therefore trusted by the server-side application to which the eWay is connected.

Note: *It is recommended to use the default KeyStore* `<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\keystore.jks` *where* `<c:\JavaCAPS>` *is the directory where the Sun Java Composite Application Platform Suite is installed and* `<MyDomain>` *is the name of your domain.*

To generate a KeyStore

Use the following command:

```
keytool -keystore clientkeystore -genkey -alias client
```

You are prompted for several pieces of information required to generate a CSR. A sample key generation section follows:

```
Enter keystore password: seebyond
What is your first and last name?
[Unknown]: development.seebeyond.com
What is the name of your organizational unit?
[Unknown]: Development
what is the name of your organization?
[Unknown]: SeeBeyond
What is the name of your City of Locality?
[Unknown]: Monrovia
What is the name of your State or Province?
[Unknown]: California
What is the two-letter country code for this unit?
[Unknown]: US
Is<CN=Foo Bar, OU=Development, O=SeeBeyond, L=Monrovia,
ST=California, C=US> correct?
[no]: yes

Enter key password for <client>
    (RETURN if same as keystore password):
```

If the KeyStore password is specified, then the password must be provided for the eWay. Press RETURN when prompted for the key password (this action makes the key password the same as the KeyStore password).

This operation creates a KeyStore file **clientkeystore** in the current working directory. You must specify a fully-qualified domain for the "first and last name" question. The reason for this use is that some CAs such as Verisign expect this properties to be a fully qualified domain name.

There are CAs that do not require the fully qualified domain, but it is recommended to use the fully-qualified domain name for the sake of portability. All the other information given must be valid. If the information can not be validated, a CA such as Verisign does not sign a generated CSR for this entry.

This KeyStore contains an entry with an alias of **client**. This entry consists of the Generated private key and information needed for generating a CSR as follows:

```
keytool -keystore clientkeystore -certreq alias client -keyalg rsa
    -file client.csr
```

This command generates a certificate signing request which can be provided to a CA for a certificate request. The file **client.csr** contains the CSR in PEM format.

Some CA (one trusted by the Web server to which the eWay is connecting) must sign the CSR. The CA generates a certificate for the corresponding CSR and signs the certificate with its private key. For more information, visit the following web sites:

http://www.thawte.com

or

http://www.verisign.com

If the certificate is chained with the CA's certificate, perform step 1; otherwise, perform step 2 in the following list:

1. The following command assumes the client certificate is in the file **client.cer** and the CA's certificate is in the file **CARoot.cer**:

   ```
   keytool -import -keystore clientstore -file client.cer -alias
       client
   ```

   This command imports the certificate (which can include more than one CA in addition to the Client's certificate).

   Also use the following command to import the CA's certificate into the KeyStore for chaining with the client's certificate:

   ```
   keytool -import -keystore clientkeystore -file CARootcer -alias
       theCARoot
   ```

2. The following command imports the client's certificate signed by the CA whose certificate was imported in the preceding step:

   ```
   keytool -import -keystore clientkeystore -file client.cer -alias
       client
   ```

The generated file **clientkeystore** contains the client's private key and the associated certificate chain used for client authentication and signing. The KeyStore and/or **clientkeystore**, can then be used as the eWay's KeyStore.

See the **"KeyStores" on page 24** for more information.

Creating a KeyStore in PKCS12 Format

This section explains how to create a PKCS12 KeyStore to work with JSSE. In a real working environment, a customer could already have an existing private key and certificate (signed by a known CA). In this case, JKS format can not be used, because it does not allow the user to import/export the private key through **keytool**. It is necessary to generate a PKCS12 database consisting of the private key and its certificate.

The generated PKCS12 database can then be used as the eWay's KeyStore. The **keytool** utility is currently lacking the ability to write to a PKCS12 database. However, it can read from a PKCS12 database.

Note: *There are additional third-party tools available for generating PKCS12 certificates, if you want to use a different tool.*

For the following example, **openssl** is used to generate the PKCS12 KeyStore:

```
cat mykey.pem.txt mycertificate.pem.txt>mykeycertificate.pem.txt
```

The existing key is in the file **mykey.pem.txt** in PEM format. The certificate is in **mycertificate.pem.txt**, which is also in PEM format. A text file must be created which contains the key followed by the certificate as follows:

```
openssl pkcs12 -export -in mykeycertificate.pem.txt -out
    mykeystore.pkcs12 -name myAlias -noiter -nomaciter
```

This command prompts the user for a password. The password is required. The KeyStore fails to work with JSSE without a password. This password must also be supplied as the password for the eWay's KeyStore password.

This command also uses the **openssl pkcs12** command to generate a PKCS12 KeyStore with the private key and certificate. The generated KeyStore is **mykeystore.pkcs12** with an entry specified by the **myAlias** alias. This entry contains the private key and the certificate provided by the **-in** argument. The **noiter** and **nomaciter** options must be specified to allow the generated KeyStore to be recognized properly by JSSE.

3.2.3 TrustStores

Creating a TrustStore

For demonstration purposes, suppose you have the following CAs that you trust: **firstCA.cert, secondCA.cert, thirdCA.cert**, located in the directory **C:\cascerts**. You can create a new TrustStore consisting of these three trusted certificates.

To create a new TrustStore

Use the following command:

```
keytool -import -file C:\cascerts\firstCA.cert -alias firstCA
    -keystore myTrustStore
```

You must enter this command two more times, but for the second and third entries, substitute **secondCA** and **thirdCA** for **firstCA**. Each of these command entries has the following purposes:

1. The first entry creates a KeyStore file name **myTrustStore** in the current working directory and imports the **firstCA** certificate into the TrustStore with an alias of **firstCA**. The format of **myTrustStore** is JKS.
2. For the second entry, substitute **secondCA** to import the **secondCA** certificate into the TrustStore, **myTrustStore**.
3. For the third entry, substitute **thirdCA** to import the **thirdCA** certificate into the TrustStore.

Once completed, myTrustStore is available to be used as the TrustStore for the eWay.

Using an Existing TrustStore

This section explains how to use an existing TrustStore such as the default Logical Host TrustStore in the location:

```
<c:\JavaCAPS>\logicalhost\is\domains\<MyDomain>\config\cacert
s.jks
```

where `<c:\JavaCAPS>` is the directory where the Sun Java Composite Application Platform Suite is installed and `<MyDomain>` is the name of your domain. The primary tool used is **keytool**, but **openssl** is also used as a reference for generating **pkcs12** KeyStores.

Notice that in the previous section, steps 2 and 3 were used to import two CAs into the TrustStore created in step 1. For example, suppose you have a trusted certificate file named: **C:\trustedcerts\foo.cert** and want to import it to the **trustedcacertsjks** TrustStore.

If you are importing certificates into an existing TrustStore, use:

```
keytool -import -file C:\cacerts\secondCA.cert -alias secondCA
    -keystore trustedcacertsjks
```

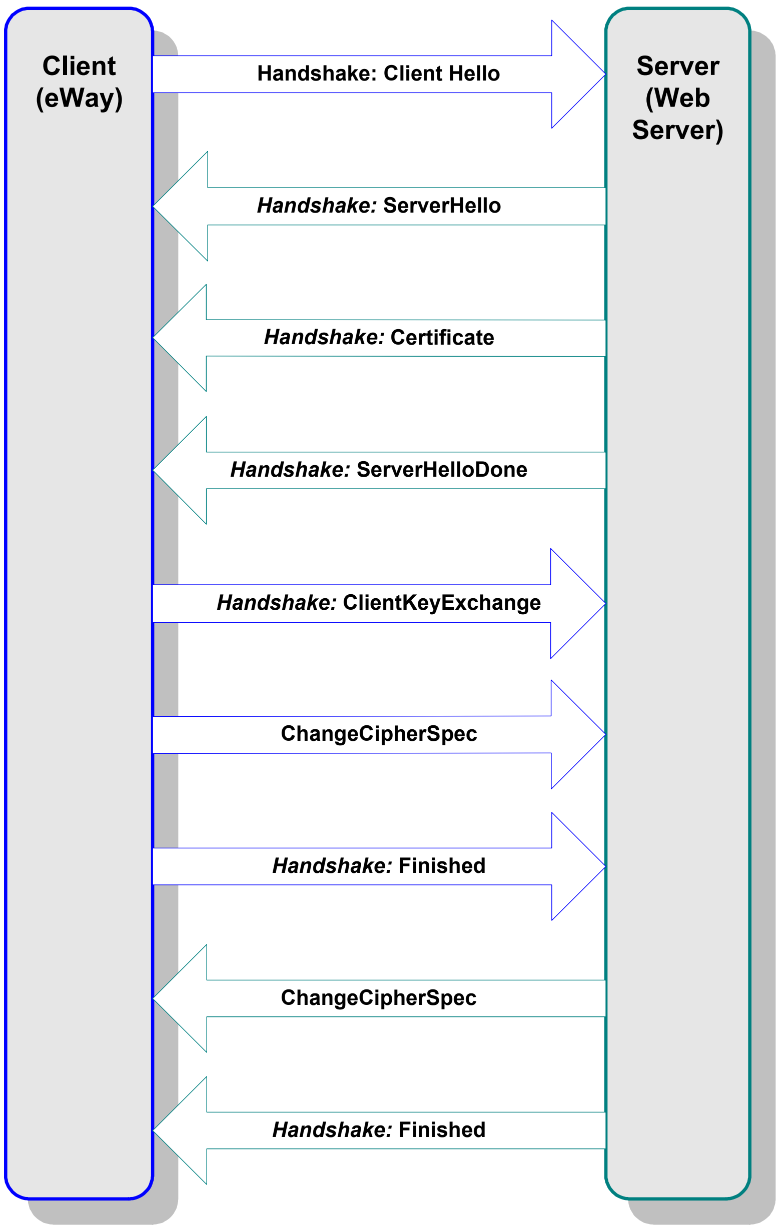
Once you are finished, **trustedcacertsjks** can be used as the TrustStore for the eWay.

3.3 SSL Handshaking

There are two options available for setting up SSL connectivity with a Web server:

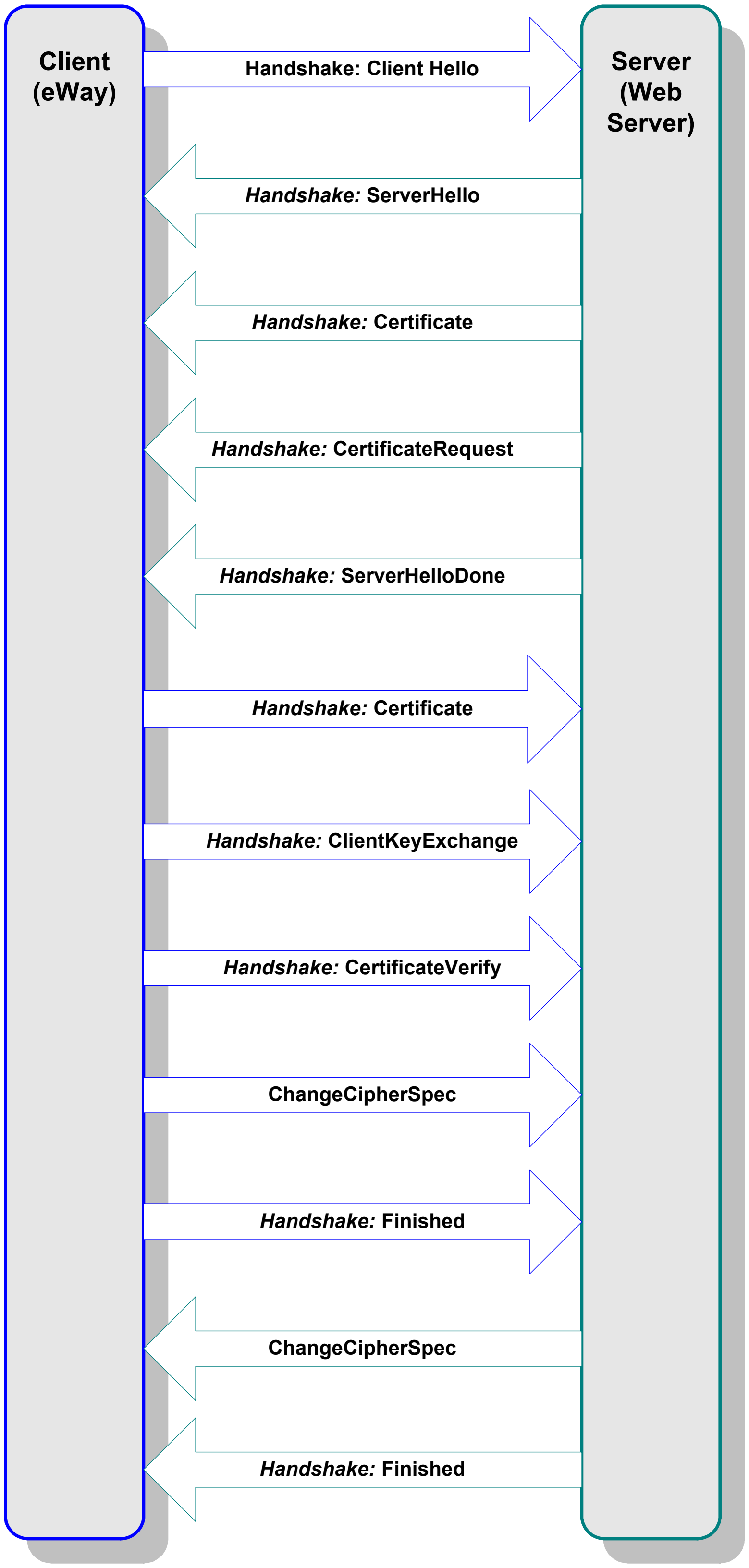
- **Server-side Authentication**: The majority of eCommerce Web sites on the Internet are configured for server-side authentication. The eWay requests a certificate from the Web server and authenticates the Web server by verifying that the certificate can be trusted. Essentially, the eWay performs this operation by looking into its TrustStore for a CA certificate with a public key that can validate the signature on the certificate received from the Web server. This option is illustrated in Figure 3.

Figure 3 Server-side Authentication



- **Dual authentication**: This option requires authentication from both the eWay and Web server. The server side (Web server) of the authentication process is the same as that described previously. In addition, however, the Web server requests a certificate from the eWay. The eWay then sends its certificate to the Web server. The server, in turn, authenticates the eWay by looking into its TrustStore for a matching trusted CA certificate. The communication channel is established by the process of both parties' requesting certificate information. This option is illustrated in Figure 4.

Figure 4 Dual Authentication



3.4 Creating a Certification Authority

The following steps describe how to create a Certification Authority (CA) using the command-line utilities supplied with WebSphere MQ.

1. Create a key repository for the CA.
2. Create a directory and in that directory, create a key repository file by entering the text shown below:

   ```
   C:\> mkdir \myCAdir
   C:\> cd \myCAdir
   C:\myCAdir> runmqckm -keydb -create -db myCA.kdb -type cms
   ```

   When prompted to create a password, type the password you want to use for the CA's key repository.
3. Create a self-signed CA certificate, which will be used to identify your CA:

   ```
   C:\myCAdir> runmqckm -cert -create -db myCA.kdb -type cms -label
   "myCAcertificate" -dn
   "CN=myCAName,O=myOrganisation,OU=myDepartment,L=myLocation,C=IN"
   -expire 1000 -size 1024
   ```
4. Extract the CA certficate into a file called myCAcertfile.cer, which you will later transfer to the key repositories of the queue manager and client application:

   ```
   C:\myCAdir> runmqckm -cert -extract -db myCA.kdb -type cms -label
   "myCAcertificate" -target myCAcertfile.cer -format ascii
   ```

3.4.1 Issuing a Certificate to a Queue Manager

Each queue manager in your infrastructure should have its own certificate, with an appropriate Distinguished Name (DN). The DN should be unique within the WebSphere MQ network.

1. Create the queue manager's key repository

   ```
   C:\myCAdir> mkdir \REPOS
   C:\myCAdir> cd \REPOS
   ```
2. Issue the following command to create a key database for the queue manager:

   ```
   C:\REPOS> runmqckm -keydb -create -db myqmgr.kdb -type cms -stash
   ```

   When prompted to create a password, type the password you want to use for the queue manager's key repository.

   The -stash option is important, as it causes a 'stash file' to be created. This file is called myqmgr.sth . It allows the queue manager to open the key repository without requesting a password from the user.
3. Generate a certificate request file for the queue manager, along with a private key:

   ```
   C:\REPOS> runmqckm -certreq -create -db myqmgr.kdb -type cms -dn
   "CN=QMNAME,O=SUN,OU=BI,L=BLR,C=IN" -label "ibmwebspheremqmyqmgr" -
   file myqmgr.req
   ```

   The label (as specified with the -label parameter) must be of the form ibmwebspheremqmyqmgr, all in lower case. This is important, as otherwise the queue manager will fail to find the certificate.

4 Transfer the certificate request file, myqmgr.req , to the directory where the CA files are located. Then change to the following directory:

```
C:\REPOS> copy myqmgr.req \myCAdir
C:\REPOS> cd \myCAdir
```

5 Sign the queue manager's certificate by running the following command:

```
C:\myCAdir> runmqckm -cert -sign -db myCA.kdb -label
"myCAcertificate" -expire 365 -format ascii -file myqmgr.req -
target myqmgr.cer
```

When prompted for the password, supply the CA key repository's password. Refer to the first step in **Creating a Certification Authority** on page 31.

6 Transfer the signed certificate (myqmgr.cer) and the public certificate of the CA (myCAcertfile.cer) back to C:\REPOS

```
C:\myCAdir> copy myqmgr.cer \REPOS
C:\myCAdir> copy myCAcertfile.cer \REPOS
C:\myCAdir> cd \REPOS
```

7 Add the public certificate of the CA to the key repository of the queue manager:

```
C:\REPOS> runmqckm -cert -add -db myqmgr.kdb -type cms -file
myCAcertfile.cer -label "theCAcert"
```

When prompted for a password, supply the queue manager key repository's password. Refer to the first step in **Issuing a Certificate to a Queue Manager** on page 31.

8 Receive the certificate (now signed by the CA) into the queue manager's key repository:

```
C:\REPOS> runmqckm -cert -receive -db myqmgr.kdb -type cms -file
myqmgr.cer
```

When prompted for a password, supply the queue manager key repository's password. Refer to **Issuing a Certificate to a Queue Manager** on page 31.

3.5 Issuing a Certificate to JCAPS

1 Create a certificate request to the Logical Host domain default keystore.jks.

```
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config>
runmqckm -certreq -create -db keystore.jks -type jks -dn
"CN=Client Identifier,O=SUN,OU=BI,L=BLR,C=IN" -label
"ibmwebspheremqmyuserid" -file myappj.req
```

When prompted to create a password, type the default password **changeit** for the Logical Host. The certificate label chosen was ibmwebspheremqmyuserid .

2 Transfer the certificate request file (myappj.req) to the directory where the CA files are located, then change to this directory:

```
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config> copy
myappj.req C:\myCAdir
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config> cd
C:\myCAdir
```

3 Sign the application's certificate by running the following:

```
C:\myCAdir> runmqckm -cert -sign -db myCA.kdb -label
"myCAcertificate" -expire 365 -format ascii -file myappj.req -
target myappj.cer
```

When prompted for a password, supply the CA key repository's password. Refer to the first step in **Creating a Certification Authority** on page 31.

4 Transfer the signed certificate (myappj.cer) and the public certificate of the CA (myCAcertfile.cer) back to C:\MYAPPJ:

```
C:\myCAdir> copy myappj.cer
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config\
C:\myCAdir> copy
myCAcertfile.cer<JavaCAPS513>\logicalhost\is\domains\<domain_name>
\config
C:\myCAdir> cd
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config
```

5 Add the CA certificate to the JCAPS keystore:

```
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config>
runmqckm -cert -add -db keystore.jks -type jks -file
myCAcertfile.cer -label "theCAcertificate"
```

When prompted for a password, supply the JCAPS keystore password as **changeit**.

6 Receive the certificate (now signed by the CA) into the JCAPS keystore:

```
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config>
runmqckm -cert -receive -db keystore.jks -type jks -file
myappj.cer
```

When prompted for a password, supply the JCAPS keystore password as **changeit**.

7 Add the CA certificate to truststore:

```
<JavaCAPS513>\logicalhost\is\domains\<domain_name>\config>runmqckm
-cert -add -db cacerts.jks -type jks -file myCAcertfile.cer -label
"theCAcertificate"
```

Chapter 4

Configuring the WebSphere eWay

This chapter describes how to create and configure the WebSphere MQ eWay.

What's in This Chapter

4.1 Creating and Configuring the WebSphere MQ eWay

All eWays contain a set of parameters with properties unique to that eWay type. After the eWays are established and a WebSphere MQ External System is created in the Project's Environment, the eWay parameters can be modified for your specific system. The WebSphere MQ eWay properties are modified from these locations:

- From the **Connectivity Map**. These properties most commonly apply to a specific eWay, and may vary from other eWays (of the same type) in the Project. The WebSphere MQ Connectivity Map properties will vary depending on whether the eWay is an Inbound or Outbound eWay.
- From the **Environment Explorer tree**. These properties are commonly global, applying to all eWays (of the same type) in the Project. The saved properties are shared by all eWays in the WebSphere MQ External System window.
- **Collaboration or Business Process**: WebSphere MQ 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.

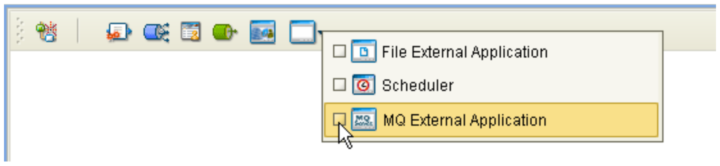
4.1.1 Selecting WebSphere MQ as the External Application

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

To create the WebSphere MQ External Application

1. From the Connectivity Map toolbar, click the **External Applications** icon.
2. Select the **WebSphere MQ External Application** from the menu (see Figure 5). The selected WebSphere MQ External Application icon appears on the Connectivity Map toolbar.

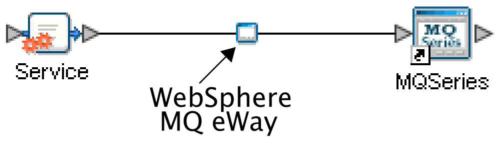
Figure 5 External Applications Selection Menu



3. Drag the new **WebSphere MQ External Application** from the toolbar onto the Connectivity Map canvas. This represents an external WebSphere MQ system.

   From the Connectivity Map, you can associate (bind) the External Application with the Service to establish an eWay (see Figure 6).

Figure 6 eWay Location



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

4.1.2 Creating Custom Properties for a WebSphere MQ eWay

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

Modifying the WebSphere MQ eWay (Connectivity Map) Properties

1. From the Connectivity Map, double click the eWay icon, located in the link between the associated External Application and the Service. If this link is for an outbound eWay, the Templates dialog box appears.
2. From the Templates dialog box, select **Outbound MQ Series eWay or Outbound MQ Series eWay (XA)** as the eWay configuration type and click **OK**. Skip this step for inbound eWays.

3 The eWay **Properties Editor** opens to the Connectivity Map properties. Make any necessary modifications and click **OK** to save the settings.

Modifying the WebSphere MQ eWay (Environment Explorer) Properties

1 From the Environment Explorer tree, right-click the Inbound or Outbound WebSphere MQ external system. Select **Properties** from the shortcut menu. The **Properties Editor** appears.

2 Make any necessary modifications to the Environment parameters of the WebSphere MQ eWays, and click **OK** to save the settings.

4.1.3 Using the Properties Editor

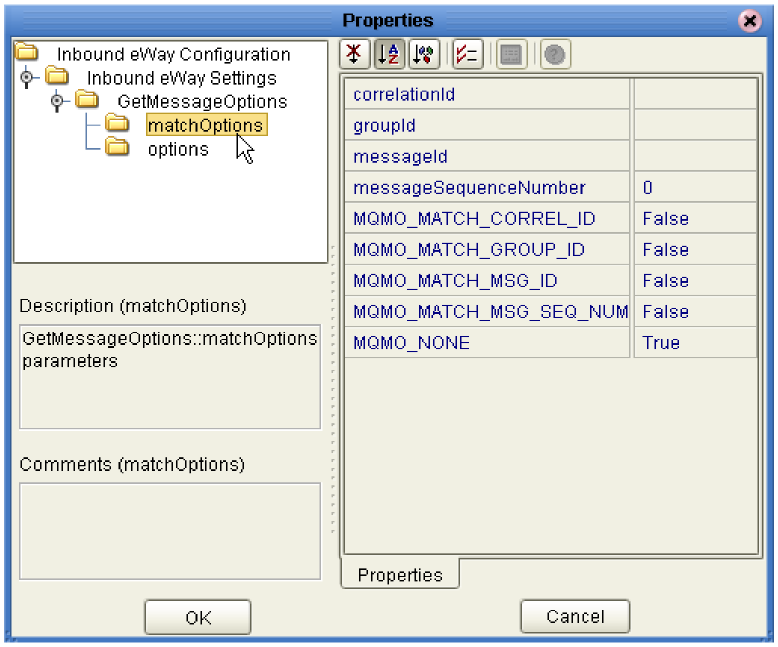
Modifications to the eWay configuration properties are made from the WebSphere MQ eWay Properties Editor.

To modify the default eWay configuration properties do the following:

1 Open the Properties Editor to the WebSphere MQ eWay properties you want to edit. The WebSphere MQ Inbound and Outbound eWays have two sets of parameters: those specific to that particular eWay (accessed from the **Connectivity Map**), and those that are common to all eWays of this type (accessed from the **Environment Explorer** tree).

2 From the upper-left pane of the Properties Editor, select a properties directory. The parameters contained in that directory are now displayed in the right pane of the Properties Editor. For example, from the Inbound eWay Connectivity Map Properties, click on the **matchOptions** properties directory to display this section's editable parameters in the right pane, as shown in Figure 7.

Figure 7 Properties Editor -- WebSphere MQ Properties



Description (matchOptions)

GetMessageOptions::matchOptions parameters

Comments (matchOptions)

correlationId	
groupId	
messageId	
messageSequenceNumber	0
MQMO_MATCH_CORREL_ID	False
MQMO_MATCH_GROUP_ID	False
MQMO_MATCH_MSG_ID	False
MQMO_MATCH_MSG_SEQ_NUM	False
MQMO_NONE	True

3 Click on any property field to make it editable. For example, click on the **messageId** parameter to edit the messageId value. If a parameter's value is true/false or multiple choice, the field, when selected, reveals a submenu of property options. If a parameter requires that you type in a value, such as a name or password, the property field provides space to type in the value and an ellipsis (. . .) button.

   Click on the ellipsis (. . .) in the properties field to open a separate configuration dialog box. This is helpful for entering 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.

4 A description of each parameter is displayed in the **Description** pane when that parameter is selected, providing an explanation of any required settings or options.

5 The **Comments** pane provides an area for recording notes and information regarding the currently selected parameter. This is saved for future referral.

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

4.2 WebSphere MQ eWay Properties

The WebSphere MQ eWay's Properties are organized as follows:

Inbound WebSphere MQ eWay Connectivity Map Properties on page 38

Outbound WebSphere MQ eWay XA Connectivity Map Properties on page 46

Outbound WebSphere MQ eWay Connectivity Map Properties on page 51

WebSphere MQ eWay Environment Properties on page 56

Note: *Creating customized individual OTD configuration settings can override the default eWay OTD configuration settings.*

4.3 Inbound WebSphere MQ eWay Connectivity Map Properties

The inbound WebSphere MQ eWay parameters, accessed from the Connectivity Map, are organized into the following sections:

- **Inbound eWay Settings** on page 38
- **Inbound eWay Settings > GetMessageOptions > matchOptions** on page 41
- **Inbound eWay Settings > GetMessageOptions > options** on page 43

4.3.1 Inbound eWay Settings

The **Inbound eWay Settings** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 3.

Table 3 Connectivity Map - Inbound eWay Settings

Name	Description	Required Value
Queue Name	Specifies the name of the queue from which messages are picked up (subscribed). ***Note:*** *Inbound mode eWays will not function if a non-local queue is specified. Non-local queues include alias queues and local queue definitions to remote queues. This limitation applies only applies to polling/ receive mode eWays, not outbound eWays used to retrieve messages.* *See* **"Accessing Non-Local Queue Managers and Non-Local Queues" on page 67**.	The name of the WebSphere MQ queue. The configured default is **default**.

Table 3 Connectivity Map - Inbound eWay Settings (Continued)

Name	Description	Required Value
Maximum Message Size	Specifies the maximum message size that the eWay is able to get from a queue. A value of zero (**0**) instructs the eWay to use the MQ-provided default size. If you specify a non-zero value, and a message on the queue is larger than this value, one of the following will occur: 1 If the **MQC.MQGMO_ACCEPT_TRUNCATED_MSG** property is set to **TRUE**, the eWay processes as much of the message from the queue as possible, producing a truncated message. 2 If the **MQC.MQGMO_ACCEPT_TRUNCATED_MSG** property is set to **FALSE**, the eWay leaves the message on the queue and raises an MQException with a completion code of **MQCC_WARNING**, and with a reason code of **MQRC_TRUNCATED_MSG_FAILED**.	A number indicating the maximum message size in bytes. The configured default is **0** (use the MQ defined default size).
Schedule Interval	Specifies the polling interval in milliseconds at which the subscribed queue is polled for messages. This is the duration of the pause, in milliseconds, between attempts to get messages from the queue.	The number of milliseconds at which the queue is polled. The configured default is **5000** (or 5 seconds).
Security Exit	Specifies the optional, package qualified name of a user-defined class that implements the **com.ibm.mq.MQSecurityExit** interface. WebSphere MQ invokes an instance of the class whenever the eWay attempts to connect to the queue manager. The named class must include a default constructor. This parameter is only used for client connections. Security Exits are not applicable to bindings connections. For more information about Security Exits, see the IBM document, "WebSphere MQ Using Java" (CSQZAW09) regarding MQSecurityExit.	The name of the user-defined class. This property also requires an assigned value for the property, **"Security Exit JAR Classpath" on page 40**.

Table 3 Connectivity Map - Inbound eWay Settings (Continued)

Name	Description	Required Value
Security Exit JAR Classpath	Specifies the absolute path to the JAR file that contains the named Security Exit. This property is required if the Security Exit is specified. The specified JAR is packaged into the application (EAR) file that is generated during Project activation. If the specified JAR cannot be accessed or found, the activation will fail. If this property value is left blank, you must ensure that a JAR file containing the Security Exit is made accessible to the runtime Environment prior deploying the Project (for example, by manually copying the JAR file into the Integration Server's lib directory prior to or during the Deployment Process). For more information about Security Exits, see the IBM document, "WebSphere MQ Using Java" (CSQZAW09) regarding MQSecurityExit.	The the absolute path for the JAR file that contains the named Security Exit class.

4.3.2 Inbound eWay Settings > GetMessageOptions > matchOptions

The **Inbound eWay Settings > GetMessageOptions > matchOptions** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 4.

Table 4 Connectivity Map - Inbound eWay Settings > GetMessageOptions > matchOptions

Name	Description	Required Value
correlationId	Specifies the correlation identifier of the message to be retrieved. Normally the queue manager returns the first message with a message identifier and correlation identifier that matches the identifiers specified.	The correlation identifier of the message.
groupId	Specifies the byte string that identifies the message group to which the physical message belongs.	A byte string that indicates the message group.
messageId	For an MQGET call, this field specifies the message identifier of the message to be retrieved. Normally, the queue manager returns the first message with a message identifier and correlation identifier that matches those identifiers specified. For an MQPUT call, this specifies the message identifier to use.	The message identifier.
messageSequence Number	Specifies the sequence number of a logical message within a group.	The sequence number of the logical message within a group. The configured default is **1**.
MQMO_MATCH_ CORREL_ID	Specifies that the retrieved message must have a correlation identifier that matches the value of the correlationId parameter. The values are: ▪ **True**: Indicates that the message must have a matching correlation identifier. ▪ **False**: Indicates that the correlation identifier is ignored and any correlation identifier will be accepted. This match is in addition to any other matches that may apply (for example, the message identifier).	**True** or **False**. The configured default is **False**.

Table 4 Connectivity Map - Inbound eWay Settings > GetMessageOptions > matchOptions

Name	Description	Required Value
MQMO_MATCH_ GROUP_ID	Specifies that the retrieved message must have a group identifier that matches the value of the groupId parameter. The values are: ▪ **True**: Indicates that the message must have a matching group identifier. ▪ **False**: Indicates that the group identifier is ignored and any group identifier is accepted. This match is in addition to any other matches that may apply (for example, the correlation identifier).	**True** or **False**. The configured default is **False**.
MQMO_MATCH_ MSG_ID	Specifies that the retrieved message must have a message identifier that matches the value of the messageId parameter. The values are: ▪ **True**: Indicates that the message must have a matching message identifier. ▪ **False**: Indicates that the message identifier is ignored and any message identifier is accepted. This match is in addition to any other matches that may apply (for example, the correlation identifier).	**True** or **False**. The configured default is **False**.
MQMO_MATCH_ MSG_SEQ_NUMBER	Specifies that the retrieved message must have a message sequence number that matches the value of the messageSequenceNumber parameter. The values are: ▪ **True**: Indicates that the message must have a matching message sequence number. ▪ **False**: Indicates that the message sequence number is ignored and any message sequence number is accepted. This match is in addition to any other matches that may apply (for example, the group identifier).	**True** or **False**. The configured default is **False**.

Table 4 Connectivity Map - Inbound eWay Settings > GetMessageOptions > matchOptions

Name	Description	Required Value
MQMO_NONE	Specifies that no matches are to be used in selecting the message to be returned. All messages on the queue are eligible for retrieval (subject to some MQGMO_ options...).	**True** or **False**. The configured default is **True**.

4.3.3 Inbound eWay Settings > GetMessageOptions > options

The **Inbound eWay Settings > GetMessageOptions > options** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 5.

Table 5 Connectivity Map - Inbound eWay Settings > GetMessageOptions > options

Name	Description	Required Value
MQGMO_ACCEPT_ TRUNCATED_MSG	Specifies whether a truncated message is accepted as a complete message. If the message buffer is too small to hold the complete message, this option allows the **MQGET** call to fill the buffer with as much as it can hold and complete its processing. Without this option, in the given situation, the **MQGET** call will still be filled to capacity, but the processing will not be considered completed. The values are: ▪ **True**: Indicates that a truncated message is accepted as a complete message. ▪ **False**: Indicates that a truncated message is not considered as a complete message.	**True** or **False**. The configured default is **True**.
MQGMO_FAIL_IF_ QUIESCING	Forces the **MQGET** call to fail if the queue manager is in the quiescing state. The values are: ▪ **True**: Indicates that calling **MQGET** fails if the queue manager is in the quiescing state. ▪ **False**: Indicates that calling **MQGET** does not fail if the queue manager is in the quiescing state.	**True** or **False**. The configured default is **True**.

Table 5 Connectivity Map - Inbound eWay Settings > GetMessageOptions > options

Name	Description	Required Value
MQGMO_SYNCPOINT	Forces the **MQGET** call to get the message under syncpoint control; the message is marked as being unavailable to other applications, but it is deleted from the queue only when the unit of work is committed. The message is made available again if the unit of work is backed out. The values are: ▪ **True**: Indicates that calling **MQGET** gets the message under syncpoint control. ▪ **False**: Indicates that **MQGET**, when called does not get the message under syncpoint control.	**True** or **False**. The configured default is **False**.
MQGMO_SYNCPOINT_ IF_PERSISTENT	Forces the **MQGET** call to get the message under syncpoint control if the message is persistent. The values are: ▪ **True**: Indicates that calling **MQGET** gets the message under syncpoint control if the message is persistent. ▪ **False**: Indicates that **MQGET**, when called does not get the message under syncpoint control if the message is persistent.	**True** or **False**. The configured default is **False**.
MQGMO_COMPLETE_ MSG	Specifies that only a complete logical message can be returned by calling **MQGET**. If the logical message is segmented, the queue manager reassembles the segments and returns the complete logical message to the application; the fact that the logical message was segmented is not apparent to the eWay. The values are: ▪ **True**: Indicates that only a complete logical message can be returned by calling **MQGET**. ▪ **False**: Indicates that a complete logical message is not required.	**True** or **False**. The configured default is **False.**
MQGMO_WAIT	Specifies that an **MQ GET** call waits (block/ suspend) until a message becomes available in the queue. The values are: ▪ **True**: Indicates that an **MQ GET** call waits until a message becomes available in the queue. ▪ **False**: Indicates that an **MQ GET** call does not wait until a message becomes available in the queue.	**True** or **False**. The configured default is **False**.

Table 5 Connectivity Map - Inbound eWay Settings > GetMessageOptions > options

Name	Description	Required Value
waitInterval	Specifies how long (in milliseconds) an **MQ GET** call waits for a message to become available in the queue. This parameter is used in conjunction with **MQGMO_WAIT**. If **MQGMO_WAIT** is set to false, waitInterval is not used. Specifying a negative value indicates that the wait will last indefinitely. Setting this value to a negative number causes the polling eWay to execute **MQ GET** calls with a wait interval of **MQWI_UNLIMITED**. With this type of get call, the eWay will block indefinitely until a suitable message is available. If the Integration Server (in association with the Logical Host) is commanded to shut down or restart while the eWay is still blocked, the Integration Server will not be able to proceed until the eWay is unblocked by the availability of a suitable MQ message. The same limitation affects the **non-polling** use of the eWay. The WebSphere MQ eWay's **OTD GMO** structure exposes a method named **setUnlimitedWait()** to Java Collaborations that, when used, sets the **waitInterval** to the value **MQWI_UNLIMITED**. If using **setUnlimitedWait()** causes the eWay to block indefinitely during a subsequent get call, the Integration Server will be unable to shut down until the eWay is unblocked.	A number indicating the period of time, in milliseconds, that an **MQ GET** call waits for a message to become available in the queue. The configured default is **0**.

4.4 Outbound WebSphere MQ eWay XA Connectivity Map Properties

The outbound WebSphere MQ eWay XA parameters, accessed from the Connectivity Map, are contained in the following section.

- **Outbound eWay (XA) Settings** on page 47
- **Outbound eWay (XA) Settings > Queue Open Options** on page 48

Running XA Transactions in Client (Outbound) Mode

The outbound WebSphere MQ eWay supports XA transactions with WebSphere MQ Manager servers running on Solaris, AIX, HP-UX, Linux or Windows (not supported for OS/400 or z/OS). If your Sun Java Composite Application Platform Suite is installed on a different computer than your WebSphere MQ server, XA mode requires that you first install the WebSphere MQ base client, and then the WebSphere MQ Extended Transactional client, on the Logical Host.

The transaction manager of the WebSphere MQ eWay in XA mode, runs in the Sun SeeBeyond Integration Server. The Integration Server (IS) requires two JAR files, **com.ibm.mq.jar** and **com.ibm.mqetclient.jar**, to be added to the Integration Server classpath. For more information on these two JAR files, see **"WebSphere MQ eWay Required JAR Files" on page 17**. For direction on adding these JAR files to the IS classpath, see **"Adding XA Required JAR Files to the Integration Server Classpath" on page 17**.

The IBM document *WebSphere MQ External Transactional Clients*, provides information on distributed XA transactions and limitations, such as WebSphere MQ API that cannot be issued in XA mode.

According to IBM, when using the WebSphere MQ Extended Transactional client, a client application can be connected to only one queue manager at a time within a single thread. This restriction applies to the WebSphere MQ Extended Transactional client. (The WebSphere MQ base client can be connected to more than one queue manager concurrently within a single thread.) For the WebSphere MQ eWay this means, in one deployment, you are only allowed to have one outbound XA mode eWay connecting to a WebSphere MQ external system. You cannot have multiple XA outbound connections to different WebSphere MQ external systems and expect the Integration Server Transactional manager to handle XA transaction for multiple WebSphere MQ queue managers.

4.4.1 Outbound eWay (XA) Settings

The **Outbound eWay (XA) Settings** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 6.

Table 6 Connectivity Map - Outbound eWay Settings

Name	Description	Required Value
Queue Name	Specifies the name of queue to which the message is published. This parameter is optional. The queue name may also be specified manually in the Business Process or Collaboration that effects the put.	The queue name to which the message is retrieved or published.
Security Exit	Specifies the optional, package qualified name of a user-defined class that implements the **com.ibm.mq.MQSecurityExit** interface. WebSphere MQ invokes an instance of the class whenever the eWay attempts to connect to the queue manager. The named class must include a default constructor. This parameter is only used for client connections. Security Exits are not applicable to bindings connections. For more information about Security Exits, see the IBM document, "WebSphere MQ Using Java" (CSQZAW09) regarding MQSecurityExit.	The name of the user-defined class. This property also requires an assigned value for the property, **"Security Exit JAR Classpath" on page 40**.
Security Exit JAR Classpath	Specifies the absolute path to the JAR file that contains the named Security Exit. This property is required if the Security Exit is specified. The specified JAR is packaged into the application (EAR) file that is generated during Project activation. If the specified JAR cannot be accessed or found, the activation will fail. If this property value is left blank, you must ensure that a JAR file containing the Security Exit is made accessible to the runtime Environment prior deploying the Project (for example, by manually copying the JAR file into the Integration Server's lib directory prior to or during the Deployment Process). For more information about Security Exits, see the IBM document, "WebSphere MQ Using Java" (CSQZAW09) regarding MQSecurityExit.	The the absolute path for the JAR file that contains the named Security Exit class.

4.4.2 Outbound eWay (XA) Settings > Queue Open Options

The **Outbound eWay Settings > Queue Open Options** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 7. These properties apply specifically to the **MQOPEN** calls.

Table 7 Connectivity Map - Outbound eWay Settings > Queue Open Options

Name	Description	Required Value
MQOO_ALTERNATE_USER_ AUTHORITY	Specifies whether an alternate user identifier is used to check the authorization for the open. **True** indicates that and alternate user identifier is used.	**true** or **false**. The configured default is **false.**
MQOO_BIND_AS_Q_DEF	Specifies whether the binding used for the queue handle is taken from the **DefBind** queue attribute. **True** indicates that the binding used is taken from **DefBind** queue attribute.	**true** or **false**. The configured default is **false.**
MQOO_BIND_NOT_FIXED	Specifies whether the local queue manager binds the queue handle to a particular instance of the destination queue, when the object being opened is a cluster queue. **True** indicates that the local queue manager **will** bind to a specific destination. This option is ignored when specified for a queue that is not a cluster queue.	**true** or **false**. The configured default is **false.**
MQOO_BIND_ON_OPEN	Specifies whether the local queue manager binds the queue handle to a particular instance of the destination queue, when the object being opened is a cluster queue. **True** indicates that the local queue manager **will not** bind to a specific destination. This option is ignored when specified for a queue that is not a cluster queue.	**true** or **false**. The configured default is **false.**
MQOO_BROWSE	Specifies whether the queue is opened to browse messages. **True** indicates that the queue is open for use with **MQGET** calls with the following options: ♦ **MQGMO_BROWSE_FIRST** ♦ **MQGMO_BROWSE_NEXT** ♦ **MQGMO_BROWSE_MSG_UNDER_CURSOR**	**true** or **false**. The configured default is **false.**
MQOO_FAIL_IF_QUIESCING	Specifies whether the **MQOPEN** call fails when the queue manager is in quiescing state. Used to control authorization checking. **True** indicates that the **MQOPEN** call will fail if queue manager is quiescing.	**true** or **false**. The configured default is **false.**

Table 7 Connectivity Map - Outbound eWay Settings > Queue Open Options (Continued)

Name	Description	Required Value
MQOO_INPUT_AS_Q_DEF	Specifies whether the queue is opened to browse messages using the queue-defined default. **True** indicates that the queue is open for use with subsequent **MQGET** calls. ***Note:*** The value of this parameter is ignored when the eWay is operating in automatic connection mode, because the eWay must be capable of both receiving and sending messages.	**true** or **false**. The configured default is **true.**
MQOO_INPUT_EXCLUSIVE	Specifies whether the queue is opened to get messages with exclusive access. **True** indicates that the queue is open for use with subsequent **MQGET** calls. Calls will fail with reason code **MQRC_OBJECT_IN_USE** if the queue is currently used (open) by this or another application for input of any type.	**true** or **false**. The configured default is **false.**
MQOO_INPUT_SHARED	Specifies whether the queue is opened to get messages with shared access. **True** indicates that the queue is open for use with subsequent **MQGET** calls. Calls will succeed, even when the queue is currently used (open) by this or another application for input of any type.	**true** or **false**. The configured default is **false.**
MQOO_INQUIRE	Specifies whether the object is opened to inquire attributes. **True** indicates that the queue, name list, process definition, or queue manager is open for use with subsequent **MQINQ** calls.	**true** or **false**. The configured default is **true.**
MQOO_OUTPUT	Specifies whether the object is opened to put messages. **True** indicates that MQOPEN call can succeed, even if the InhibitPut queue attribute is set to **MQQA_PUT_INHIBITED** (though subsequent **MQPUT** calls will fail). ***Note:*** The value of this parameter is ignored when the eWay is operating in automatic connection mode, because the eWay must be capable of both receiving and sending messages.	**true** or **false**. The configured default is **true.**

Table 7 Connectivity Map - Outbound eWay Settings > Queue Open Options (Continued)

Name	Description	Required Value
MQOO_PASS_ALL_CONTEXT	Specifies whether to allow all context to pass. **True** indicates that the **MQPMO_PASS_ALL_CONTEXT** option is specified in the PutMsgOpts parameter when a message is put on a queue, and gives the message identity and origin context information from an input queue opened with the **MQOO_SAVE_ALL_CONTEXT** option. **True** also indicates that **MQOO_PASS_ IDENTITY_CONTEXT** is implied and does not need to be specified.	**true** or **false**. The configured default is **false.**
MQOO_PASS_IDENTITY_ CONTEXT	Specifies whether to allow identity context to pass. **True** indicates that the **MQPMO_PASS_IDENTITY_CONTEXT** option to be specified in the **PutMsgOpts** parameter when a message is put on a queue. This gives the message the identity context information from an input queue opened with the MQOO_SAVE_ALL_CONTEXT option. True indicates that the MQOO_OUTPUT option must be specified.	**true** or **false**. The configured default is **false.**
MQOO_RESOLVE_NAMES	Specifies MQOO_RESOLVE_NAMES. Select **True** if you want to use the **resolved queue manager name** and **resolved queue name attributes** of the **ImqQueue** class.	**true** or **false**. The configured default is **false.**
MQOO_SAVE_ALL_CONTEXT	Specifies whether to save context when message is retrieved. True indicates that context is saved. Context information is associated with this queue handle and set from the context of any message retrieved using this handle.	**true** or **false**. The configured default is **false.**
MQOO_SET	Specifies whether the queue is open to set attributes. True indicates that the queue is open to set attributes and for use with subsequent MQSET calls.	**true** or **False**. The configured default is **False.**
MQOO_SET_ALL_CONTEXT	Specifies whether to allow all context to be set. **True** indicates that all context can be set. The MQPMO_SET_ALL_CONTEXT option is specified in the PutMsgOpts parameter when a message is put on a queue. Gives the identity and origin context information contained in the MsgDesc parameter specified on the MQPUT or MQPUT1 call to the message.	**true** or **false**. The configured default is **false.**

Table 7 Connectivity Map - Outbound eWay Settings > Queue Open Options (Continued)

Name	Description	Required Value
MQOO_SET_IDENTITY_CONTEXT	Specifies whether to allow identity context to be set. **True** indicates that identity context can be set. The MQPMO_SET_IDENTITY_CONTEXT option can be specified in the PutMsgOpts parameter when a message is put on a queue. Gives the identity and origin context information contained in the MsgDesc parameter specified on the MQPUT or MQPUT1 call to the message.	**true** or **false**. The configured default is **false.**

4.5 Outbound WebSphere MQ eWay Connectivity Map Properties

The outbound WebSphere MQ eWay parameters, accessed from the Connectivity Map, are contained in the following section.

- **Outbound eWay Settings** on page 51
- **Outbound eWay Settings > Queue Open Options** on page 53

4.5.1 Outbound eWay Settings

The **Outbound eWay Settings** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 8.

Table 8 Connectivity Map - Inbound eWay Settings > GetMessageOptions > options

Name	Description	Required Value
Queue Name	Specifies the name of queue to which the message is published. This parameter is optional. The queue name may also be specified manually in the Business Process or Collaboration that effects the put.	The queue name to which the message is retrieved or published.

Table 8 Connectivity Map - Inbound eWay Settings > GetMessageOptions > options

Name	Description	Required Value
Security Exit	Specifies the optional, package qualified name of a user-defined class that implements the **com.ibm.mq.MQSecurityExit** interface. WebSphere MQ invokes an instance of the class whenever the eWay attempts to connect to the queue manager. The named class must include a default constructor. This parameter is only used for client connections. Security Exits are not applicable to bindings connections. For more information about Security Exits, see the IBM document, "WebSphere MQ Using Java" (CSQZAW09) regarding MQSecurityExit.	The name of the user-defined class. This property also requires an assigned value for the property, **"Security Exit JAR Classpath" on page 40**.
Security Exit JAR Classpath	Specifies the absolute path to the JAR file that contains the named Security Exit. This property is required if the Security Exit is specified. The specified JAR is packaged into the application (EAR) file that is generated during Project activation. If the specified JAR cannot be accessed or found, the activation will fail. If this property value is left blank, you must ensure that a JAR file containing the Security Exit is made accessible to the runtime Environment prior deploying the Project (for example, by manually copying the JAR file into the Integration Server's lib directory prior to or during the Deployment Process). For more information about Security Exits, see the IBM document, "WebSphere MQ Using Java" (CSQZAW09) regarding MQSecurityExit.	The the absolute path for the JAR file that contains the named Security Exit class.

4.5.2 Outbound eWay Settings > Queue Open Options

The **Outbound eWay Settings > Queue Open Options** section of the WebSphere MQ eWay Connectivity Map properties contains the top-level parameters displayed in Table 7. These properties apply specifically to the **MQOPEN** calls.

Table 9 Connectivity Map - Outbound eWay Settings > Queue Open Options

Name	Description	Required Value
MQOO_ALTERNATE_USER_ AUTHORITY	Specifies whether an alternate user identifier is used to check the authorization for the open. **True** indicates that and alternate user identifier is used.	**true** or **false**. The configured default is **false.**
MQOO_BIND_AS_Q_DEF	Specifies whether the binding used for the queue handle is taken from the **DefBind** queue attribute. **True** indicates that the binding used is taken from **DefBind** queue attribute.	**true** or **false**. The configured default is **false.**
MQOO_BIND_NOT_FIXED	Specifies whether the local queue manager binds the queue handle to a particular instance of the destination queue, when the object being opened is a cluster queue. **True** indicates that the local queue manager **will** bind to a specific destination. This option is ignored when specified for a queue that is not a cluster queue.	**true** or **false**. The configured default is **false.**
MQOO_BIND_ON_OPEN	Specifies whether the local queue manager binds the queue handle to a particular instance of the destination queue, when the object being opened is a cluster queue. **True** indicates that the local queue manager **will not** bind to a specific destination. This option is ignored when specified for a queue that is not a cluster queue.	**true** or **false**. The configured default is **false.**
MQOO_BROWSE	Specifies whether the queue is opened to browse messages. **True** indicates that the queue is open for use with **MQGET** calls with the following options: ♦ **MQGMO_BROWSE_FIRST** ♦ **MQGMO_BROWSE_NEXT** ♦ **MQGMO_BROWSE_MSG_UNDER_CURSOR**	**true** or **false**. The configured default is **false.**

Table 9 Connectivity Map - Outbound eWay Settings > Queue Open Options (Continued)

Name	Description	Required Value
MQOO_FAIL_IF_QUIESCING	Specifies whether the **MQOPEN** call fails when the queue manager is in quiescing state. Used to control authorization checking. **True** indicates that the **MQOPEN** call will fail if queue manager is quiescing.	**true** or **false**. The configured default is **false.**
MQOO_INPUT_AS_Q_DEF	Specifies whether the queue is opened to browse messages using the queue-defined default. **True** indicates that the queue is open for use with subsequent **MQGET** calls. ***Note:*** The value of this parameter is ignored when the eWay is operating in automatic connection mode, because the eWay must be capable of both receiving and sending messages.	**true** or **false**. The configured default is **true.**
MQOO_INPUT_EXCLUSIVE	Specifies whether the queue is opened to get messages with exclusive access. **True** indicates that the queue is open for use with subsequent **MQGET** calls. Calls will fail with reason code **MQRC_OBJECT_IN_USE** if the queue is currently used (open) by this or another application for input of any type.	**true** or **false**. The configured default is **false.**
MQOO_INPUT_SHARED	Specifies whether the queue is opened to get messages with shared access. **True** indicates that the queue is open for use with subsequent **MQGET** calls. Calls will succeed, even when the queue is currently used (open) by this or another application for input of any type.	**true** or **false**. The configured default is **false.**
MQOO_INQUIRE	Specifies whether the object is opened to inquire attributes. **True** indicates that the queue, name list, process definition, or queue manager is open for use with subsequent **MQINQ** calls.	**true** or **false**. The configured default is **true.**
MQOO_OUTPUT	Specifies whether the object is opened to put messages. **True** indicates that MQOPEN call can succeed, even if the InhibitPut queue attribute is set to **MQQA_PUT_INHIBITED** (though subsequent **MQPUT** calls will fail). ***Note:*** The value of this parameter is ignored when the eWay is operating in automatic connection mode, because the eWay must be capable of both receiving and sending messages.	**true** or **false**. The configured default is **true.**

Table 9 Connectivity Map - Outbound eWay Settings > Queue Open Options (Continued)

Name	Description	Required Value
MQOO_PASS_ALL_CONTEXT	Specifies whether to allow all context to pass. **True** indicates that the **MQPMO_PASS_ALL_CONTEXT** option is specified in the PutMsgOpts parameter when a message is put on a queue, and gives the message identity and origin context information from an input queue opened with the **MQOO_SAVE_ALL_CONTEXT** option. **True** also indicates that **MQOO_PASS_ IDENTITY_CONTEXT** is implied and does not need to be specified.	**true** or **false**. The configured default is **false.**
MQOO_PASS_IDENTITY_ CONTEXT	Specifies whether to allow identity context to pass. **True** indicates that the **MQPMO_PASS_IDENTITY_CONTEXT** option to be specified in the **PutMsgOpts** parameter when a message is put on a queue. This gives the message the identity context information from an input queue opened with the MQOO_SAVE_ALL_CONTEXT option. True indicates that the MQOO_OUTPUT option must be specified.	**true** or **false**. The configured default is **false.**
MQOO_RESOLVE_NAMES	Specifies MQOO_RESOLVE_NAMES. Select **True** if you want to use the **resolved queue manager name** and **resolved queue name attributes** of the **ImqQueue** class.	**true** or **false**. The configured default is **false.**
MQOO_SAVE_ALL_CONTEXT	Specifies whether to save context when message is retrieved. True indicates that context is saved. Context information is associated with this queue handle and set from the context of any message retrieved using this handle.	**true** or **false**. The configured default is **false.**
MQOO_SET	Specifies whether the queue is open to set attributes. True indicates that the queue is open to set attributes and for use with subsequent MQSET calls.	**true** or **False**. The configured default is **False.**

Table 9 Connectivity Map - Outbound eWay Settings > Queue Open Options (Continued)

Name	Description	Required Value
MQOO_SET_ALL_CONTEXT	Specifies whether to allow all context to be set. **True** indicates that all context can be set. The MQPMO_SET_ALL_CONTEXT option is specified in the PutMsgOpts parameter when a message is put on a queue. Gives the identity and origin context information contained in the MsgDesc parameter specified on the MQPUT or MQPUT1 call to the message.	**true** or **false**. The configured default is **false.**
MQOO_SET_IDENTITY_CONTEXT	Specifies whether to allow identity context to be set. **True** indicates that identity context can be set. The MQPMO_SET_IDENTITY_CONTEXT option can be specified in the PutMsgOpts parameter when a message is put on a queue. Gives the identity and origin context information contained in the MsgDesc parameter specified on the MQPUT or MQPUT1 call to the message.	**true** or **false**. The configured default is **false.**

4.6 WebSphere MQ eWay Environment Properties

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

- **Inbound MQSeries eWay > Inbound eWay Environment Configuration** on page 56
- **Outbound MQSeries eWay (XA) > Outbound eWay Environment Configuration** on page 58
- **Outbound MQSeries eWay (XA) Connection Retry Settings** on page 60
- **Outbound MQSeries eWay (XA) > Connection Pool Settings** on page 61
- **Outbound MQSeries eWay > Outbound eWay Environment Configuration** on page 62
- **Outbound MQSeries eWay > Connection Retry Settings** on page 64
- **Outbound MQSeries eWay > Connection Pool Settings** on page 65
- **Outbound MQSeries eWay > Connection Establishment Mode** on page 66

4.6.1 Inbound MQSeries eWay > Inbound eWay Environment Configuration

The **Inbound MQSeries eWay > Inbound eWay Environment Configuration** section of the WebSphere MQ eWay Environment properties contains the top-level parameters displayed in Table 10.

Table 10 Inbound MQSeries eWay > Inbound eWay Environment Configuration

Name	Description	Required Value
Host Name	Specifies name of the computer on which the queue manager resides. This property must be left blank to cause the eWay to use **Bindings mode** rather than **Client mode**. **Bindings mode** allows the eWay to communicate directly with queue manager without a TCP/IP connection. In this mode, the eWay and the queue manager need to be installed on the same machine. When using a **Client mode** connection, the eWay communicates with the queue manager using a TCP/IP-based connection.	The name of the specific queue manager host. Leave the value blank to cause the eWay to use **Bindings** mode.
Port Number	Specifies the number of the listen port on which the queue manager is bound.	A number indicating the port on which the queue manager is bound.
Queue Manager Name	Specifies the name of the local queue manager to which the eWay connects. ***Note:*** *Use only a local queue manager name in the eWay Environment Configuration, whether bindings or Client mode is used. See* **"Accessing Non-Local Queue Managers and Non-Local Queues" on page 67**	The name of the local queue manager.
Channel Name	Specifies the name of the channel being used.	The name of the channel.
Coded Character Set ID	Specifies the **Client Coded Character Set ID** (CCSID). When left blank, the eWay uses a default, platform-dependent CCSID. The eWay must use a Client CCSID compatible with the queue manager's CCSID, in order that character-based data sent to or received from the queue manager is encoded/decoded properly. If, for any reason, it becomes necessary to send character data that utilizes a different CCSID than the one specified by this setting to a queue manager, then you may invoke the eWay OTD's **MsgHeader.setCharacterSet** method from the Collaboration to temporarily override the setting.	A supported CCSID (integer) value, or none at all (blank). For a table of supported CCSID, please consult the entry for the variable, **"MQEnvironment.CCSID"** in IBM document **SC34-6066-00**, **"WebSphere MQ Using Java"**, of your WebSphere MQ software installation.
UserID	Specifies the user ID required to access the queue manager. If none is required, leave this parameter blank.	A User ID required to access the queue manager.

Table 10 Inbound MQSeries eWay > Inbound eWay Environment Configuration (Continued)

Name	Description	Required Value
Password	Specifies the user password required to access the queue manager. If a password is not required, leave this parameter blank.	A user password that grants access to a specific queue manager.
SSL Enabled	When SSL is enabled, all communications are sent over a secure channel.	**Yes** or **No**. The configured default is **No.**

4.6.2 Outbound MQSeries eWay (XA) > Outbound eWay Environment Configuration

The **Outbound MQSeries eWay (XA) > Outbound eWay Environment Configuration** section of the WebSphere MQ eWay Environment properties contains the top-level parameters displayed in Table 11.

Table 11 Outbound MQSeries eWay (XA) > Outbound eWay Environment Configuration

Name	Description	Required Value
Host Name	Specifies name of the computer on which the queue manager resides. This property must be left blank to cause the eWay to use **Bindings mode** rather than **Client mode**. **Bindings mode** allows the eWay to communicate directly with queue manager, without a TCP/IP connection. In this mode, the eWay and the queue manager need to be installed on the same machine. When the eWay is configured to use a **Client mode** connection, the eWay communicates with the queue manager using a TCP/IP-based connection.	The name of the specific queue manager host. Leave the value blank to cause the eWay to use **Bindings** mode. ***Note:*** WebSphere MQ eWay (outbound) support for XA requires Bindings mode. The eWay's HostName and Channel Name property values must be left blank for the eWay to operate in Bindings mode.
Port Number	Specifies the number of the listen port on which the queue manager is bound.	A number indicating the port on which the queue manager is bound.

Table 11 Outbound MQSeries eWay (XA) > Outbound eWay Environment Configuration

Name	Description	Required Value
Queue Manager Name	Specifies the name of the local queue manager to which the eWay connects. ***Note:*** *Use only a local queue manager name in the eWay Environment Configuration, whether bindings or Client mode is used. See* **“Accessing Non-Local Queue Managers and Non-Local Queues” on page 67**	The name of the local queue manager.
Channel Name	Specifies the name of the channel being used.	The name of the channel.
Coded Character Set ID	Specifies the **Client Coded Character Set ID** (CCSID). When left blank, the eWay uses a default, platform-dependent CCSID. The eWay must use a Client CCSID compatible with the queue manager's CCSID, in order that character-based data sent to or received from the queue manager is encoded/decoded properly. If, for any reason, it becomes necessary to send character data that utilizes a different CCSID than the one specified by this setting to a queue manager, then you may invoke the eWay OTD's **MsgHeader.setCharacterSet** method from the Collaboration to temporarily override the setting.	A supported CCSID (integer) value, or none at all (blank). For a table of supported CCSID, please consult the entry for the variable, **“MQEnvironment.CCSID”** in IBM document **SC34-6066-00**, **“WebSphere MQ Using Java”**, of your WebSphere MQ software installation.
User ID	Specifies the user ID required to access the queue manager. If none is required, leave this parameter blank.	A User ID required to access the queue manager.
Password	Specifies the user password required to access the queue manager. If a password is not required, leave this parameter blank.	A user password that grants access to a specific queue manager.
SSL Enabled	When SSL is enabled, all communications are sent over a secure channel.	**Yes** or **No**. The configured default is **No.**

4.6.3 Outbound MQSeries eWay (XA) Connection Retry Settings

The **Outbound MQSeries eWay (XA) > Connection Retry Settings** section of the WebSphere MQ eWay Environment properties provides parameters for retrying outbound eWay connection establishment. This section contains the top-level parameters displayed in Table 12.

Table 12 Environment - Outbound MQSeries eWay (XA) - Connection Retry Settings

Name	Description	Required Value
Connection Retry Count	Specifies the maximum number of attempts made to connect to the destination queue manager. If the queue manager cannot be accessed for any reason, this setting specifies how many reattempts are made to complete the processing.	An integer indicating the maximum number of connection attempts. The configured default is **0**.
Connection Retry Interval	Specifies the amount of time (in milliseconds) between attempts to connect to the destination queue manager or queue. This is the pause between each reattempt to access the destination queue manager. Used in conjunction with the 'Connection Retry Count' setting.	An integer indicating the wait time in milliseconds between connection attempts. The configured default is **1000**.

4.6.4 Outbound MQSeries eWay (XA) > Connection Pool Settings

The **Outbound MQSeries eWay (XA) > Connection Pool Settings** section of the WebSphere MQ eWay Environment properties provides parameters for controlling the outbound eWay's connection pool size. This section contains the top-level parameters displayed in Table 13.

Table 13 Environment - Outbound MQSeries eWay (XA) - Connection Pool Settings

Name	Description	Required Value
Steady Pool Size	Specifies the minimum number of physical connections the pool will keep available at all times. A value of **0** (zero) indicates that there will be no physical connections in the pool and that new connections will be created as needed.	An integer indicating the maximum number of connection kept available. The configured default is **0**.
Max Pool Size	Specifies the maximum number of physical connections the pool can contain. A value of **0** (zero) indicates that there is no maximum.	An integer indicating the maximum pool size. The configured default is **10**.
Max Idle Timeout	Specifies the amount of time, in seconds, before an unused connection is removed from the pool. When this is set to greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** indicates that idle connections can remain in the pool indefinitely. **0** (zero) indicates that there is no maximum.	An integer indicating the idle time in seconds. The configured default is **300**.

4.6.5 Outbound MQSeries eWay > Outbound eWay Environment Configuration

The **Outbound MQSeries eWay > Outbound eWay Environment Configuration** section of the WebSphere MQ eWay Environment properties contains the top-level parameters displayed in Table 14.

Table 14 Environment - Outbound MQSeries eWay > Outbound eWay Environment Configuration

Name	Description	Required Value
Host Name	Specifies name of the computer on which the queue manager resides. This property must be left blank to cause the eWay to use **Bindings mode** rather than **Client mode**. **Bindings mode** allows the eWay to communicate directly with queue manager, without a TCP/IP connection. In this mode, the eWay and the queue manager need to be installed on the same machine. When the eWay is configured to use a **Client mode** connection, the eWay communicates with the queue manager using a TCP/IP-based connection.	The name of the specific queue manager host. Leave the value blank to cause the eWay to use **Bindings** mode. ***Note:*** WebSphere MQ eWay (outbound) support for XA requires Bindings mode. The eWay's HostName and Channel Name property values must be left blank for the eWay to operate in Bindings mode.
Port Number	Specifies the number of the listen port on which the queue manager is bound.	A number indicating the port on which the queue manager is bound.
Queue Manager Name	Specifies the name of the queue manager to which the eWay connects. ***Note:*** *Use only a local queue manager name in the eWay Environment Configuration, whether bindings or Client mode is used. See* **"Accessing Non-Local Queue Managers and Non-Local Queues" on page 67**	The name of the local queue manager.
Channel Name	Specifies the name of the channel being used.	The name of the channel.

Table 14 Environment - Outbound MQSeries eWay > Outbound eWay Environment Configuration (Continued)

Name	Description	Required Value
Coded Character Set ID	Specifies the **Client Coded Character Set ID** (CCSID). When left blank, the eWay uses a default, platform-dependent CCSID. The eWay must use a Client CCSID compatible with the queue manager's CCSID, in order that character-based data sent to or received from the queue manager is encoded/decoded properly. If, for any reason, it becomes necessary to send character data that utilizes a different CCSID than the one specified by this setting to a queue manager, then you may invoke the eWay OTD's **MsgHeader.setCharacterSet** method from the Collaboration to temporarily override the setting.	A supported CCSID (integer) value, or none at all (blank). For a table of supported CCSID, please consult the entry for the variable, **"MQEnvironment.CCSID"** in IBM document **SC34-6066-00**, **"WebSphere MQ Using Java"**, of your WebSphere MQ software installation.
UserID	Specifies the user ID required to access the queue manager. If none is required, leave this parameter blank.	A User ID required to access the queue manager.
Password	Specifies the user password required to access the queue manager. If a password is not required, leave this parameter blank.	A user password that grants access to a specific queue manager.
SSL Enabled	When SSL is enabled, all communications are sent over a secure channel.	**Yes** or **No**. The configured default is **No.**

4.6.6 Outbound MQSeries eWay > Connection Retry Settings

The **Outbound MQSeries eWay > Connection Retry Settings** section of the WebSphere MQ eWay Environment properties provides parameters for retrying outbound eWay connection establishment. This section contains the top-level parameters displayed in Table 15.

Table 15 Environment - Outbound MQSeries eWay - Connection Retry Settings

Name	Description	Required Value
Connection Retry Count	Specifies the maximum number of attempts made to connect to the destination queue manager. If the queue manager cannot be accessed for any reason, this setting specifies how many reattempts are made to complete the processing.	An integer indicating the maximum number of connection attempts. The configured default is **0**.
Connection Retry Interval	Specifies the amount of time (in milliseconds) between attempts to connect to the destination queue manager. This is the pause between each reattempt to access the destination queue manager or queue. Used in conjunction with the 'Connection Retry Count' setting.	An integer indicating the wait time in milliseconds between connection attempts. The configured default is **1000**.

4.6.7 Outbound MQSeries eWay > Connection Pool Settings

The **Outbound MQSeries eWay > Connection Pool Settings** section of the WebSphere MQ eWay Environment properties provides parameters for controlling the outbound eWay's connection pool size. This section contains the top-level parameters displayed in Table 16.

Table 16 Environment - Outbound MQSeries eWay - Connection Pool Settings

Name	Description	Required Value
Steady Pool Size	Specifies the minimum number of physical connections the pool will keep available at all times. A value of **0** (zero) indicates that there will be no physical connections in the pool and that new connections will be created as needed.	An integer indicating the maximum number of connection kept available. The configured default is **0**.
Max Pool Size	Specifies the maximum number of physical connections the pool can contain. A value of **0** (zero) indicates that there is no maximum.	An integer indicating the maximum pool size. The configured default is **10**.
Max Idle Timeout	Specifies the amount of time, in seconds, before an unused connection is removed from the pool. When this is set to greater than **0**, the container removes or destroys any connections that are idle for the specified duration. A value of **0** indicates that idle connections can remain in the pool indefinitely. **0** (zero) indicates that there is no maximum.	An integer indicating the idle time in seconds. The configured default is **300**.

4.6.8 Outbound MQSeries eWay > Connection Establishment Mode

The **Outbound MQSeries eWay > Connection Establishment Mode** section of the WebSphere MQ eWay Environment properties contains the top-level parameters displayed in Table 17.

Table 17 Environment - Outbound MQSeries eWay - Connection Establishment Mode

Name	Description	Required Value
Connection Mode	Specifies whether the eWay automatically connects to the external system upon startup or connects using manual mode. When set to **Manual**, the eWay will not connect to the external system on startup, and instead expects the user to initiate the connection by invoking the MQ eWay OTD's **connectToQueueManager** method. Manual mode is only available when using Java Collaboration Definitions. This allows you to dynamically connect to different Queue Managers. Any parameters assigned in the Java Collaboration will override the same parameters specified in the Connectivity Map or Environment properties.	**Automatic** or **Manual**. The configured default is Automatic.

4.7 Polling and Reconnection Logic

The WebSphere MQ eWay runs in two modes: Polling and Non-Polling.

Polling Mode

The eWay in its inbound capacity, runs in polling mode. If the eWay looses its connection, error messages and monitor alerts are posted and the eWay automatically attempts to reconnect until a connection is reestablished. Once it reconnects, polling resumes the retrieval of available messages.

Non-Poll Mode

An eWay performing a put or a get operation, runs in non-poll mode. If the eWay looses its connection, it informs the Collaboration that the connection is down by throwing an exception. It is the Collaboration author's responsibility to add business logic to the Collaboration to catch the exception and preserve the data (for example, saving undelivered messages to a safe location or invoking **backout()** on the eWay's Message OTD).

4.8 Accessing Non-Local Queue Managers and Non-Local Queues

When used with alias queues and remote queues, the WebSphere MQ eWay functions with several restrictions.

Alias queues and remote queues with local queue definitions may be accessed in the same way as actual local queues, through the use of the eWay OTD's **accessQueue(String)** method. Remote queues without local queue definitions need to use the **accessQueue(String, String)** method instead.

Also, when alias queues or remote queues are used, the eWay cannot proactively verify the connection (and reconnect, if necessary) before each OTD operation. This is because the eWay verifies connections by querying queue objects, and it is not possible to query alias queues and remote queues. This means that when alias queues or remote queues are used with the eWay, the Collaboration is responsible for recovering connection failures itself, including reestablishing the queue manager and queue connections as needed.

For more information, refer to the *WebSphere MQ eWay Javadoc*.

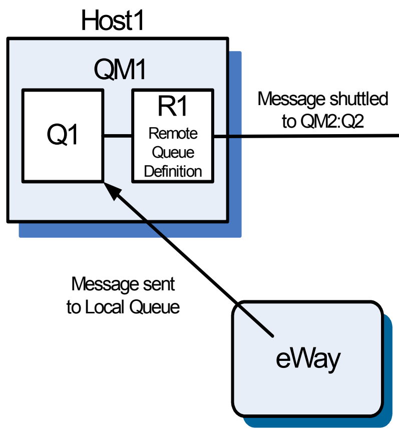
4.8.1 Connecting to a Remote WebSphere MQ Queue.

When an eWay connects to a local queue manager and accesses one of its queues, that queue is a local queue. When an eWay connects to a remote queue manager and accesses one of its queues, then that queue, is also a local queue. In WebSphere MQ terms, a remote queue is a queue that is managed by a queue manager other than the one to which the application (in this case, the eWay) is connected.

For example, say that there are two queue managers, **QM1** and **QM2**. **QM1** manages a queue (**Q1**) and runs on **Host1**. **QM2** manages a queue (**Q2**) and runs on **Host2**.

Furthermore, say that need to send messages to **Q2**, but the eWay may only communicate with **Host1** (that is, Host2 is unreachable from the system in which the eWay is executing). By creating the appropriate channels and a remote queue definition (**R1** on **QM1**), messages sent to **R1** can be shuttled automatically to **Q2** on **QM2**.

Figure 8 Connecting to a Remote Queue



For this example, the Queues and the eWay are configured as follows:

1 If either **QM1** or **QM2** do not have a transmission queue defined, create one. Both queue managers require one transmission queue each. In this example, assume that both queue managers have the transmission queue '**xmit**'.

2 Create a Sender Channel for **QM1** that points to **Host2** and transmission queue **xmit**. The name of the channel must match the Receiver Channel created in the next step.

3 Create a Receiver Channel for **Q2**. The name of the channel must match the Sender Channel created in the previous step.

4 In **QM1**, create a Remote Queue Definition (**R1)**. Designate **Q2** as its remote queue, **QM2** as its remote queue manager, and **xmit** as its transmission queue.

5 Configure the MQ eWay to connect to **Host1**, **QM1**, and have it put messages into queue **R1**.

Note that messages cannot be read/GET from remote queues, only PUT. In the example situation above, to read the messages placed in **QM2**:**Q2** via **R1**, an eWay needs to connect directly to **QM2** (**Host2**), thereby interacting with **Q2** as a local queue.

Chapter 5

Using the WebSphere MQ eWay With eInsight

This chapter describes how to use the WebSphere MQ eWay with the Sun Java Composite Application Platform Suite's eInsight Business Process Manager and its engine's Web Services interface.

Note: *You must have the* **eInsight.sar** *file installed to use the Web Services interface.*

What's in This Chapter

5.1 eInsight Engine and Components

eGate components can be deployed as Activities in eInsight Business Processes. Using the Enterprise Designer with eInsight, you can add an Activity to a Business Process, then associate that Activity with an eGate component, such as an eWay. Then, when eInsight runs the Business Process, it automatically invokes that component using its Web Services interface. eGate components that can interface with eInsight in this way include:

- Object Type Definitions (OTDs)
- eWays
- Collaborations

See the *eInsight Business Process Manager User's Guide* for details.

5.2 The WebSphere MQ eWay With eInsight

An eInsight Business Process Activity can be associated with the WebSphere MQ eWay during the system design phase. To make this association, select the desired operators under the eWay in the Enterprise Explorer and drag it onto the eInsight Business Process Designer canvas.

The WebSphere MQ eWay has the following operators available:

- **receive**
- **mqget**: Obtains the next message from a queue.
- **mqput**: Sends a message to a queue.
- **mqgetbulk**: Obtains all the messages from a queue or you can specify the number of messages retrieved each time.
- **mqputbulk**: Sends all messages to a queue.

The operation is automatically changed to an Activity with an icon identifying the component that is the basis for the Activity. At run time, eInsight invokes each step in the order defined in the Business Process. Using eInsight's Web Services interface, the Activity in turn invokes the WebSphere MQ eWay.

5.3 Importing a Sample Project

To import a sample eWay Project to the Enterprise Designer do the following:

1. The sample files are uploaded with the eWay's documentation SAR file and downloaded from the Sun Java Composite Application Platform Suite Installer's Documentation tab. The **WebSphere_MQ_eWay_Sample.zip** file contains the various sample Project zip files. Extract the samples to a local file.
2. Save all unsaved work before importing a Project.
3. From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import** from the shortcut menu. The **Import Manager** appears.
4. Browse to the directory that contains the sample Project zip file. Select the sample file (for this sample, **prjWebSphereMQ_Sample_BPEL.zip**) and click **Import**. After the sample Project is successfully imported, click **Close**.
5. Before an imported sample Project can be run you must do the following:
   - Create an Environment (see **"Creating an Environment" on page 81**)
   - Configure the eWays for your specific system (see **"Configuring the eWay Properties" on page 81**)
   - Create a Deployment Profile (see **"Creating and Activating the Deployment Profile" on page 84**)

- Create and start a domain (see **"Creating and Starting the Domain" on page 86**)
- Build and deploy the Project (see **"Running the Sample" on page 86**)

5.4 WebSphere MQ eWay eInsight Sample Project

A sample Project for the WebSphere MQ eWay using eInsight Business Processes, **prjWebSphereMQ_Sample_BPEL.zip**, is included with the documentation SAR installation.

The **prjWebSphereMQ_Sample_BPEL** Project demonstrates the following:

Get:

1. The File eWay receives a message from an external directory and triggers the Business Process.
2. The WebSphere MQ eWay, in get mode, retrieves a message from an MQ Queue. The message is published to an outbound File eWay.
3. The File eWay publishes the message to an external directory.

Put:

1. The File eWay receives a message from an external directory and triggers the Business Process.
2. The WebSphere MQ eWay, in put mode, sends a message to an MQ Queue. The content of the input file is applied to the message.

Receive:

1. The WebSphere MQ eWay, in receive mode, retrieves a message from an MQ Queue. The message is published to an outbound File eWay.
2. The File eWay publishes the message to an external directory.

5.5 Create the WebSphere MQ Queue

The first step in creating the sample Project is to install and configure **IBM's WebSphere MQ Server** and **MQ queue manager** on the local host.

It is assumed that the reader is experienced in the use of the WebSphere MQ queue manager. For the sample implementation do the following:

1. Open IBM WebSphere MQ Explorer.
2. Create a new queue manager.
3. From the WebSphere MQ queue manager create a new queue.

IBM WebSphere MQ Server and Queue Manager Limits and Settings

- When using the WebSphere MQ queue manager on UNIX, you must be a member of the mqm group to create and start the MQ queue manager.
- It is essential that the WebSphere MQ Administrator regularly monitor the number of messages in the queue. Message expiration settings should be set to allow for extended storage.

WebSphere MQ is limited in the number of messages that can be sent before a commit is executed, and the number of physical messages that can exist on the queue at any one time. This can result in exception errors when upper limits for these numbers are exceeded. Memory and performance of the specific server may also effect the results.

5.6 Creating the prjWebSphereMQ_Sample_BPEL Project

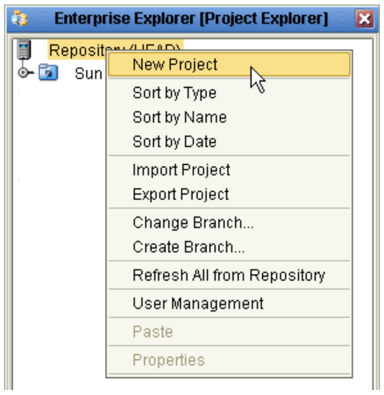
The following pages provide step by step directions for manually creating the **prjWebSphereMQ_Sample_BPEL** Project.

5.6.1 Creating a Project

The next step is to create a new Project using the Enterprise Designer.

1. Start the Enterprise Designer.
2. From the Enterprise Explorer's Project Explorer tab, right-click the Repository and select **New Project** (see Figure 9). A new Project (Project1) appears on the Project Explorer tree.

Figure 9 Enterprise Explorer - New Project



3. Rename the Project (for this sample, **prjWebSphereMQ_Sample_BPEL**).

5.6.2 Creating the Business Processes

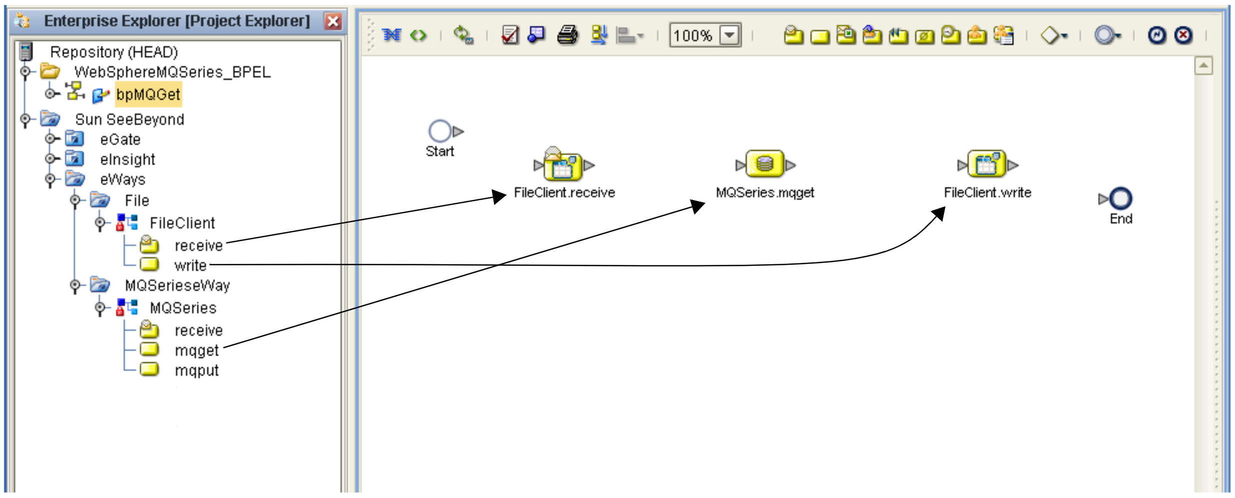
An eInsight Business Process is a collection of activities that uses universally supported process-related language (BPEL) to create business interaction protocols. The prjWebSphereMQ_Sample_BPEL Project includes three Business Processes:

- bpMQGet
- bpMQPut
- bpMQReceive

Creating the MQGet Business Process

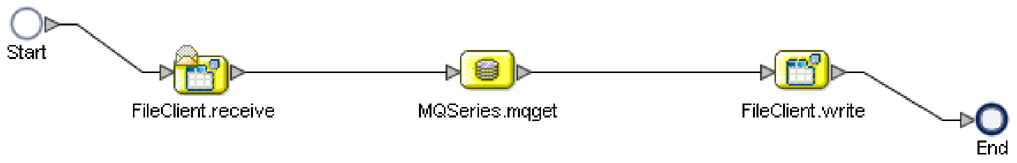
1. From the Enterprise Designer's Project Explorer tree, right-click **prjWebSphereMQ_Sample_BPEL**, and select **New** > **Business Process** from the shortcut menu. The eInsight Business Process Designer appears and **BusinessProcess1** is added to the Project Explorer tree. Rename the Business Process to **bpMQGet**.
2. From the Project Explorer tree, expand the **Sun SeeBeyond > eWays > MQSerieseWay > MQSeries**, and **Sun SeeBeyond > eWays > File > FileClient** nodes to expose the available Business Process elements.
3. Populate the eInsight Business Process Designer's canvas with the following elements from the Project Explorer tree, as displayed in Figure 10:
   - **receive**, under **Sun** SeeBeyond > eWays > File > FileClient
   - **mqget**, under **Sun** SeeBeyond > eWays > MQSerieseWay > MQSeries
   - **write**, under **Sun** SeeBeyond > eWays > File > FileClient

Figure 10 eInsight Business Process Designer - bpMQGet - Populate the Canvas



4. Link the modeling elements by clicking on the element's connector and dragging the cursor to the next element's connector, making the following links as displayed in Figure 11 on page 74.
   - **Start** to **FileClient.receive**
   - **FileClient.receive** to **MQSeries.mqget**
   - **MQSeries.mqget** to **FileClient.write**
   - **FileClient.write** to **End**

Figure 11 eInsight Business Process Designer - Link the Modeling Elements

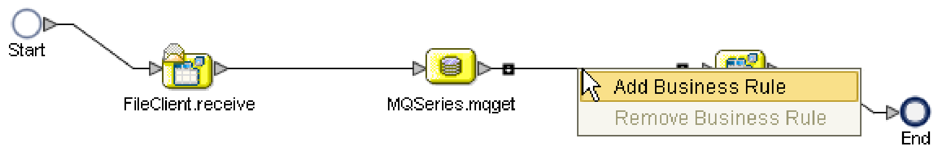


Configuring the bpMQGet Modeling Elements

Business Rules, created between the Business Process elements, define the relationship between the input and output Attributes of the elements.

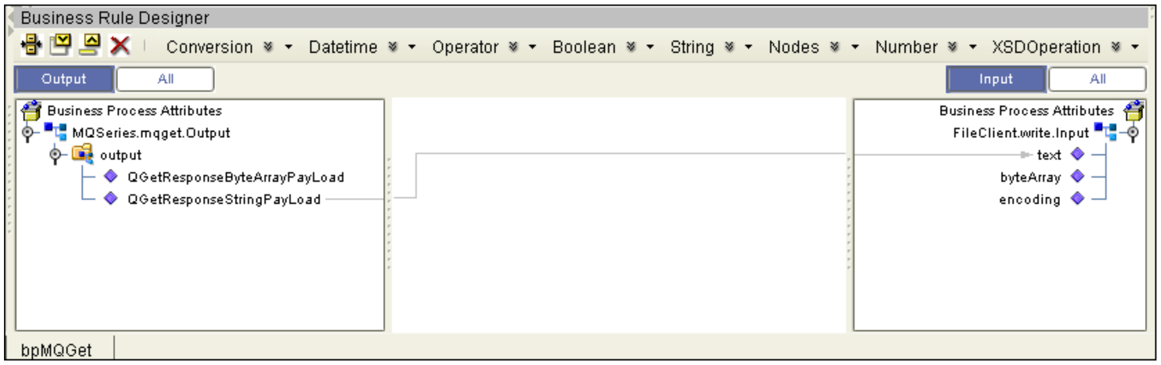
1. Right-click the link between the **MQSeries.mqget** and **FileClient.write** Activities and select **Add Business Rules** from the shortcut menu as displayed in Figure 12.

Figure 12 eInsight Business Process Designer - Adding Business Rules



2. From the eInsight Business Process Designer toolbar, click the **Display Business Rule Designer** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.
3. Click on the **Business Rules** icon in the link between **MQSeries.mqget** and **FileClient.write** to display the Business Rule Output and Input Attributes in the Business Rule Designer. These Attributes can now be modified.
4. Map **QGetResponseStringPayLoad**, under **MQSeries.mqget.Output > output** in the Output pane of the Business Rule Designer, to **text**, under **FileClient.write.Input** in the Input pane of the Business Rule Designer. To do this, click on **QGetResponseStringPayLoad** and drag your cursor to the **text** node. A link is created between the two elements (see Figure 13).

Figure 13 eInsight Business Rule Designer

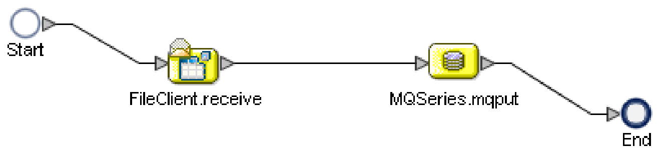


5. From the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process Code** icon to synchronize the graphical interface to the Business Process code.

Creating the bpMQPut Business Process

1. Right-click the 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 **bpMQPut**.
2. Populate the eInsight Business Process Designer's modeling canvas with the following activities from the Project Explorer tree as displayed in **Figure 14 on page 75**:
   - **receive**, under **Sun SeeBeyond > eWays > File > FileClient**
   - **mqput**, under **Sun SeeBeyond > eWays > MQSerieseWay > MQSeries**
3. Link the modeling elements by clicking on the element connector and dragging the cursor to the next element connector, making the following links as displayed in Figure 14:
   - **Start** to **FileClient.receive**
   - **FileClient.receive** to **MQSeries.mqput**
   - **MQSeries.mqput** to **End**

Figure 14 Business Process Designer - bpMQPut



Configuring the bpMQPut Modeling Elements

To create the **bpMQPut** Business Rules do the following:

1. Right-click the link between the **FileClient.receive** and **MQSeries.mqput** Activities and select **Add Business Rule** from the shortcut menu.
2. From the eInsight Business Process Designer toolbar, click the **Display Business Rule Designer** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.
3. Click on the Business Rule icon in the link between **FileClient.receive** and **MQSeries.mqput** to display the Business Rule's Input and Output Attributes in the Business Rule Designer. These Attributes can now be modified.
4. Map **text**, under **FileClient.receive.Output** in the Output pane of the Business Rule Designer, to **QPutRequestStringPayLoad** under **MQSeries.mqput.Input > input** in the Input pane of the Business Rule Designer.
5. From the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process** icon to synchronize the graphical interface to the Business Process code.

Creating the bpMQReceive Business Process

1. Right-click the 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 **bpMQReceive**.
2. Populate the eInsight Business Process Designer's modeling canvas with the following activities from the Project Explorer tree:
   - **receive**, under **Sun SeeBeyond > eWays > MQSerieseWay > MQSeries**
   - **write**, under **Sun SeeBeyond > eWays > File > FileClient**
3. Link the modeling elements by clicking on the element connector and dragging the cursor to the next element connector, making the following links:
   - **Start** to **MQSeries.receive**
   - **MQSeries.receive** to **FileClient.write**
   - **FileClient.write** to **End**

Configuring the bpMQReceive Modeling Elements

To create the **bpMQReceive** Business Rules do the following:

1. Add a Business Rule between the **MQSeries.receive** and **FileClient.write** Activities.
2. From the eInsight Business Process Designer toolbar, click the **Display Business Rule Designer** button. The Business Rule Designer appears at the bottom of the eInsight Business Process Designer.
3. Click on the Business Rule icon in the link between **MQSeries.receive** and **FileClient.write** to display the Business Rule's Input and Output Attributes in the Business Rule Designer. These Attributes can now be modified.
4. Map **ByteArrayData**, under **MQSeries.receive.Output > MQAppconnMessage > MsgBody** in the Output pane of the Business Rule Designer, to **byteArray** under **FileClient.write.Input** in the Input pane of the Business Rule Designer.
5. From the Business Process Designer toolbar, click the **Synchronize Graphical Model and Business Process** icon to synchronize the graphical interface to the Business Process code.
6. Save all of your current changes to the Repository.

5.6.3 Creating the Connectivity Maps

Connectivity Maps provide a canvas for assembling and configuring a Project's components. The prjWebSphereMQ_Sample_BPEL Project uses three Connectivity Maps:

- cmMQGet
- cmMQPut
- cmMQReceive

To create the prjWebSphereMQ_Sample_BPEL Project's Connectivity Maps, do the following:

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

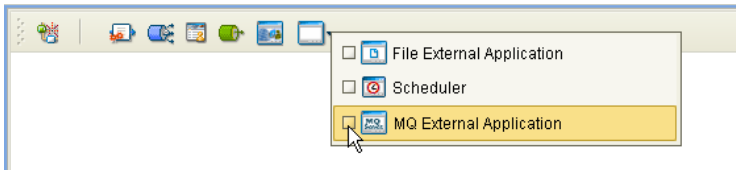
2 The New Connectivity Map appears and a node for the Connectivity Map is added to the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map **cmMQGet**.

3 Create two additional Connectivity Maps named **cmMQPut** and **cmMQReceive**.

Selecting the External Applications

In the Connectivity Maps, eWays are associated with External Systems. For example, to establish a connection to a WebSphere MQ application, you must first select WebSphere MQ (MQ) as an External Application (see Figure 15).

Figure 15 Connectivity Map - External Applications



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

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

Populating the Connectivity Maps

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas. You can add the Project components by dragging the icons from the toolbar to the canvas.

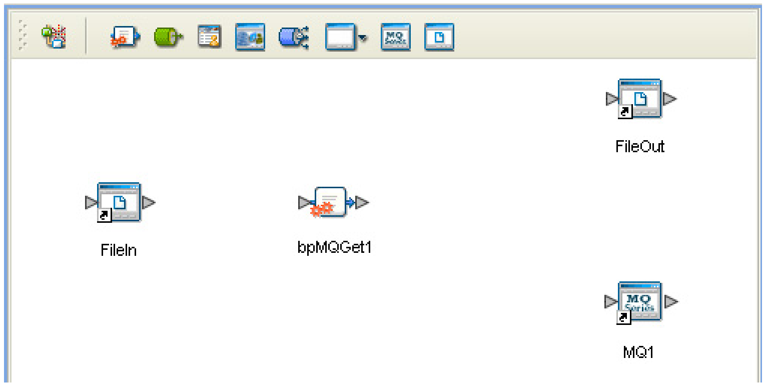
1 To populate the cmMQGet Connectivity Map, drag the following components onto the canvas as displayed in Figure 16.

- File External System (2)
- MQ External System
- Service (A service is a container for Java Collaborations, Business Processes, eTL processes, and so forth)

2 Rename components with the following names (as displayed in Figure 16):

- File1 (File External Application on the left) to **FileIn**
- File2 to **FileOut**
- cmMQGet_Service1 to **bpMQGet1**

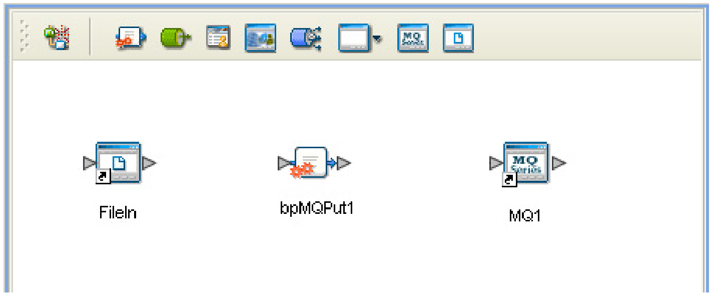
Figure 16 cmMQGet Connectivity Map



3 Open the **cmMQPut** Connectivity and populate the canvas with the following components (as displayed in Figure 17):

- File External System (rename to **FileIn**)
- MQ External System
- Service (rename to **bpMQPut1**)

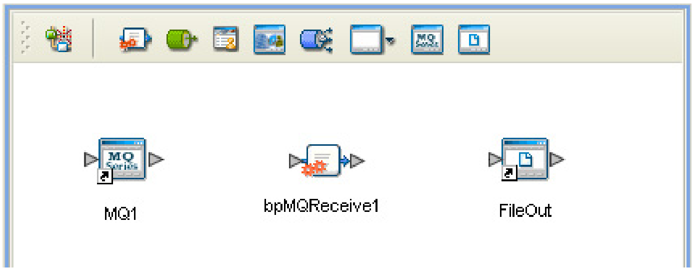
Figure 17 cmMQPut Connectivity Map



4 Populate the **cmMQReceive** Connectivity Map with the following components (as displayed in Figure 18 on page 78):

- File External System (rename to **FileOut**)
- Service (rename to **bpMQReceive1**)
- MQ External System

Figure 18 cmMQReceive Connectivity Map



5 Save your current changes to the Repository.

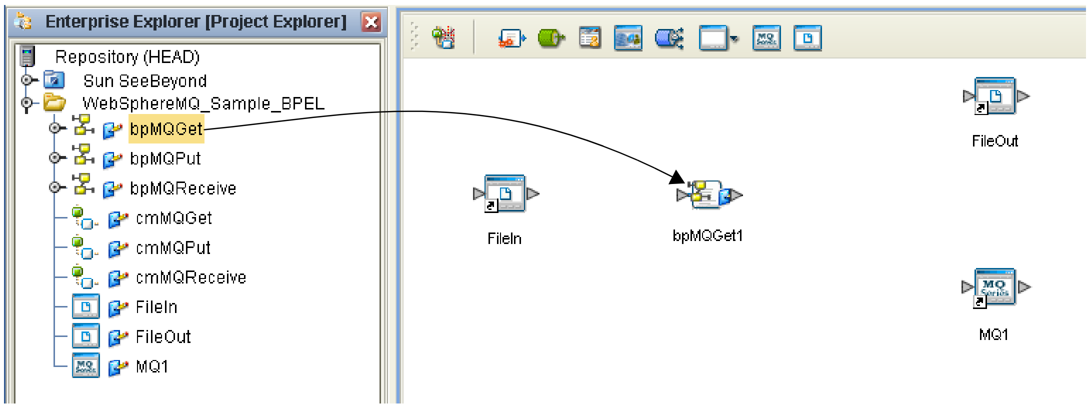
5.6.4 Binding the eWay Components

After the Business Process and Connectivity Maps have been created, the components can be associated and bindings created in the Connectivity Maps.

1 From the Project Explorer, double-click **cmMQGet** to display the **cmMQGet** Connectivity Map.

2 Drag and drop the **bpMQGet** Business Process from the Project Explorer to the **bpMQGet1** Service. If the Business Process was successfully associated, the Service icon changes to a Business Process (see Figure 19).

Figure 19 cmMQGet Connectivity Map - Binding the Components



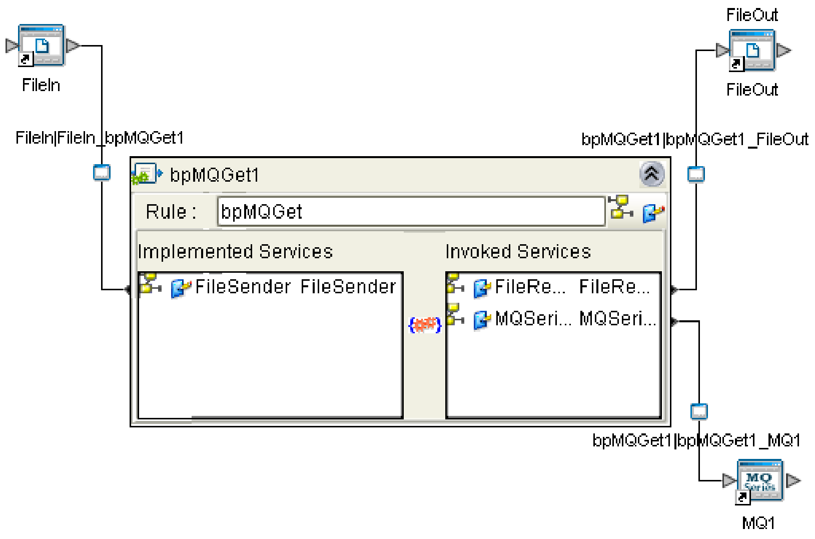
3 From the Connectivity Map canvas, double-click **bpMQGet1**. The **bpMQGet1** binding dialog box appears using the **bpMQGet** Rule.

4 From the **bpMQGet1** binding dialog box, map **FileSender** (under Implemented Services) to the output node of the inbound **FileIn** External Application. To do this, click on **FileSender** under Implemented Services in the **bpMQGet1** binding box, and drag your cursor to the output node of the **FileIn** External Application. A link now joins the two components.

5 From the **bpMQGet1** binding dialog box, map **FileReceiver** (under Invoked Services) to the input node of the **FileOut** External Application.

6 From the **bpMQGet1** binding dialog box, map **MQSeriesWebService** (under Invoked Services) to the input node of the **MQ1** External Application. (see **Figure 20 on page 80**).

Figure 20 cmMQGet Connectivity Map - Binding the Components



7 Minimize the **bpMQGet1** binding dialog box.

8 From the Project Explorer, double-click **cmMQPut** to display the **cmMQPut** Connectivity Map.

9 Drag and drop the **bpMQPut** Business Process from the Project Explorer to the **bpMQPut1** Service.

10 From the Connectivity Map, double-click the **bpMQPut1** Service. The **bpMQPut1** binding dialog box appears with the **bpMQPut** Rule.

11 From the **bpMQPut1** binding dialog box, map **FileSender** (under Implemented Services) to the output node of the **FileIn** External Application.

12 From the **bpMQPut1** binding dialog box, map **MQSeriesWebService** (under Invoked Services) to the input node of the **MQ1** External Application.

13 Minimize the **bpMQPut1** binding dialog box.

14 From the Project Explorer, double-click **cmMQReceive** to display the **cmMQReceive** Connectivity Map.

15 Drag and drop the **bpMQReceive** Business Process from the Project Explorer to the **bpMQReceive1** Service.

16 From the Connectivity Map, double-click the **bpMQReceive1** Service. The **bpMQReceive1** binding dialog box appears with the **bpMQReceive** Rule.

17 From the **bpMQReceive1** binding dialog box, map **MQListener** (under Implemented Services) to the output node of the **MQ1** External Application.

18 From the **bpMQReceive1** binding dialog box, map **FileReceiver** (under Invoked Services) to the input node of the outbound **FileOut** External Application.

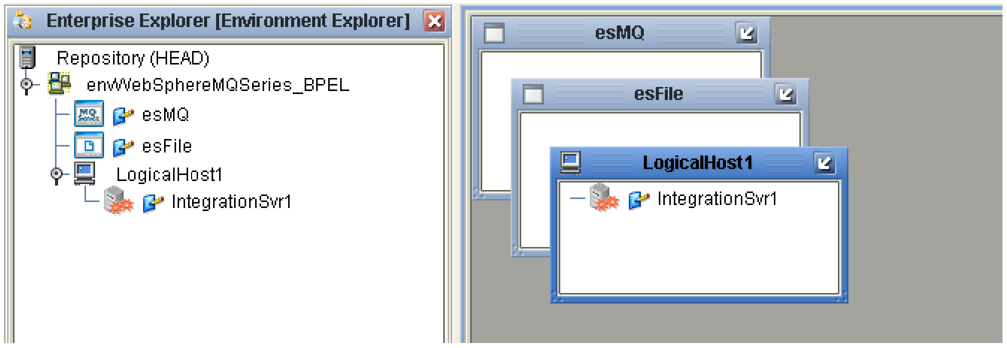
19 Minimize the **bpMQReceive1** binding dialog box, and save all your current changes to the Repository.

5.6.5 Creating an Environment

Environments include the external systems, Logical Hosts, integration servers and message servers used by a Project and contain the configuration information for these components. Environments are created using the Enterprise Designer's Environment Explorer and 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 **envWebSphereMQSeries_BPEL**.
4. Right-click **envWebSphereMQSeries_BPEL** and select **New > MQ External System**. Name the External System **esMQ** and click **OK**. **esMQ** is added to the Environment Editor.
5. Right-click **envWebSphereMQSeries_BPEL** and select **New > File External System**. Name the External System **esFile** and click **OK**. **esFile** is added to the Environment Editor.
6. Right-click **envWebSphereMQSeries_BPEL** and select **New > Logical Host**. **LogicalHost1** is added to the Environment Editor.
7. From the Environment Explorer tree, right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1.
8. Save changes to the repository. The Environment Explorer and Environment Editor now appear as displayed in Figure 21.

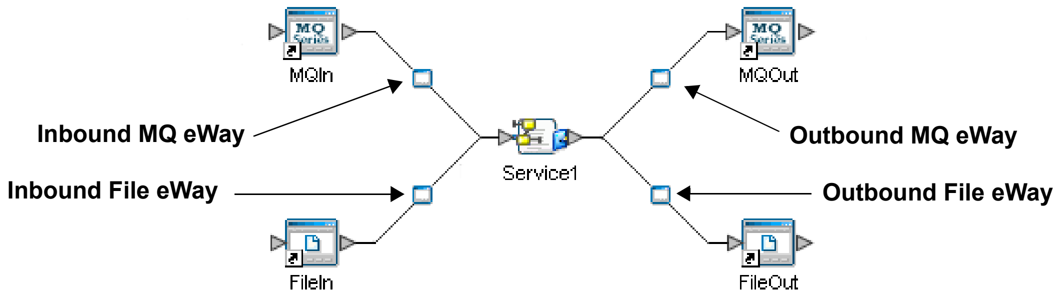
Figure 21 Environment Editor



5.6.6 Configuring the eWay Properties

The **prjWebSphereMQ_Sample_BPEL** Project contains WebSphere MQ eWays and File eWays, each represented in the Connectivity Map as a node between an External Application and a Service. eWays facilitate communication and movement of data between the External Applications and the eGate system (see **Figure 22 on page 82**).

Figure 22 eWays



The eWay properties are configured from both the Connectivity Map and the Environment. To configure the eWays do the following:

Configuring the File eWay Properties

1. From the cmMQGet Connectivity Map, double-click the inbound **FileIn** eWay (see Figure 22). The **Properties Editor** opens to the inbound File eWay properties.
2. Modify the properties for your system, including the settings for the inbound File eWay in Table 18, and click **OK**.

Table 18 cmMQGet - FileIn eWay Settings

Inbound File eWay Connection Parameters	
Input file name	filetrigger_mq.txt

3. In the same way, modify the properties of the cmMQPut Connectivity Map's inbound **FileIn** eWay, entering **filein_mq.txt** as the Input file name property value.
4. From the cmMQGet and cmMQReceive Connectivity Maps, modify the outbound **FileOut** eWay properties for your system, including the settings in Table 19.

Table 19 Outbound File eWay Settings

Outbound File eWay Connection Parameters	
Output file name	fileout_mq.dat

5. From the **Environment Explorer** tree, right-click the File eWay External System (**esFile** in this sample), and select **Properties** from the shortcut menu. The Properties Editor appears.
6. Modify the File eWay Environment properties for your system, including the settings in Table 20, and click **OK**.

Table 20 File eWay Environment Properties

File eWay Environment Properties
Inbound File eWay > Parameter Settings Set as directed, otherwise use the default settings

File eWay Environment Properties	
Directory	*Select a directory, for example C:/temp*
Outbound File eWay > Parameter Settings Set as directed, otherwise use the default settings	
Directory	*Select a directory, for example C:/temp*

Configuring the WebSphere MQ eWay Properties

The WebSphere MQ eWay properties must be set in both the Project Explorer and Environment Explorer. For more information on the WebSphere MQ eWay properties and the Properties Editor, see **"Creating and Configuring the WebSphere MQ eWay" on page 34** or see the *Sun SeeBeyond eGate Integrator User's Guide*.

Modifying the WebSphere MQ eWay Connectivity Map Properties

1. From the cmMQGet Connectivity Map, double-click the MQ eWay. The Properties Editor opens to the outbound WebSphere MQ eWay Connectivity Map properties.
2. Modify the inbound MQ eWay properties for your system and click **OK**.
3. In the same way, open the cmMQPut Connectivity Map and modify the outbound MQ eWay properties for your system.
4. In the same way, open the cmMQReceive Connectivity Map and modify the inbound MQ eWay properties for your system.

Modifying the WebSphere MQ eWay Environment Explorer Properties

1. From the **Environment Explorer** tree, right-click the WebSphere MQ External System (**esMQ** in this sample), and select **Properties**. The Properties Editor opens to the WebSphere MQ eWay Environment properties.
2. Modify the WebSphere MQ eWay Environment properties for your system, including the settings in Table 21, and click **OK**.

Table 21 WebSphere MQ Environment Explorer eWay Settings

MQ eWay Environment Explorer Properties	
Inbound MQSeries eWay - Inbound eWay Environment Configuration Set as directed, otherwise use the default settings.	
MQ Host Name	*The name of the server where the specific queue manager runs.*
Port Number	*Port number on which the queue manager is bound. The default is 1414.*
Queue Manager Name	*The name of the WebSphere MQ queue manager*
User ID	User ID required by queue manager (optional)
Password	Password required by queue manager
SSL ENabled	When SSL is enabled, all communications are sent over a secure channel

5.6.7 Configuring the Integration Server

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

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

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

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

5.6.8 Creating and Activating the Deployment Profile

Deployment Profiles are specific instances of a Project in a particular Environment. A Deployment Profile contains information regarding the assignment of services and message destinations to integration and message servers (JMS IQ Managers). It also contains version information for all versionable objects in the Project. Deployment Profiles are created using the Deployment Editor.

The prjWebSphereMQ_Sample_BPEL Project performs multiple operations and requires two different Deployment Profiles; **dp_bpGetPut**, and **dp_bpMQReceive**. This allows part of the Project to be undeployed.

Create the dp_bpGetPut Deployment Project

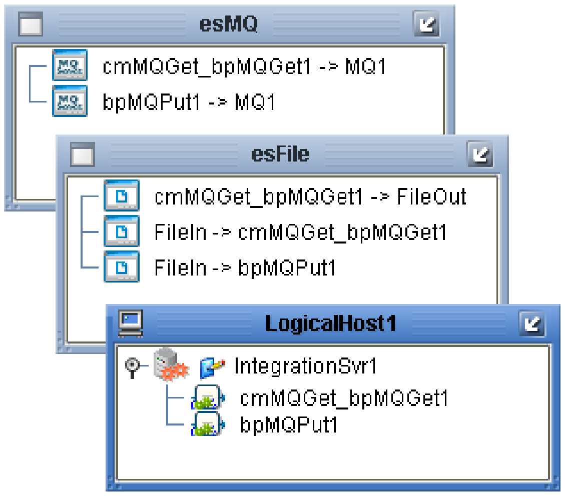
To create the dp_bpGetPut Deployment Profile do the following:

1 From the Project Explorer, right-click the **prjWebSphereMQ_Sample_BPEL** Project and select **New** > **Deployment Profile** from the shortcut menu. The **Create Deployment Profile for prjWebSphereMQ_Sample_BPEL** dialog box appears.

2 Enter **dp_bpMQGetPut** as the name for the Deployment Profile. Select **envWebSphereMQSeries_BPEL** as the Environment, and make sure that only the **cmMQGet** and **cmMQPut** Connectivity Maps are selected (checked). Click **OK**. The Deployment Editor appears.

3 From the Deployment Editor, click **Automap**. This automatically maps all of your Project components to the correct External Systems and Integration Server. The **Automap Results** dialog box appears. Click **Close**. The **dp_bpMQGetPut** Deployment Profile now contains the components displayed in **Figure 23 on page 85**.

Figure 23 dp_bpMQGetPut Deployment Profile



Create the dp_bpMQReceive Deployment Project

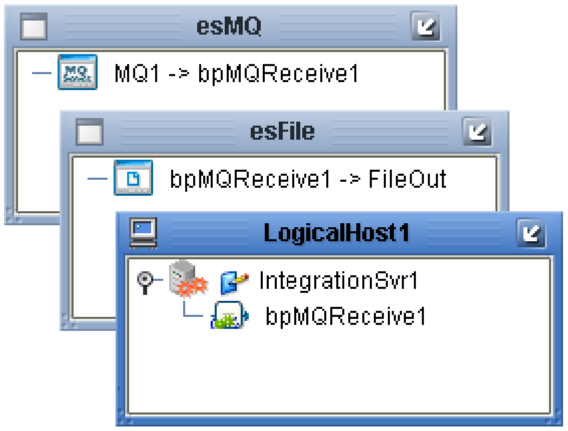
To create the dp_bpMQReceive Deployment Profile do the following:

1 From the Project Explorer, right-click the **prjWebSphereMQ_Sample_BPEL** Project and select **New** > **Deployment Profile** from the shortcut menu. The **Create Deployment Profile for prjWebSphereMQ_Sample_BPEL** dialog box appears.

2 Enter **dp_bpMQReceive** as the name for the Deployment Profile. Select **envWebSphereMQSeries_BPEL** as the Environment, and make sure that only the **cmMQReceive** Connectivity Map is selected (checked). Click **OK**.

3 From the Deployment Editor, click **Automap**. Review and close the **Automap Results** dialog box. The **dp_bpMQReceive** Deployment Profile now contains the components displayed in Figure 23.

Figure 24 dp_bpMQReceive Deployment Profile



4 Save your current changes to the Repository.

5.6.9 Creating and Starting the Domain

To deploy your Project, you must first create a domain. After the domain is created, the Project is built and then deployed.

Create and Start the Domain

1. Navigate to your ***<JavaCAPS51>*\logicalhost** directory (where *<JavaCAPS51>* is the location of your Java Integration 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 appears to indicate 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.

5.6.10 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 for each of your Deployment Profiles.
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. Do this for both of your Deployment Profiles.
2. A message appears when the project is successfully deployed. You can now test your sample.

Note: *Projects can also be deployed from the Enterprise Manager. For more information about using the Enterprise Manager to deploy, monitor, and manage your projects, see the Sun SeeBeyond eGate™ Integrator System Administration Guide.*

5.6.11 Running the Sample

The prjWebSphereMQ_Sample_BPEL Project Demonstrates three different operations (see **"WebSphere MQ eWay eInsight Sample Project" on page 71**).

To run the **Get Message** and **Put Message** operations, the **dp_bpMQReceive** Deployment Profile must be undeployed from the Sun SeeBeyond Enterprise Manager.

To run the **Put Message** and **Receive Message** operations, both the **dp_bpMQGetPut** and **dp_bpMQReceive** Deployment Profiles must be deployed. In this case, the **Receive Message** operation will supersede the **Get Message** operation.

For information on deploying and undeploying your deployments using the Sun SeeBeyond Enterprise Manager, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

To run your deployed sample Project do the following:

1. From your configured input directory, paste (or rename) the sample input file to trigger the eWay.
2. From your output directory, verify the output data.

Chapter 6

Using the WebSphere MQ eWay with Java Collaborations

This chapter provides an introduction to the WebSphere MQ eWay components and information about how these components are created and implemented in an eGate Project. It is assumed that the reader understands the basics of creating a Project using the Enterprise Designer. For more information on creating an eGate Project see the *Sun SeeBeyond eGate Integrator Tutorial* and the *Sun SeeBeyond eGate Integrator User's Guide*.

What's in This Chapter

6.1 WebSphere MQ eWay Components

This chapter presents a sample WebSphere MQ eWay Project created using the same procedures as the sample end-to-end Project provided in the *Sun SeeBeyond eGate Integrator Tutorial*. The eWay components that are unique to the WebSphere MQ eWay include the following:

WebSphere MQ eWay Properties file

The Properties file for the WebSphere MQ eWay contains the parameters necessary to connect with a specific external system. These parameters are set using the Properties dialog box. For more information about the WebSphere MQ eWay Properties file and the Properties dialog box see **"Configuring the WebSphere eWay" on page 34**.

MQSeries OTD

An MQSeries OTD is provided with the eWay and contains methods and attributes used to create the Business Rules that invoke the WebSphere MQ program.

6.2 WebSphere MQ eWay Concerns

WebSphere MQ is unable to segment messages equal to or larger than 100 MB that are sent by way of client connections. In this situation, WebSphere MQ raises an exception with a reason code of 2030 (MQRC_MESSAGE_TOO_BIG_FOR_Q). To resolve this problem, use bindings connections to WebSphere MQ when sending messages equal to or greater than 100 MB.

6.3 WebSphere MQ eWay Sample Project (JCD)

A sample Project for the WebSphere MQ eWay using Java Collaboration Definitions, **WebSphereMQ_Sample_JCD.zip**, is included with the documentation SAR installation.

The **WebSphereMQ_Sample_JCD Project** demonstrates the following:

Get:

1. The File eWay receives a message from an external directory and triggers the Collaboration.
2. The WebSphere MQ eWay, in get mode, retrieves a message from an MQ Queue. The message is published to an outbound File eWay.
3. The File eWay publishes the message to an external directory.

Put:

1. The File eWay receives a message from an external directory and triggers the Collaboration.
2. The WebSphere MQ eWay, in put mode, sends a message to an MQ Queue. The content of the input file is applied to the message.

Receive:

1. The WebSphere MQ eWay, in receive mode, polls the queue. Arriving messages are retrieved and published to an outbound File eWay.
2. The File eWay publishes the message to an external directory.

6.4 Importing a Sample Project

To import a sample eWay Project to the Enterprise Designer do the following:

1. The sample files are uploaded with the eWay’s documentation SAR file and downloaded from the Sun Java Composite Application Platform Suite Installer’s Documentation tab. The **WebSphere_MQ_eWay_Sample.zip** file contains the various sample Project zip files. Extract the samples to a local file.
2. Save all unsaved work before importing a Project.

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

4 Browse to the directory that contains the sample Project zip file. Select the sample file (for this sample, **WebSphereMQ_Sample_JCD.zip**) and click **Import**. After the sample Project is successfully imported, click **Close**.

5 Before an imported sample Project can be run you must do the following:

- Create an Environment (see **"Creating an Environment" on page 106**)
- Configure the eWays for your specific system (see **"Configuring the eWay Properties" on page 107**)
- Create a Deployment Profile (see **"Creating and Activating the Deployment Profile" on page 110**)
- Create and start a domain (see **"Creating and Starting the Domain" on page 111**)
- Build and deploy the Project (see **"Running the Sample" on page 112**)

6.5 Create the WebSphere MQ Queue

The first step in creating the sample Project is to install and configure **IBM's WebSphere MQ Server** and **MQ queue manager** on the local host.

It is assumed that the reader is experienced in the use of the WebSphere MQ queue manager. For the sample implementation do the following:

1 Open IBM WebSphere MQ Explorer.

2 Create a new queue manager.

3 From the WebSphere MQ queue manager create a new queue.

IBM WebSphere MQ Server and Queue Manager Limits and Settings

- When using the WebSphere MQ queue manager on UNIX, you must be a member of the mqm group to create and start the MQ queue manager.
- It is essential that the WebSphere MQ Administrator regularly monitor the number of messages in the queue. Message expiration settings should be set to allow for extended storage.
- WebSphere MQ is limited in the number of messages that can be sent before a commit is executed, and the number of physical messages that can exist on the queue at any one time. This can result in exception errors when upper limits for these numbers are exceeded. Memory and performance of the specific server may also effect the results.

6.6 Creating the WebSphereMQ_Sample_JCD Project

The following pages provide step by step directions for manually creating the **WebSphereMQ_Sample_JCD** Project.

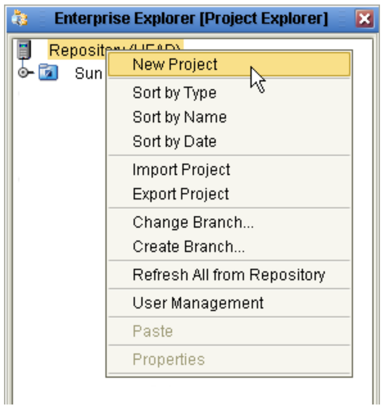
6.6.1 Creating a Project

After you have created the WebSphere MQ queue, the next step is to create a new Project using the Enterprise Designer.

1 Start the Enterprise Designer.

2 From the Enterprise Explorer's Project Explorer tab, right-click the Repository and select **New Project** (see Figure 25). A new Project (Project1) appears on the Project Explorer tree.

Figure 25 Enterprise Explorer - New Project



3 Rename the Project (for this sample, **WebSphereMQ_Sample_JCD**).

6.6.2 Creating the Connectivity Maps

Connectivity Maps provide a canvas for assembling and configuring a Project's components. The WebSphereMQ_Sample_JCD Project uses three Connectivity Maps:

- cmMQGet
- cmMQPut
- cmMQReceive

To create the WebSphereMQ_Sample_JCD Project's Connectivity Maps, do the following:

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

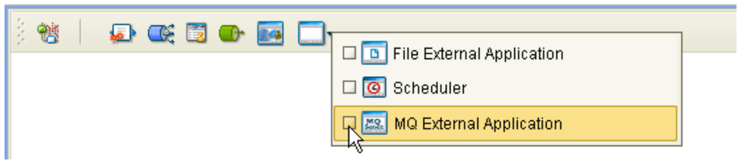
2 The New Connectivity Map appears and a node for the Connectivity Map is added to the Project Explorer tree labeled **CMap1**. Rename the Connectivity Map **cmMQGet**.

3 Create two additional Connectivity Maps named **cmMQPut** and **cmMQReceive**.

Selecting the External Applications

In the Connectivity Maps, eWays are associated with External Systems. For example, to establish a connection to a WebSphere MQ application, you must first select WebSphere MQ (MQ) as an External Application (see Figure 26).

Figure 26 Connectivity Map - External Applications



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

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

Populating the Connectivity Maps

The icons in the toolbar represent the available components used to populate the Connectivity Map canvas. You can add the Project components by dragging the icons from the toolbar to the canvas.

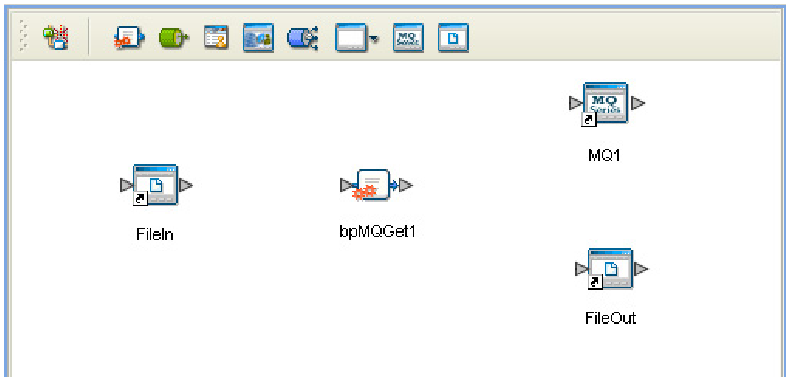
1 To populate the cmMQGet Connectivity Map, drag the following components onto the canvas as displayed in Figure 27.

- File External System (2)
- MQ External System
- Service (A service is a container for Java Collaborations, Business Processes, eTL processes, and so forth)

2 Rename components with the following names (as displayed in Figure 27):

- File1 (File External Application on the left) to **FileIn**
- File2 to **FileOut**
- cmMQGet_Service1 to **jcdMQGet1**

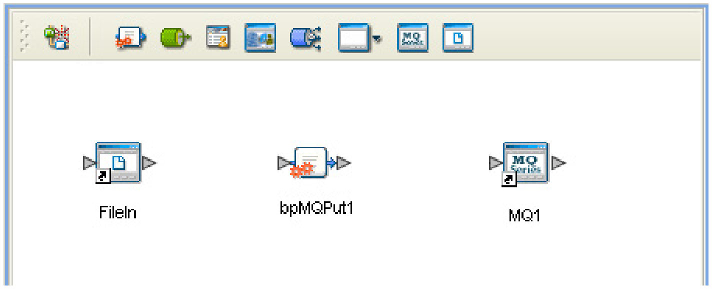
Figure 27 cmMQGet Connectivity Map



3 Open the **cmMQPut** Connectivity and populate the canvas with the following components (as displayed in Figure 28):

- File External System (rename to **FileIn**)
- MQ External System
- Service (rename to **jcdMQPut1**)

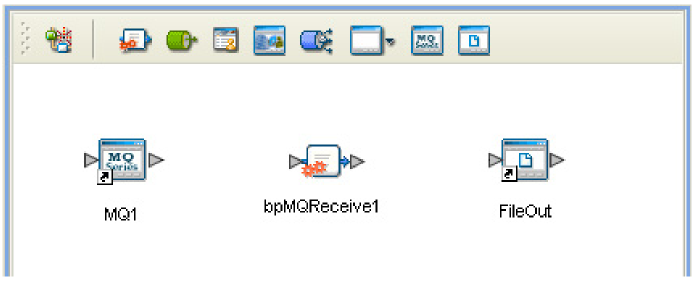
Figure 28 cmMQPut Connectivity Map



4 Populate the **cmMQReceive** Connectivity Map with the following components (as displayed in Figure 29):

- File External System (rename to **FileOut**)
- Service (rename to **jcdMQReceive1**)
- MQ External System

Figure 29 cmMQReceive Connectivity Map



5 Save your current changes to the Repository.

6.6.3 Creating the Java Collaboration Definitions

The next step in the sample is to create the Java Collaboration Definitions using the **Collaboration Definition Wizard (Java)**. Once the Collaborations have been created, the Business Rules of the Collaborations are written using the Collaboration Editor. The WebSphereMQ_Sample_JCD Project includes three Java Collaboration Definitions:

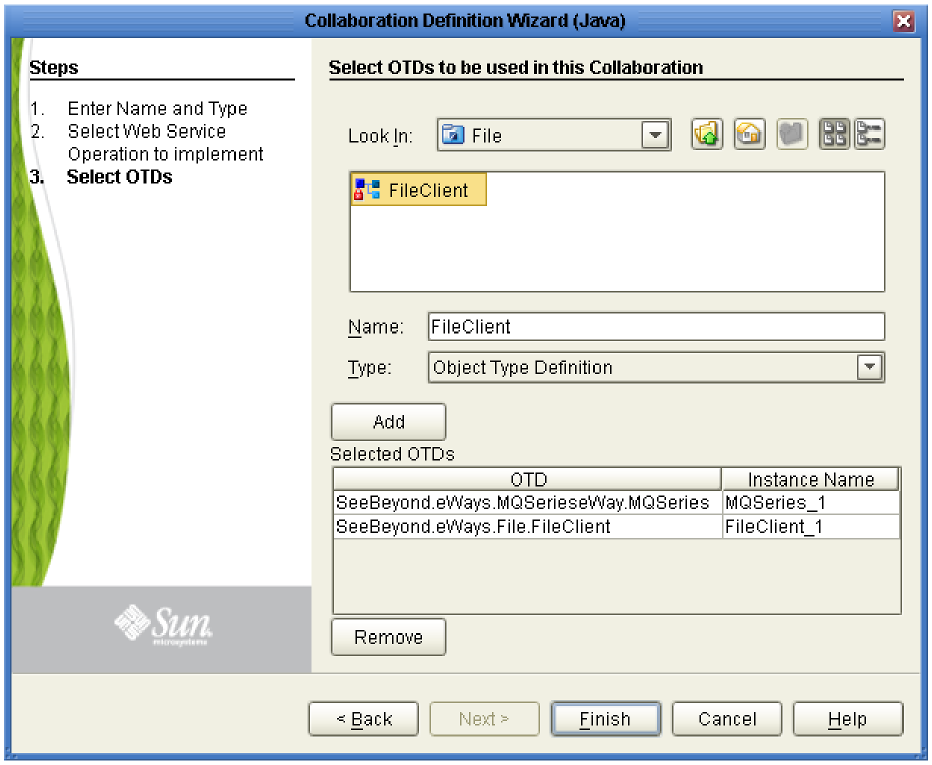
- jcdMQGet
- jcdMQPut
- jcdMQReceive

Creating the jcdMQGet Collaboration

The jcdMQGet Collaboration defines transactions from the WebSphere MQ External Application to the outbound File External Application.

1. From the Project Explorer, right-click the **WebSphereMQ_Sample_JCD** Project and select **New** > **Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard** appears.
2. Enter a Collaboration name (for this sample **jcdMQGet**) 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 Name field now displays **receive**. Click **Next**.
4. For Step 3 of the wizard, double-click **Sun SeeBeyond > eWays > MQSerieseWay > MQSeries.** The **MQSeries_1** OTD is added to the Selected OTDs.
5. Click **UP One Level** to return to the eWay directory, and double-click **Sun SeeBeyond > File** > **FileClient**. The **FileClient_1** OTD is added to the Selected OTDs field (see Figure 30).

Figure 30 Collaboration Definition Wizard (Java) - Select Web Service Interface


Collaboration Definition Wizard (Java)
Steps
1. Enter Name and Type
2. Select Web Service Operation to implement
3. Select OTDs
Select OTDs to be used in this Collaboration
Look In: File
FileClient
Name: FileClient
Type: Object Type Definition
Add
Selected OTDs
Remove
< Back
Next >
Finish
Cancel
Help

OTD	Instance Name
SeeBeyond.eWays.MQSerieseWay.MQSeries	MQSeries_1
SeeBeyond.eWays.File.FileClient	FileClient_1

6 Click **Finish**. The Collaboration Editor (Java) with the new **jcdMQGet** Collaboration appears in the right pane of the Enterprise Designer.

Creating the jcdMQPut Java Collaboration

The **jcdMQPut** Collaboration (Java) defines transactions made between the inbound File eWay and the outbound MQ eWay.

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

2 Enter a Collaboration Definition name (for this sample **jcdMQPut**) 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 Name field now displays **receive.** Click **Next**.

4 For Step 3, **Select OTDs**, from the Select OTDs selection window, double-click **Sun SeeBeyond > eWays > MQSerieseWay > MQSeries**. The Selected OTDs field now lists the **MQSeries_1** OTD.

5 Click **Finish**. The Collaboration Editor with the new **jcdMQPut** Collaboration appears.

Creating the jcdReceive Java Collaboration

The **jcdReceive** Collaboration (Java) defines transactions between the inbound MQ eWay application and the Outbound File eWay.

1. From the Project Explorer, right-click the sample Project and select **New > Collaboration Editor (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
2. Enter a Collaboration Definition name (for this sample **jcdReceive**) and click **Next**.
3. For Step 2 of the wizard, from the Web Services Interfaces selection window, double-click **Sun SeeBeyond** > **eWays** > **eWays > MQSerieseWay > MQSeries** > **receive**. The Name field now displays **receive.** Click **Next**.
4. For Step 3, **Select OTDs**, from the Select OTDs selection window, double-click **Sun SeeBeyond** > **eWays** > **File** > **FileClient**. The **FileClient_1** OTD is added to the Selected OTDs field.
5. Click **Finish**. The Collaboration Editor with the new **jcdReceive** Collaboration appears.
6. Save your current changes to the Repository

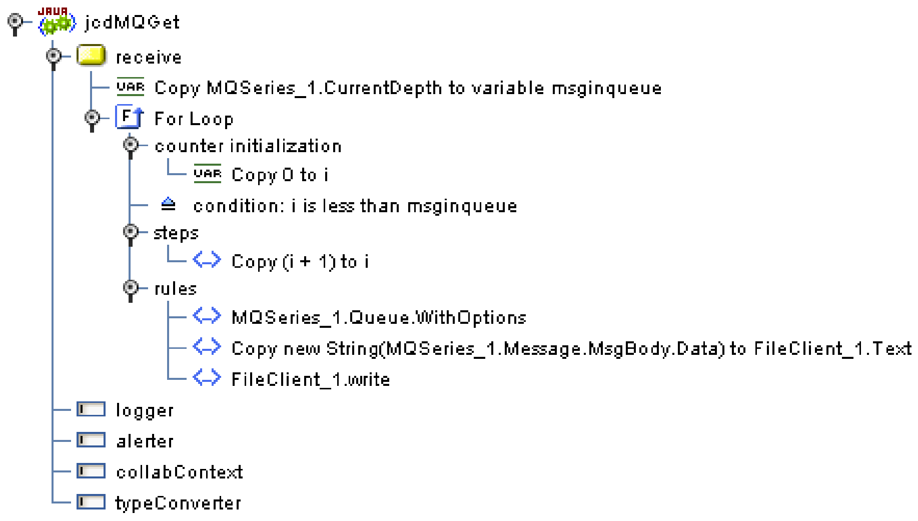
6.6.4 Creating the Business Rules

The next step in the sample is to create the Business Rules of the Collaborations using the Collaboration Editor (Java). The Business Rules for the jcdMQGet, jcdMQPut, and jcdRedeive Collaborations are created using the Collaboration Editor (Java).

Create the jcdMQGet Collaboration Business Rules

The jcdMQGet Collaboration contains the Business Rules displayed in Figure 31.

Figure 31 jcdMQGet Collaboration Business Rules

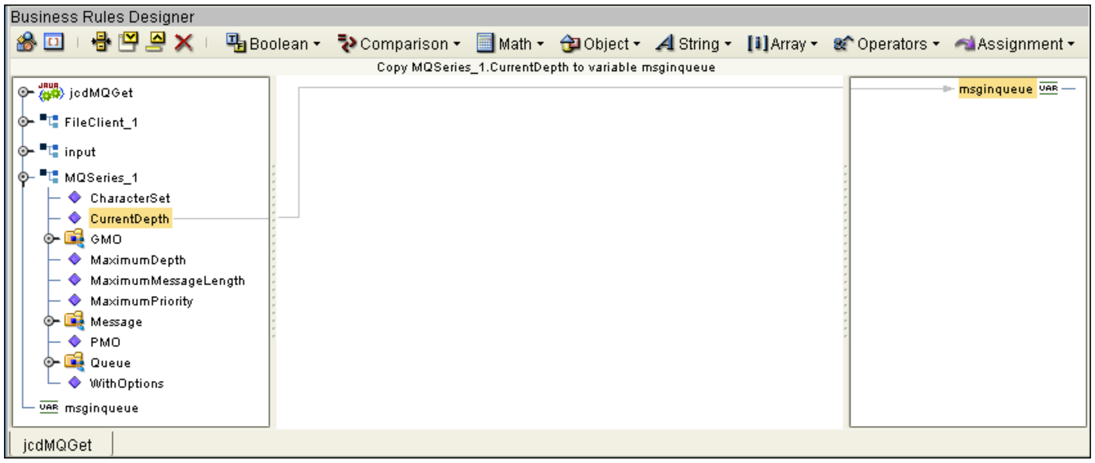


```
jcdMQGet
  receive
    Copy MQSeries_1.CurrentDepth to variable msginqueue
    For Loop
      counter initialization
        Copy 0 to i
      condition: i is less than msginqueue
      steps
        Copy (i + 1) to i
      rules
        MQSeries_1.Queue.WithOptions
        Copy new String(MQSeries_1.Message.MsgBody.Data) to FileClient_1.Text
        FileClient_1.write
  logger
  alerter
  collabContext
  typeConverter
```

To create the jcdMQGet Collaboration Business Rules do the following:

1. From the Project Explorer tree, double-click **jcdMQGet** to open the Collaboration Editor (Java) to the **jcdMQGet** Collaboration.
2. To create comments for the Business Rules, click the comment icon on the Business Rules toolbar. The **Enter a Comment** dialog box appears. Enter the comment and click **OK**. The comment is placed on the Business Rules tree under the last selected item. Once the Comment is created, it can be moved by clicking the comment and dragging it up or down the Business Rules tree to a new location.
3. Create the **Copy MQSeries_1.CurrentDepth to variable msginqueue** (variable) Business Rule:
   - A From the Business Rules toolbar, click the Local Variable icon. The **Create Variable** dialog box appears.
   - B From the **Create Variable** dialog box, enter **msginqueue** as the name, and under **Type** select **Primitive: int**. Click **OK**.
   - C Map the **CurrentDepth** under **MQSeries_1** in the left pane of the Business Rules Designer, to the **msginqueue** variable in the right pane of the Business Rules Designer. To do this, click on the **CurrentDepth** in the left pane of the Business Rules Designer, and drag your cursor to the **msginqueue** variable in the right pane of the Business Rules Designer. A link is now visible between the two nodes.(see **Figure 32 on page 98**).

Figure 32 Collaboration Editor - Business Rules Designer



4 The logic created by the **For Loop** (and the rules under the For Loop) allows the Collaboration to retrieve all of the messages in the queue at one time. Create the **For Loop** with the counter initialization variable:

A From the Business Rules toolbar, click the **For Loop** icon. A For Loop is added to the Business Rules tree.

B Right-click **counter initialization** under the For Loop, and select **Local Variable** from the shortcut menu. The **Create Variable** dialog box appears.

C From the **Create Variable** dialog box, enter **i** as the name, and under **Type** select **Primitive: int**. Click **OK**.

5 Create the **condition: i is less than msginqueue** rule under the For Loop:

A From the Business Rules tree, select **condition: ?** under the For Loop.

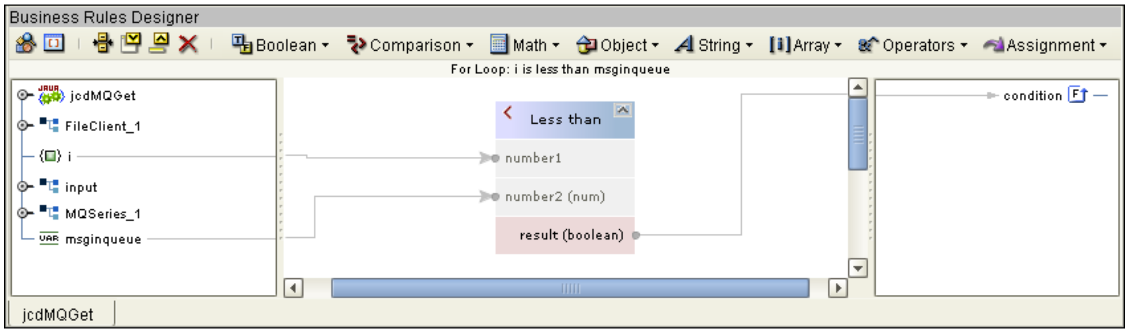
B From the Business Rules Designer's **Comparison** menu, select **Less than**. The **Less than** method box appears.

C Map the **i** field in the left pane of the Business Rules Designer, to the **number1** input node of the **Less than** method box.

D Map the **msginqueue** variable in the left pane of the Business Rules Designer, to the **number2** input node of the **Less than** method box.

E Map the **result (boolean)** output node of the **Less or equal** method box, to **condition** in the right pane of the Business Rules Designer (see Figure 33).

Figure 33 Collaboration Editor - Business Rules Designer



6 Create the **Copy (i + 1) to i** rule under the For Loop > steps:

A From the Business Rules tree, select **steps** under the For Loop, and from the Business Rules toolbar click the **Rule** icon. A new rule is added under For Loop > steps.

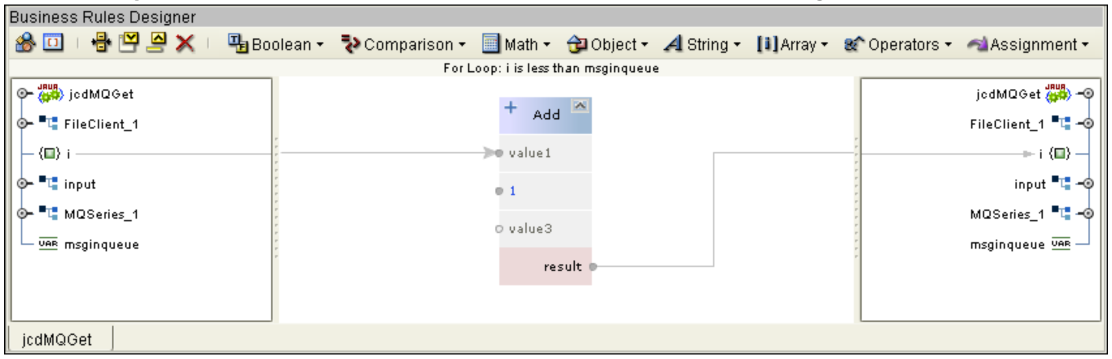
B From the Business Rules Designer's Math menu select **Add**. An Add method box appears.

C Map the **i** variable in the left pane of the Business Rules Designer, to the **value1** input node of the Add method box.

D From the **Add** method box, double-click the **value2** field and enter **1** as the value.

E Map the **result** output node of the **Add** method box, to the **i** variable in the right pane of the Business Rules Designer (see Figure 34).

Figure 34 Collaboration Editor - Business Rules Designer



7 Create the **MQSeries_1.Queue.WithOptions** rule under the For Loop rules:

A From the Business Rules tree, select **rules** under the For Loop, and from the Business Rules toolbar click the **Rule** icon. A new rule is added under For Loop > rules.

B From the left pane of the Business Rules Designer, right-click **Queue** under **MQSeries_1** and select **Select method to call** from the shortcut menu.

C From the method selection window, select **getWithOptions()**. The **getWithOptions** method box appears.

8 Create the **Copy new String(MQSeries_1.Message.MsgBody.Data) to FileClient_1.Text** rule under the For Loop > rules:

A From the Business Rules toolbar click the **Rule** icon to add a new rule.

B From the Business Rules Designer, click the **Class Browser** icon. The **Class Browser** dialog box appears.

C From the **Class Browser** dialog box, select **String** as the class, select the **String(byte[] bytes)** constructor in the right pane, and click **Select**. The **String** method box appears.

D Map **Data** under **MQSeries_1 > Message > MsgBody** in the left pane of the Business Rules Designer, to the **bytes(byte[])** input node of the **String** method box.

E Map the **result(String)** output node of the **String** method box to **Text** under **FileClient_1** in the right pane of the Business Rules Designer.

9 Create the **FileClient_1.write** rule under the For Loop > rules:

A From the Business Rules toolbar click the **Rule** icon to add a new rule.

B From the left pane of the Business Rules Designer, right-click **FileClient_1** and select **Select method to call** from the shortcut menu.

C From the method selection window, select **write()**. The **write** method box appears.

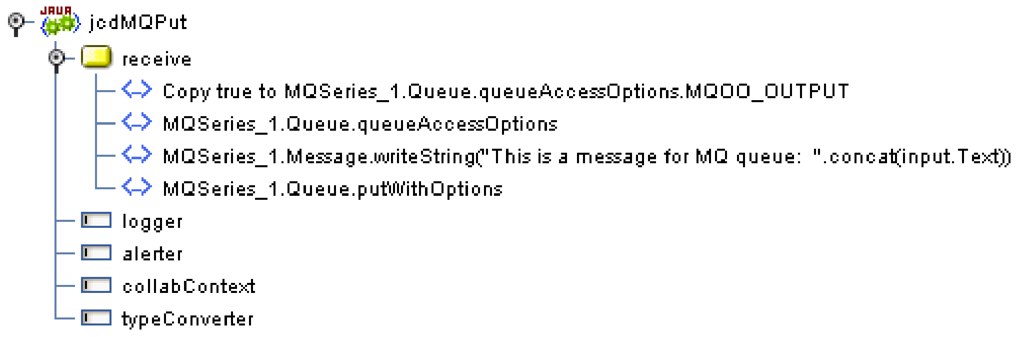
10 From the editor's toolbar, click **Validate** to check the Collaboration for errors.

11 Save your current changes to the repository.

Create the jcdMQPut Collaboration Business Rules

The jcdMQPut Collaboration contains the Business Rules displayed in Figure 35.

Figure 35 jcdMQPut Collaboration Business Rules



```
jcdMQPut
  receive
    Copy true to MQSeries_1.Queue.queueAccessOptions.MQOO_OUTPUT
    MQSeries_1.Queue.queueAccessOptions
    MQSeries_1.Message.writeString("This is a message for MQ queue: ".concat(input.Text))
    MQSeries_1.Queue.putWithOptions
  logger
  alerter
  collabContext
  typeConverter
```

To create the jcdMQPut Collaboration Business Rules do the following:

1 From the Project Explorer tree, double-click **jcdMQPut** to open the Collaboration Editor (Java) to the **jcdMQPut** Collaboration.

2 Create the **Copy true to MQSeries_1.Queue.queueAccessOptions.MQOO_INPUT_AS_Q_DEF** Business Rule:

A From the left pane of the Business Rules Designer, right-click **Queue** under **MQSeries_1** and select **Select method to call** from the shortcut menu.

B From the method selection window, select **getqueueAccessOptions()**. The **getqueueAccessOptions** method box appears.

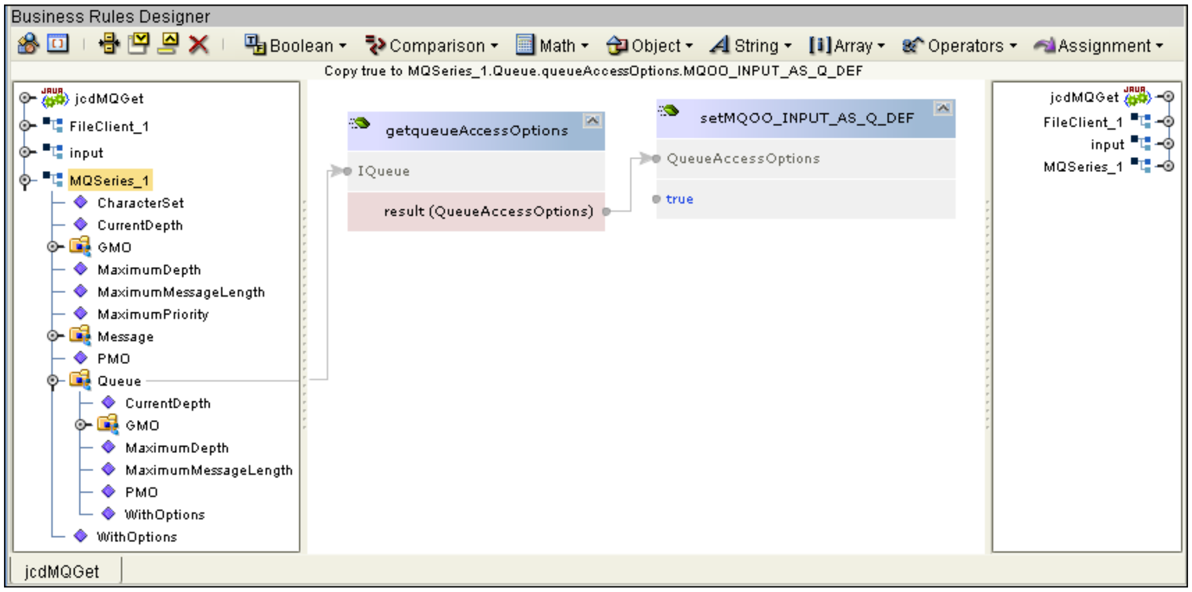
C Right-click the **result(QueueAccessOption)** output node of the **getqueueAccessOptions** method box select **Browse this type** from the shortcut menu. The Class Browser dialog box appears.

D From the Class Browser dialog box select **QueueAccessOptions** as the class, and select the **setMqoo_INPUT_AS_Q_DEF(boolean arg0)** method. Click **Select**. The **setMqoo_INPUT_AS_Q_DEF** method box appears.

E Double-click the arg0(boolean) input node of the **setMqoo_INPUT_AS_Q_DEF** method box and enter **true** as the value.

F Map the **result(QueueAccessOption)** output node of the **getqueueAccessOptions** method box to the **QueueAccessOption** input node of the **setMqoo_INPUT_AS_Q_DEF** method box. To do this, click on the **result(QueueAccessOption)** output node of the **getqueueAccessOptions** method box, and drag your cursor to the **QueueAccessOption** input node of the **setMqoo_INPUT_AS_Q_DEF** method box.(see Figure 36).

Figure 36 Collaboration Editor - Business Rules Designer



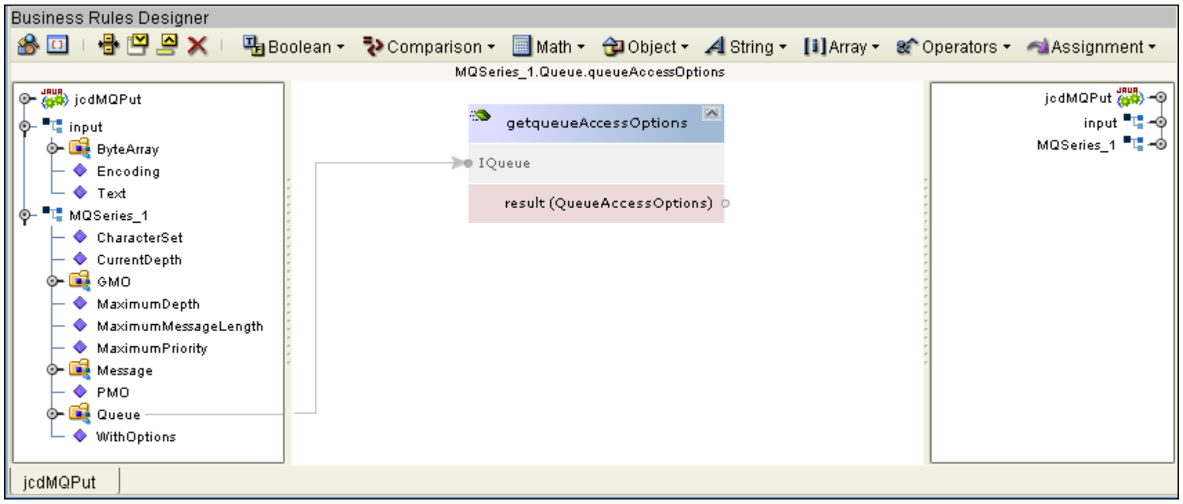
3 Create the **MQSeries_1.Queue.queueAccessOptions** Business Rule:

A From the Business Rules toolbar click the **rule** button to add a new rule.

B From the left pane of the Business Rules Designer, right-click **Queue** under **MQSeries_1** and select **Select method to call** from the shortcut menu.

C From the method selection window, select **getqueueAccessOptions()**. The **getqueueAccessOptions** method box appears (see **Figure 37 on page 102**).

Figure 37 Collaboration Editor - Business Rules Designer



4 Create the **MQSeries_1.Message.writeString("This is a message for MQ queue: ".concat(input.Text))** rule:

A From the Business Rules toolbar click the **rule** button to add a new rule.

B From the left pane of the Business Rules Designer, right-click **Message** under **MQSeries_1** and select **Select method to call** from the shortcut menu.

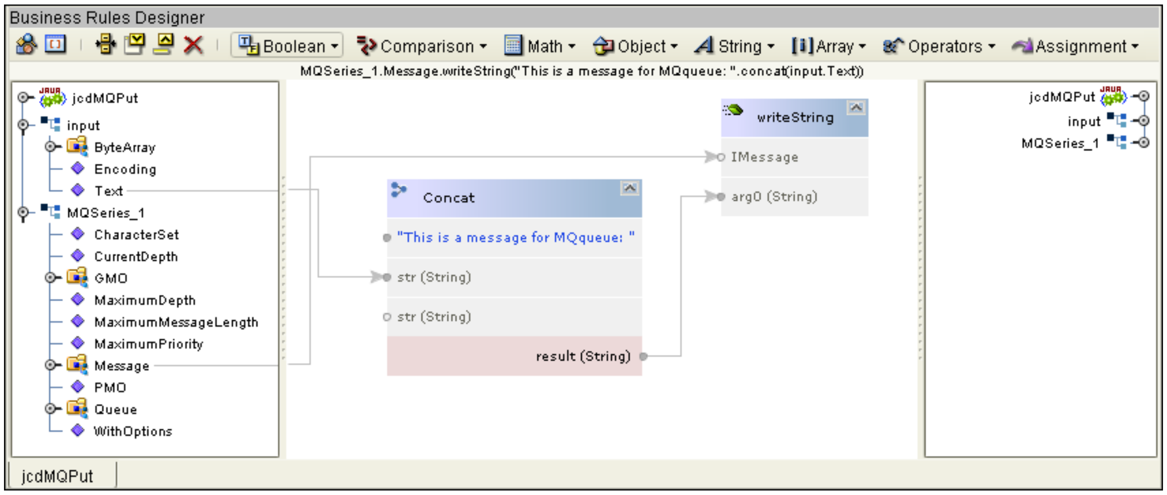
C From the method selection window, select **writeString*(String arg0)***. The **writeString** method box appears.

D From the Business Rules Designer's String menu, select **concat**. The concat method box appears. Double-click the first String field and enter **This is a message for MQ queue:** as the value.

E Map **Text** under **input** in the left pane of the Business Rules Designer, to the first **str(String)** input node of the **concat** method box.

F Map the **result(String)** output node of the **concat** method box, to the **arg0(String)** input node of the **writeString** method box (see Figure 38).

Figure 38 Collaboration Editor - Business Rules Designer



5 Create the **MQSeries_1.Queue.putWithOptions** Business Rule:

A From the Business Rules toolbar click the **rule** button to add a new rule.

B From the left pane of the Business Rules Designer, right-click **Queue** under **MQSeries_1** and select **Select method to call** from the shortcut menu.

C From the method selection window, select **putWithOptions()**. The **putWithOptions** method box appears.

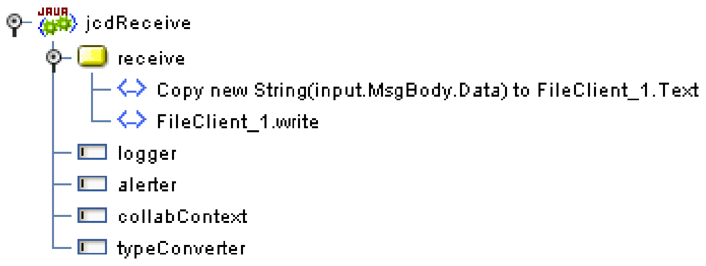
6 From the editor's toolbar, click **Validate** to check the Collaboration for errors.

7 Save your current changes to the repository.

Create the jcdReceive Collaboration Business Rules

The jcdReceive Collaboration contains the Business Rules displayed in Figure 35.

Figure 39 jcdReceive Collaboration Business Rules



To create the jcdReceive Collaboration Business Rules do the following:

1 From the Project Explorer tree, double-click **jcdReceive** to open the Collaboration Editor (Java) to the **jcdReceive** Collaboration.

2 Create the **Copy new String(input.MsgBody.Data) to FileClient_1.Text** Business Rule:

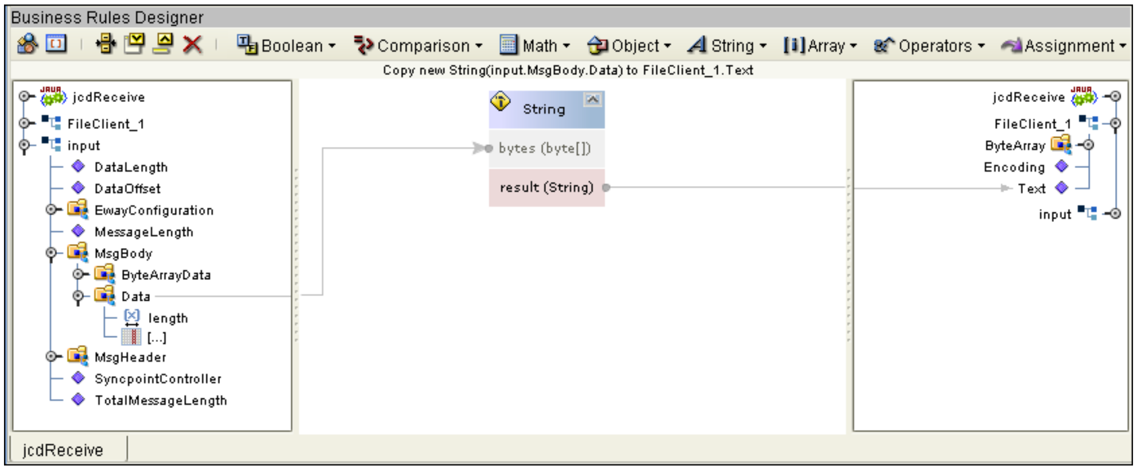
A From the Business Rules Designer toolbar click Class Browser. The Class Browser dialog box appears.

B From the Class Browser dialog box select **String** as the class, and select the **String(byte[] bytes)** constructor in the right pane. Click **Select**. The **String** constructor method box appears.

C Map **Data** under **input > MsgBody** in the left pane of the Business Rules Designer, to the **bytes(byte[])** input node of the **String** constructor method box.

D Map the **result(String)** output node of the **String** constructor method box, to **Text** under **FileClient_1** in the right pane of the Business Rules Designer (see **Figure 40 on page 104**).

Figure 40 Collaboration Editor - Business Rules Designer



3 To create the **FileClient_1.write** Business Rule do the following:

A From the Business Rules toolbar click the **rule** button to add a new rule.

B Right-click the **FileClient_1** node in the left pane of the Business Rule Designer, and click **Select method to call**. The method selection menu appears.

C Select **write()** from the method selection menu. The **write** method box appears.

4 From the editor's toolbar, click **Validate** to check the Collaboration for errors.

5 Save your current changes to the repository.

For information on how to create Business Rules using the Collaboration Editor see the *Sun SeeBeyond eGate Integrator User's Guide*.

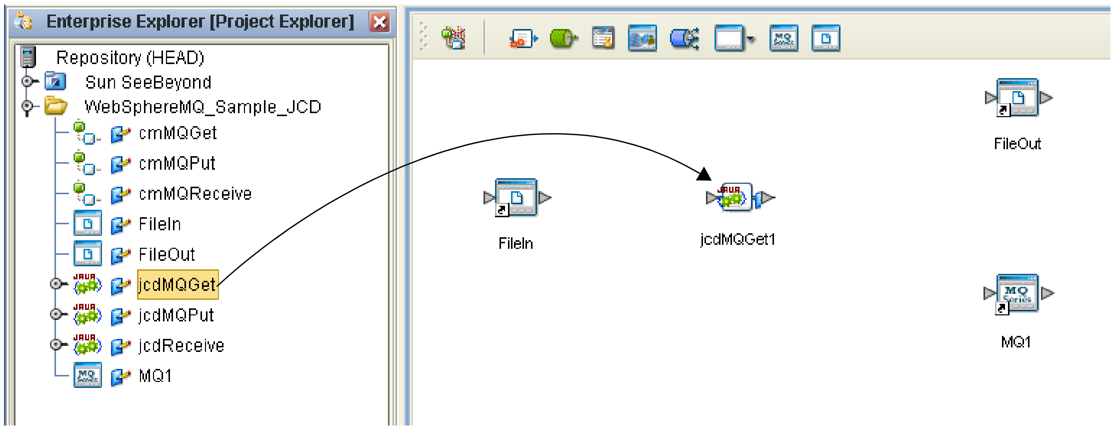
6.6.5 Binding the eWay Components

After the Collaborations and Connectivity Maps have been created, the components can be associated and bindings created in the Connectivity Maps.

1 From the Project Explorer, double-click **cmMQGet** to display the **cmMQGet** Connectivity Map.

2 Drag and drop the **jcdMQGet** Business Process from the Project Explorer to the **jcdMQGet1** Service. If the Business Process was successfully associated, the Service icon changes to a Business Process (see **Figure 41 on page 105**).

Figure 41 cmMQGet Connectivity Map - Binding the Components



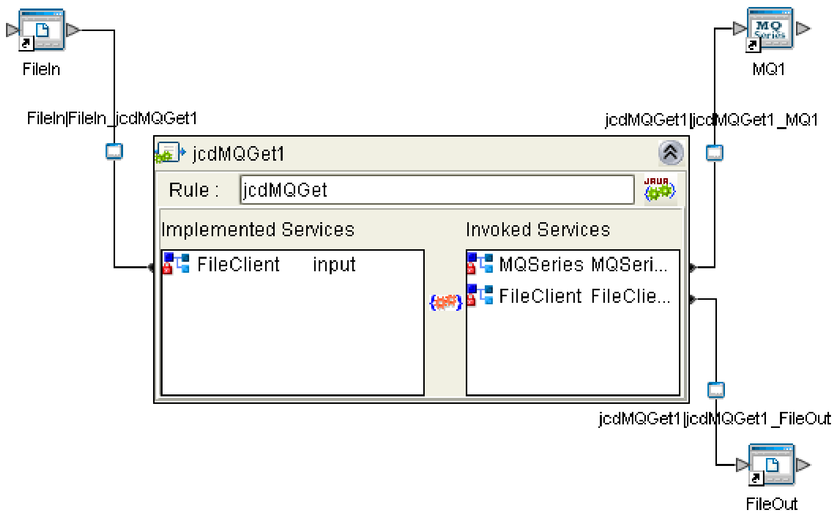
3 From the Connectivity Map canvas, double-click **jcdMQGet1**. The **jcdMQGet1** binding dialog box appears using the **jcdMQGet** Rule.

4 From the **jcdMQGet1** binding dialog box, map **FileClient Input** (under Implemented Services) to the output node of the inbound **FileIn** External Application. To do this, click on **FileClient Input** under Implemented Services in the **jcdMQGet1** binding box, and drag your cursor to the output node of the **FileIn** External Application. A link now joins the two components.

5 From the **jcdMQGet1** binding dialog box, map **MQSeries** (under Invoked Services) to the input node of the **MQ1** External Application.

6 From the **jcdMQGet1** binding dialog box, map **FileClient** (under Invoked Services) to the input node of the **FileOut** External Application. (see Figure 42).

Figure 42 cmMQGet Connectivity Map - Binding the Components



7 Minimize the **jcdMQGet1** binding dialog box.

8 From the Project Explorer, double-click **cmMQPut** to display the **cmMQPut** Connectivity Map.

9 Drag and drop the **jcdMQPut** Business Process from the Project Explorer to the **jcdMQPut1** Service.

10 From the Connectivity Map, double-click the **jcdMQPut1** Service. The **jcdMQPut1** binding dialog box appears with the **jcdMQPut** Rule.

11 From the **jcdMQPut1** binding dialog box, map **FileClient Input** (under Implemented Services) to the output node of the **FileIn** External Application.

12 From the **jcdMQPut1** binding dialog box, map **MQSeries** (under Invoked Services) to the input node of the **MQ1** External Application.

13 Minimize the **jcdMQPut1** binding dialog box.

14 From the Project Explorer, double-click **cmMQReceive** to display the **cmMQReceive** Connectivity Map.

15 Drag and drop the **jcdMQReceive** Business Process from the Project Explorer to the **jcdMQReceive1** Service.

16 From the Connectivity Map, double-click the **jcdMQReceive1** Service. The **jcdMQReceive1** binding dialog box appears with the **jcdMQReceive** Rule.

17 From the **jcdMQReceive1** binding dialog box, map **MQSeries Input** (under Implemented Services) to the output node of the **MQ1** External Application.

18 From the **jcdMQReceive1** binding dialog box, map **FileClient** (under Invoked Services) to the input node of the outbound **FileOut** External Application.

19 Minimize the **jcdMQReceive1** binding dialog box, and save all your current changes to the Repository.

6.6.6 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 Explorer and 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 **envWebSphereMQSeries_JCD**.

4 Right-click **envWebSphereMQSeries_JCD** and select **New > MQ External System**. Name the External System **esMQ** and click **OK**. **esMQ** is added to the Environment Editor.

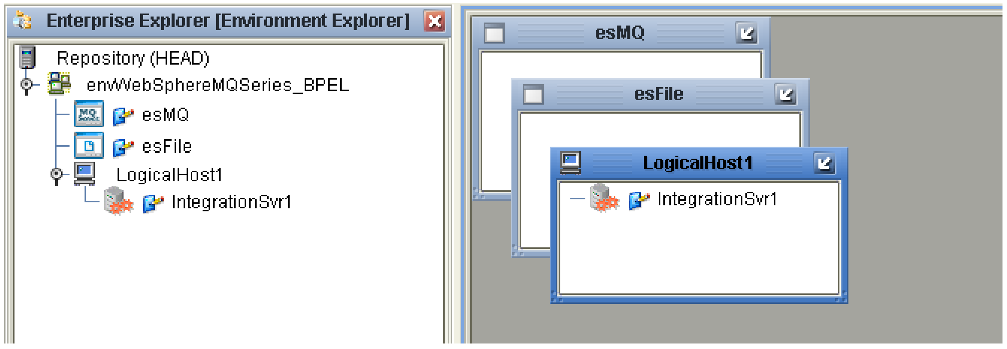
5 Right-click **envWebSphereMQSeries_JCD** and select **New > File External System**. Name the External System **esFile** and click **OK**. **esFile** is added to the Environment Editor.

6 Right-click **envWebSphereMQSeries_JCD** and select **New > Logical Host**. **LogicalHost1** is added to the Environment Editor.

7 From the Environment Explorer tree, right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server**. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1.

8 Save changes to the repository. The Environment Explorer and Environment Editor now appear as displayed in Figure 43.

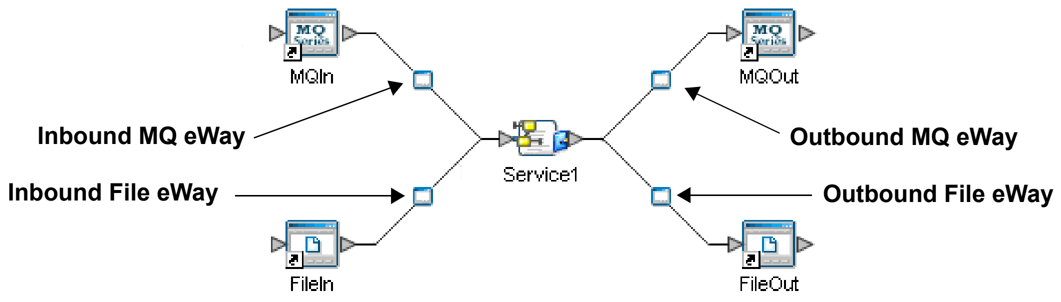
Figure 43 Environment Editor



6.6.7 Configuring the eWay Properties

The **WebSphereMQ_Sample_JCD** Project contains WebSphere MQ eWays and File eWays, each represented in the Connectivity Map as a node between an External Application and a Service. eWays facilitate communication and movement of data between the External Applications and the eGate system (see Figure 44).

Figure 44 eWays



The eWay properties are configured from both the Connectivity Map and the Environment. To configure the eWays do the following:

Configuring the File eWay Properties

1 From the cmMQGet Connectivity Map, double-click the inbound **FileIn** eWay (see Figure 44). The **Properties Editor** opens to the inbound File eWay properties.

2 Modify the properties for your system, including the settings for the inbound File eWay in **Table 22 on page 108**, and click **OK**.

Table 22 cmMQGet - FileIn eWay Settings

Inbound File eWay Connection Parameters	
Input file name	filetrigger_mq.txt

3 In the same way, modify the properties of the cmMQPut Connectivity Map's inbound **FileIn** eWay, entering **filein_mq.txt** as the Input file name property value.

4 From the cmMQGet and cmMQReceive Connectivity Maps, modify the outbound **FileOut** eWay properties for your system, including the settings in Table 23.

Table 23 Outbound File eWay Settings

Outbound File eWay Connection Parameters	
Output file name	fileout_mq.dat

5 From the **Environment Explorer** tree, right-click the File eWay External System (**esFile** in this sample), and select **Properties** from the shortcut menu. The Properties Editor appears.

6 Modify the File eWay Environment properties for your system, including the settings in Table 24, and click **OK**.

Table 24 File eWay Environment Properties

File eWay Environment Properties	
Inbound File eWay > Parameter Settings Set as directed, otherwise use the default settings	
Directory	*Select a directory, for example C:/temp*
Outbound File eWay > Parameter Settings Set as directed, otherwise use the default settings	
Directory	*Select a directory, for example C:/temp*

Configuring the WebSphere MQ eWay Properties

The WebSphere MQ eWay properties must be set in both the Project Explorer and Environment Explorer. For more information on the WebSphere MQ eWay properties and the Properties Editor, see **"Creating and Configuring the WebSphere MQ eWay" on page 34** or see the *Sun SeeBeyond eGate Integrator User's Guide*.

Modifying the WebSphere MQ eWay Connectivity Map Properties

1 From the cmMQGet Connectivity Map, double-click the MQ eWay. The Properties Editor opens to the outbound WebSphere MQ eWay Connectivity Map properties.

2 Modify the inbound MQ eWay properties for your system and click **OK**.

3 In the same way, open the cmMQPut Connectivity Map and modify the outbound MQ eWay properties for your system.

4 In the same way, open the cmMQReceive Connectivity Map and modify the inbound MQ eWay properties for your system.

Modifying the WebSphere MQ eWay Environment Explorer Properties

1. From the **Environment Explorer** tree, right-click the WebSphere MQ External System (**esMQ** in this sample), and select **Properties**. The Properties Editor opens to the WebSphere MQ eWay Environment properties.
2. Modify the WebSphere MQ eWay Environment properties for your system, including the settings in Table 25, and click **OK**.

Table 25 WebSphere MQ Environment Explorer eWay Settings

MQ eWay Environment Explorer Properties	
Inbound MQSeries eWay - Inbound eWay Environment Configuration Set as directed, otherwise use the default settings.	
MQ Host Name	*The name of the specific queue manager server*
Port Number	*Port number on which the queue manager is bound. The default is 1414.*
Queue Manager Name	*The name of the WebSphere MQ queue manager*
User ID	User ID required by queue manager
Password	Password required by queue manager
Outbound MQSeries eWay - Outbound eWay Environment Configuration Set as directed, otherwise use the default settings.	
MQ Host Name	*The name of the specific queue manager server*
Port Number	*Port number on which the queue manager is bound. The default is 1414.*
Queue Manager Name	*The name of the WebSphere MQ queue manager*
User ID	User ID required by queue manager
Password	Password required by queue manager

6.6.8 Configuring the Integration Server

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

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.
3. Click the ellipsis. The **Password Settings** dialog box appears. 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*.

6.6.9 Creating and Activating the Deployment Profile

Deployment Profiles are specific instances of a Project in a particular Environment. A Deployment Profile contains information regarding the assignment of services and message destinations to integration and message servers (JMS IQ Managers). It also contains version information for all versionable objects in the Project. Deployment Profiles are created using the Deployment Editor.

The WebSphereMQ_Sample_JCD Project performs multiple operations and requires two different Deployment Profiles; **dp_jcdMQGetPut**, and **dp_jcdReceive**. This allows part of the Project to be undeployed.

Create the dp_jcdMQGetPut Deployment Project

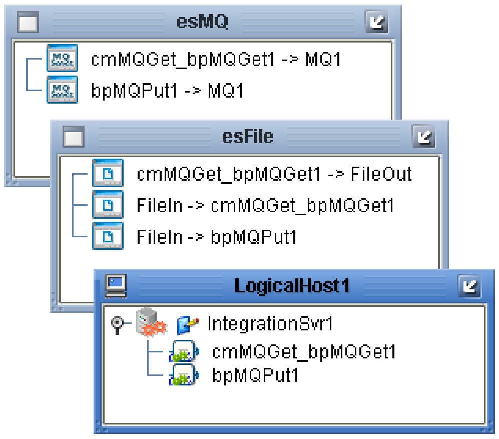
To create the dp_jcdMQGetPut Deployment Profile do the following:

1 From the Project Explorer, right-click the **WebSphereMQ_Sample_JCD** Project and select **New** > **Deployment Profile** from the shortcut menu. The **Create Deployment Profile for WebSphereMQ_Sample_JCD** dialog box appears.

2 Enter **dp_jcdMQGetPut** as the name for the Deployment Profile. Select **envWebSphereMQSeries_JCD** as the Environment, and make sure that only the **cmMQGet** and **cmMQPut** Connectivity Maps are selected (checked). Click **OK**.

3 From the Deployment Editor, click **Automap**. This automatically maps all of your Project components to the correct External Systems and Integration Server. The **Automap Results** dialog box appears. Click **Close**. The **dp_jcdMQGetPut** Deployment Profile now contains the components displayed in Figure 45.

Figure 45 dp_jcdMQGetPut Deployment Profile



Create the dp_jcdReceive Deployment Project

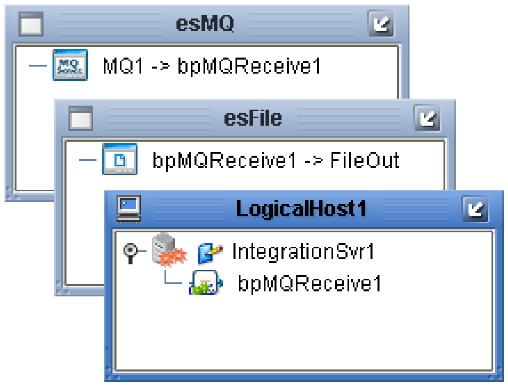
To create the dp_jcdReceive Deployment Profile do the following:

1 From the Project Explorer, right-click the **WebSphereMQ_Sample_JCD** Project and select **New** > **Deployment Profile** from the shortcut menu. The **Create Deployment Profile for WebSphereMQ_Sample_JCD** dialog box appears.

2 Enter **dp_jcdReceive** as the name for the Deployment Profile. Select **envWebSphereMQSeries_JCD** as the Environment, and make sure that only the **cmMQReceive** Connectivity Map is selected (checked). Click **OK**.

3 From the Deployment Editor, click **Automap**. Review and close the **Automap Results** dialog box. The **dp_jcdReceive** Deployment Profile now contains the components displayed in Figure 45.

Figure 46 dp_jcdMQReceive Deployment Profile



4 Save your current changes to the Repository.

6.6.10 Creating and Starting the Domain

To deploy your Project, you must first create a domain. After the domain is created, the Project is built and then deployed.

Create and Start the Domain

1 Navigate to your ***<JavaCAPS51>*\logicalhost** directory (where *<JavaCAPS51>* is the location of your Java Integration 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 appears to indicate 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.

6.6.11 Building and Deploying the Project

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

Build the Project

1 From the Deployment Editor toolbar, click the **Build** icon for each of your Deployment Profiles.

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. Do this for both of your Deployment Profiles.
2. A message appears when the project is successfully deployed. You can now test your sample.

Note: *Projects can also be deployed from the Enterprise Manager. For more information about using the Enterprise Manager to deploy, monitor, and manage your projects, see the Sun SeeBeyond eGate™ Integrator System Administration Guide.*

6.6.12 Running the Sample

The WebSphereMQ_Sample_JCD Project Demonstrates three different operations (see **"WebSphere MQ eWay Sample Project (JCD)" on page 89**).

To run the **Put Message** and **Get Message** operations, the **dp_jcdReceive** Deployment Profile must be undeployed from the Sun SeeBeyond Enterprise Manager.

To run the **Put Message** and **Receive Message** operations, both the **dp_jcdMQGetPut** and **dp_jcdMQReceive** Deployment Profiles must be deployed. In this case, the **Receive Message** operation will supersede the **Get Message** operation.

For information on deploying and undeploying your Projects using the Sun SeeBeyond Enterprise Manager, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

To run your deployed sample Project do the following:

1. From your configured input directory, paste (or rename) the sample input file to trigger the eWay.
2. From your output directory, verify the output data.

Appendix A

Mapping WebSphere MQ Header Fields

This section describes the mapping of JMS Standard header items and their equivalent WebSphere MQ header fields.

What's in this Appendix:

A.1 Mapping Between JMS Standard Header Items and WebSphere MQ Header Fields

JMS Standard header items and their equivalent WebSphere MQ header fields can be set using the Collaboration Editor (Java). For information on mapping between JMS header items and WebSphere MQ header fields see IBM MQSeries online documentation at:

http://www-306.ibm.com/software/integration/mqfamily/library/manualsa/csqzaw06/csqzaw065b.htm

Chapter 12, Table 20, of the IBM document at the above Web site provides JMS header fields used to set or get MQSeries header fields (only some of which are available using this procedure). The Collaboration Editor (Java) sets the header properties by calling **readProperty()** or **writeProperty()**.

For detailed information on creating Business Rules using the Collaboration Editor (Java) see the *Sun SeeBeyond eGate Integrator User's Guide*.

Index