

SeeBeyond ICAN Suite

COM/DCOM eWay Intelligent Adapter User's Guide

Release 5.0.1



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

SeeBeyond, e*Gate, e*Way, and e*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

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 This work is protected as an unpublished work under the copyright laws.

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

Version 20041118155125.

Contents

Chapter 1

Introducing the COM/DCOM eWay	6
About COM/DCOM	6
The COM/DCOM eWay Intelligent Adapter	7
What's New in This Release	7
About this Document	7
Organization of Information	7
Scope of the Document	8
Intended Audience	8
Document Conventions	8
SeeBeyond Web Site	9
SeeBeyond Documentation Feedback	9

Chapter 2

Installing the COM/DCOM eWay	10
Supported Operating Systems	10
System Requirements	10
External System Requirements	11
Installing the COM/DCOM eWay	11
Installing the COM/DCOM eWay	11
Adding the eWay to an Existing ICAN Suite Installation	12
Adding JAR files to the Integration Server Classpath	13
Adding DLL files to the Integration Server Library Path	13
After Installation	14
Increasing the ICAN Enterprise Designer Heap Size	14

Chapter 3

Configuring the eWay Properties	15
Configuring the COM/DCOM eWay	15
Selecting COM/DCOM as the External Application	15
Using the Properties Editor	16

COM/DCOM eWay Properties	17
Server	17
Dynamic Configuration	17
Alerting and Logging	18

Chapter 4

COM/DCOM eWay OTD Overview	19
Object Type Definitions	19
The COM OTDs	19
Create Methods	20
Query Methods	20
Using the COM OTD Wizard	22

Chapter 5

Implementing a COM/DCOM eWay Project	26
COM/DCOM eWay Components	26
Supported Data Types	27
SAFEARRAY Constraints	27
COM/DCOM eWay Considerations	28
COM/DCOM eWay Sample Projects	28
The COMeWay_MSWord Sample Project	28
The COMeWay_MSExcel Sample Project	29
The COMeWay_MSDAO Sample Project	30
Importing a Sample Project	30
Creating the COMeWay_MSWord Project	31
Create a Project	31
Create a Connectivity Map	31
Select the External Applications	31
Populate the Connectivity Map	32
Creating the otdMSWord OTD	32
Creating the Collaboration Definitions	34
The jceMSWord Collaboration (Java)	34
Using the Collaboration Editor (Java)	35
Creating Collaboration Bindings	51
Creating an Environment	53
Configuring the eWays	54
Configuring the COM/DCOM eWay	55
Creating and Activating the Deployment Profile	56
Running the Project	57
Creating the COMeWay_MSExcel Sample Project	58
Create a Project	58
Create a Connectivity Map	58

Contents

Select the External Applications	58
Populate the Connectivity Map	58
Creating the otdMSEExcel OTD	59
Creating the Collaboration Definitions	60
The jceMSEExcel Collaboration (Java)	60
Using the Collaboration Editor (Java)	61
Creating Collaboration Bindings	62
Creating an Environment	63
Configuring the eWays	64
Configuring the COM/DCOM eWay	65
Creating and Activating the Deployment Profile	66
Running the Project	67
Implementing the COMeWay_MSDAO Sample Project	68
Preparing the Imported COMeWay_MSDAO Sample Project	68
Running the Project	68

Chapter 6

Java Methods and Classes for the COM/DCOM eWay	70
COM/DCOM eWay Classes	70
COM/DCOM Javadoc	71
COM/DCOM Runtime Exceptions	71
Index	72

Introducing the COM/DCOM eWay

This document describes how to install and configure the COM/DCOM eWay Intelligent Adapter (also called the COM/DCOM eWay throughout this document), as well as how to implement the eWay in a typical eGate environment.

This chapter provides a brief overview of operations and components, general features, and system requirements of the COM/DCOM eWay.

What's in This Document

- [About COM/DCOM](#) on page 6
- [The COM/DCOM eWay Intelligent Adapter](#) on page 7
- [What's New in This Release](#) on page 7
- [About this Document](#) on page 7
- [SeeBeyond Web Site](#) on page 9

1.1 About COM/DCOM

The Microsoft™ *Component Object Model* (COM) is a component software architecture that allows developers to partition an application into multiple components that can be developed and installed independently of each other. COM is the underlying architecture that forms the foundation for higher-level software services, like those provided by OLE (Object Linking and Embedding). OLE services span various aspects of component software, including compound documents, custom controls, inter-application scripting, data transfer, and other software interactions. Using COM allows software objects to be reused for a variety of applications. Because of its binary standard, COM allows any two components to communicate regardless of the language in which they were written.

The Microsoft *Distributed Component Object Model* (DCOM) is an extension of COM, and supports communication among objects residing on different computers; LANs, WANs, and the Internet. With DCOM, these software objects can be reused over a distributed environment.

COM objects or components are individual modular software routines that can be reused within applications. COM objects are reusable compiled binary objects, as opposed to reusable sections of code. Creating an instance of a COM object provides a reference through which you can access the object's functionality.

1.2 The COM/DCOM eWay Intelligent Adapter

The COM/DCOM eWay allows the SeeBeyond Integrated Composite Application Network (ICAN) Suite to create an instance of an automation-compatible COM object and access the methods and properties of that object.

One aspect of COM is "automation" based on the **COM IDispatch interface**. Objects that implement the IDispatch interface are known to be automation-compatible. Automation-compatible components are said to be "scriptable" and/or are capable of being "driven" by an automation client. This is possible because the IDispatch interface is well-known and those components that implement this interface adhere to a strict contract that is based on a "**late binding**" concept. "Late binding" refers to a programming principle whereby the actual operation invoked is not determined until runtime. Objects that implement the IDispatch interface achieve this through the concept of the **Invoke** method on the IDispatch interface, which allows the user to call a method by name. The COM/DCOM eWay is designed to work with automation-compatible components: that is, those that implement the **IDispatch interface**.

1.3 What's New in This Release

Version 5.0.1 is a maintenance release of the COM/DCOM eWay. This user's guide includes the following changes:

- Screenshots have been updated to reflect changes to ICAN Suite 5.0.5.
- Support has been added for the Deployment Editor Auto Map feature (see [Creating and Activating the Deployment Profile](#) on page 56).

1.4 About this Document

This section provides a brief outline of the COM/DCOM eWay User's Guide.

1.4.1. Organization of Information

This book includes the following chapters:

- **Chapter 1 "Introducing the COM/DCOM eWay"** provides an overview of the Microsoft™ *Component Object Model* (COM) and the COM/DCOM eWay.
- **Chapter 2 "Installing the COM/DCOM eWay"** provides the supported operating systems and system requirements for the COM/DCOM eWay. It also includes directions for installing the COM/DCOM eWay and accessing the accompanying documentation and sample projects.
- **Chapter 3 "Configuring the eWay Properties"** describes the process of configuring the COM/DCOM eWay to run in your environment.

- **Chapter 4 “COM/DCOM eWay OTD Overview”** provides an overview of the Object Type Definitions (OTDs) created from the COM components Type Library files. It also describes how to use the COM OTD Wizard to create these OTDs.
- **Chapter 5 “Implementing a COM/DCOM eWay Project”** describes the features and functionality of the COM/DCOM eWay using the Enterprise Designer and the Collaboration Editor (Java).
- **Chapter 6 “Java Methods and Classes for the COM/DCOM eWay”** describes the COM/DCOM eWay Java classes and provides directions for accessing the COM/DCOM eWay Javadoc.

1.4.2. Scope of the Document

This user’s guide provides a description of the COM/DCOM eWay Intelligent Adapter. 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 COM/DCOM 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 ICAN Suite system. This person must also understand any operating systems on which the ICAN Suite will be installed (Windows, UNIX, and/or HP NonStop Server), and must be thoroughly familiar with Windows-style GUI operations.

1.4.4. Document Conventions

The following conventions are observed throughout this document.

Table 1 Document Conventions

Text	Convention	Example
Names of buttons, files, icons, parameters, variables, methods, menus, and objects	Bold text	<ul style="list-style-type: none"> ▪ Click OK to save and close. ▪ From the File menu, select Exit. ▪ Select the logicalhost.exe file. ▪ Enter the timeout value. ▪ Use the getClassname() method. ▪ Configure the Inbound File eWay.
Command line arguments, code samples	Fixed font. Variables are shown in <i>bold italic</i> .	<code>bootstrap -p <i>password</i></code>
Hypertext links	Blue text	See “SeeBeyond Web Site” on page 9
Hypertext links for Web addresses (URLs) or email addresses	Blue underlined text	http://www.seebeyond.com docfeedback@seebeyond.com

1.5 SeeBeyond Web Site

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

<http://www.seebeyond.com>

1.6 SeeBeyond Documentation Feedback

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

docfeedback@seebeyond.com

Installing the COM/DCOM eWay

This chapter explains the procedures for installing the COM/DCOM eWay.

What's in This Document

- [Supported Operating Systems](#) on page 10
- [System Requirements](#) on page 10
- [Installing the COM/DCOM eWay](#) on page 11
- [Adding JAR files to the Integration Server Classpath](#) on page 13
- [Adding DLL files to the Integration Server Library Path](#) on page 13
- [After Installation](#) on page 14

2.1 Supported Operating Systems

The COM/DCOM eWay is available for the following operating systems:

- Windows XP, Windows 2000, and Windows Server 2003

2.2 System Requirements

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

Although the COM/DCOM 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.

For additional system requirements, see the **COM/DCOM eWay Readme**, accessed from the Enterprise Manager's **Documentation** tab.

2.2.1. External System Requirements

The COM/DCOM eWay has no external system requirements.

2.3 Installing the COM/DCOM eWay

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

2.3.1. Installing the COM/DCOM eWay

The ICAN Suite installation process includes the following operations:

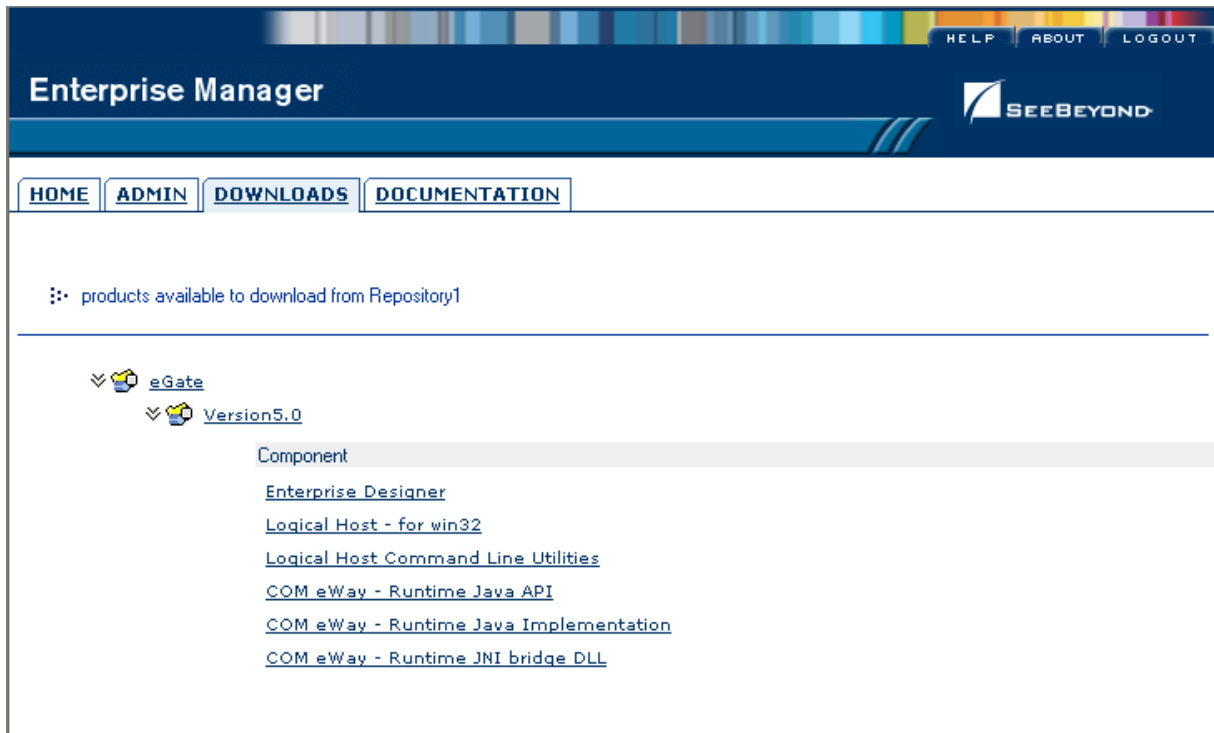
- Install the eGate Repository
- Upload products to the Repository
- Download components (including the eGate Enterprise Designer and Logical Host)

Follow the directions for installing the ICAN Suite in the *SeeBeyond ICAN Suite Installation Guide*. After you have installed eGate and any other purchased core products, do the following:

- 1 From the Enterprise Manager's **ADMIN** tab, browse to the **Add-ons** directory and select the **ProductsManifest.xml**, and click **Submit**. The available Add-on product list is now displayed.
- 2 Browse to and select the following files located in the **Add-ons** directory:
 - ♦ **FileeWay.sar** (to install the File eWay, used in the sample project)
 - ♦ **COMeWay.sar** (to install the COM/DCOM eWay)
- 3 Click on **upload now** to upload the selected products.
- 4 Click on the Manifest File field's **Browse** option, browse to the Add-on **Documentation** directory, select the **ProductsManifest.xml**, and click **Submit**. The available Add-on documentation list is now displayed.
- 5 Select and upload the following file:
 - ♦ **COMeWayDocs.sar** (to upload the COM/DCOM eWay User's Guide, Javadoc, and sample projects to the Enterprise Manager).
- 6 Click on **upload now** to upload the selected product.
- 7 Click on the Enterprise Manager's **DOWNLOADS** tab. The Component list, as displayed in **Figure 1 on page 12**, includes three COM eWay components:
 - ♦ **Runtime Java API** (stccomruntime.api.jar) and **Runtime Java Implementation** (stccomruntime.impl.jar) which must be copied to the Integration Server classpath.

- ◆ **Runtime JNI bridge DLL** (comewayruntime.dll) which must be copied to the Integration Server Library path (see Figure 1).

Figure 1 Enterprise Manager - DOWNLOADS



- 8 Download all three components to a temporary directory. To add the .jar files to the Integration Server classpath see [Adding JAR files to the Integration Server Classpath](#) on page 13. To add the .dll file to the Library classpath see [Adding DLL files to the Integration Server Library Path](#) on page 13
- 9 Continue installing the eGate Integrator as instructed in the *SeeBeyond ICAN Suite Installation Guide*.

Adding the eWay to an Existing ICAN Suite Installation

If you are installing the eWay to an existing ICAN installation, do the following:

- 1 Complete steps 1 through 8 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. 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. Click **Finish**.
- 7 When prompted, restart the IDE to complete the installation.

2.3.2. Adding JAR files to the Integration Server Classpath

The **Runtime Java API** (stccomruntime.api.jar) and **Runtime Java Implementation** (stccomruntime.impl.jar) can be added to the Integration Server classpath after you have created an **Environment** and added a **Logical Host** for your project. To add the .jar files to the Integration Server classpath, do the following:

- 1 From the Environment Explorer tree, right-click the **Logical Host** and select **Upload File** from the shortcut menu. The Upload Third Party Files dialog box appears.
- 2 Click **Add** and browse to the temporary directory to which you downloaded the .jar files. Add both **stccomruntime.api.jar** and **stccomruntime.impl.jar** to the **Third Party Files** field (see Figure 2).

Figure 2 Upload Third Party Files



- 3 Click **OK**.

2.3.3. Adding DLL files to the Integration Server Library Path

To add the **Runtime JNI bridge DLL** (comewayruntime.dll) to the Library path, do the following:

- 1 Copy the **comewayruntime.dll** file from the temporary directory to which you downloaded the file.
- 2 Paste the file to the following location:

<ican50>\logicalhost\stcis\lib

where <ican50> is your installed ICAN Suite.

After Installation

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

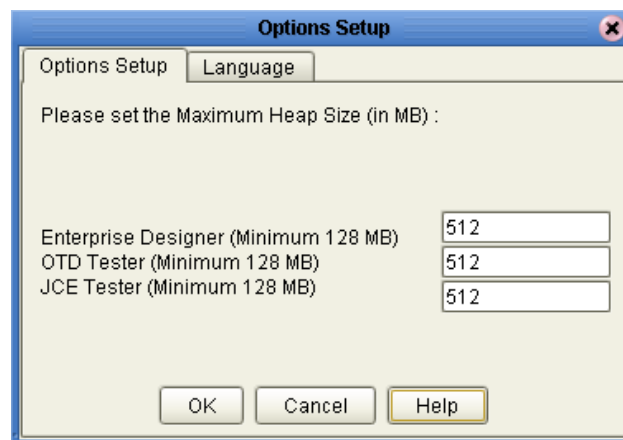
The eWay's User Guide, Javadoc, Readme, and sample projects, can be accessed from the Enterprise Manager's Documentation tab.

2.3.4. Increasing the ICAN Enterprise Designer Heap Size

Due to the possible size of the generated COM OTDs, the Enterprise Designer **Heap Size** may need to be increased prior to using eGate with the COM/DCOM eWay. If the heap size is not increased it may result in a **OutOfMemoryError** message. To increase the heap size from the Enterprise Designer do the following:

- 1 From the Enterprise Designer Menu bar click **Tools** and select **Options**. The **Options Setup** dialog box appears.
- 2 Make sure that the heap size is set to 512 MB or more. If not, increase the configured heap size for the Enterprise Designer, OTDTester, and JCE Tester to 512 MB as displayed in Figure 3, and click **OK**.

Figure 3 Options Setup - Heap Size



- 3 Close and restart the Enterprise Designer to allow your changes to take effect.

If an **OutOfMemoryError** message occurs while trying to open the Enterprise Designer, the heap size settings may be changed prior to starting the Enterprise Designer. In this case change the settings in the **heapSize.bat** file, located in `<ICAN_Home>\edesigner\bin` (where `<ICAN_Home>` is the ICAN install directory). Open **heapSize.bat** with a text editor and change the heap size settings to **512**. Save the file, and restart the Enterprise Designer.

Configuring the eWay Properties

This chapter describes how to create and configure the COM/DCOM eWay properties.

What's in This Document

- [Configuring the COM/DCOM eWay](#) on page 15
- [Using the Properties Editor](#) on page 16
- [COM/DCOM eWay Properties](#) on page 17
- [Alerting and Logging](#) on page 18

3.1 Configuring the COM/DCOM eWay

All eWays contain a set of properties unique to that eWay type. After the component eWays are created and a COM/DCOM External System is created in the project's Environment, the eWay properties can be modified for your specific system. The COM/DCOM eWay's properties are modified only 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 COM/DCOM External System.

3.1.1 Selecting COM/DCOM as the External Application

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

Modifying the COM/DCOM eWay (Environment Explorer) Properties

Once an Environment has been created and a new COM External System has been added, the COM/DCOM Environment properties can be modified.

- 1 From the Environment Explorer tree, right-click the COM/DCOM external system and select **Properties** from the shortcut menu. The **Properties Editor** appears.
- 2 Make any necessary modifications to the Environment parameters of the COM/DCOM eWay and click **OK** to save the settings.

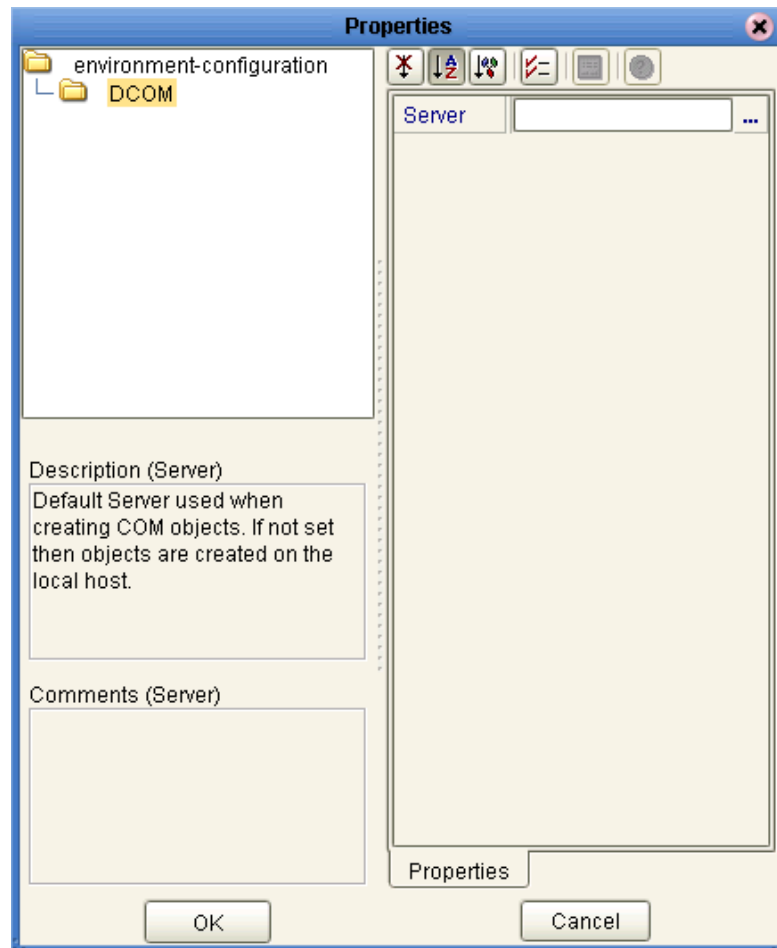
3.1.2. Using the Properties Editor

Modifications to the eWay configuration properties are made from the COM/DCOM eWay Properties Editor.

To modify the default eWay configuration properties

- 1 From the upper-left pane of the Properties Editor, select a subdirectory of the configuration directory. The parameters contained in that subdirectory are now displayed in the Properties pane of the Properties Editor. For example, clicking on the **DCOM** subdirectory displays the editable parameters in the right pane, as shown in Figure 4.

Figure 4 Properties Editor - COM/DCOM eWay Properties



- 2 The COM/DCOM eWay properties contain only one configurable parameter: **Server**. To edit this property, click on the property field.

To open a separate configuration dialog box, click on the ellipsis (. . .) in the properties field. This is helpful for large values that cannot be fully displayed in the parameter's property field. Enter the property value in the dialog box and click **OK**. The value is now displayed in the parameter's property field.

- 3 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.
- 4 The **Comments** pane provides an area for recording notes and information regarding the currently selected parameter. This is saved for future referral.
- 5 After modifying the configuration properties, click **OK** to close the Properties Editor and save the changes.

3.2 COM/DCOM eWay Properties

The COM/DCOM eWay contains only one configurable property: **Server**, an Environment property located under the DCOM subnode.

Server

Description

Specifies the default server used when creating an instance of a DCOM component. (that is, a remote server executable). This property is not required when using an in-process component (for example, a .dll)

Required Values

The name of the server on which the DCOM component is to be created. If the name is not specified, then objects are created on the local host.

Note: *This property can also be configured dynamically from the Collaboration. See [Dynamic Configuration](#) on page 17 for more information.*

3.3 Dynamic Configuration

The **Server** property can also be dynamically configured from the Collaboration at runtime. This automatically supersedes the current Environment property.

Dynamic configuration allows the user to change configuration settings (based on the data input or Collaboration Rule logic) on the fly. In this case, the **Server** property can be changed by specifying a different server in the Collaboration using the Collaboration Editor. From the point that you reset the Server property, all other DCOM objects will be created on the new server.

3.4 Alerting and Logging

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

The alerts/status notifications for the COM/DCOM eWay are currently limited to Started, Running, Stopping, and Stopped.

COM/DCOM eWay OTD Overview

This chapter provides a short overview of the OTDS created from the COM components Type Library files by the COM OTD Wizard. This chapter also describes how to use the COM OTD Wizard to build an OTD.

What's in This Document

- [Object Type Definitions](#) on page 19
- [The COM OTDs](#) on page 19
- [Using the COM OTD Wizard](#) on page 22

4.1 Object Type Definitions

The basic functionality of eGate is to handle messages. It does this by means of Collaborations. To operate on a message, the Collaboration needs a description of the message format. The format description may follow a standard and be available in some standard metadata format, or it may not be, in which case, you need a convenient way to define the format. Object Type Definitions provide the solution.

OTDs describe external data formats that characterize the input and output data structures in a Collaboration Definition. OTDs typically have a specific external representation format used to store and transport the OTD content through an eGate Project. OTDs define both this external representation and the run-time data structures.

4.1.1 The COM OTDs

The COM OTD Wizard scans the Type Library for creatable coclasses. For every creatable coclass it finds, it creates a factory method named `create[CoClass_name]()`. These factory methods allow you to create instances of COM objects defined by those CoClasses. Each returns a Java interface, which is also generated by the OTD wizard. That interface contains the methods exposed on the corresponding COM interface. These methods can then be called like any normal Java method.

Interface instances can be acquired in a Collaboration in a number of ways:

- Through **Create methods** on the OTD
- Through **Query methods** on the OTD
- Through a **Return value**

- Through an **Output parameter**
- Through an **Input/Output parameter**

Regardless of how it is acquired, each of these acquired interfaces must be released after an application is finished using it. The **Release** method must be called to clear the local interface handle and remove the COM object from the server memory.

Create Methods

The Create method allows the user to create an instance of a CoClass.

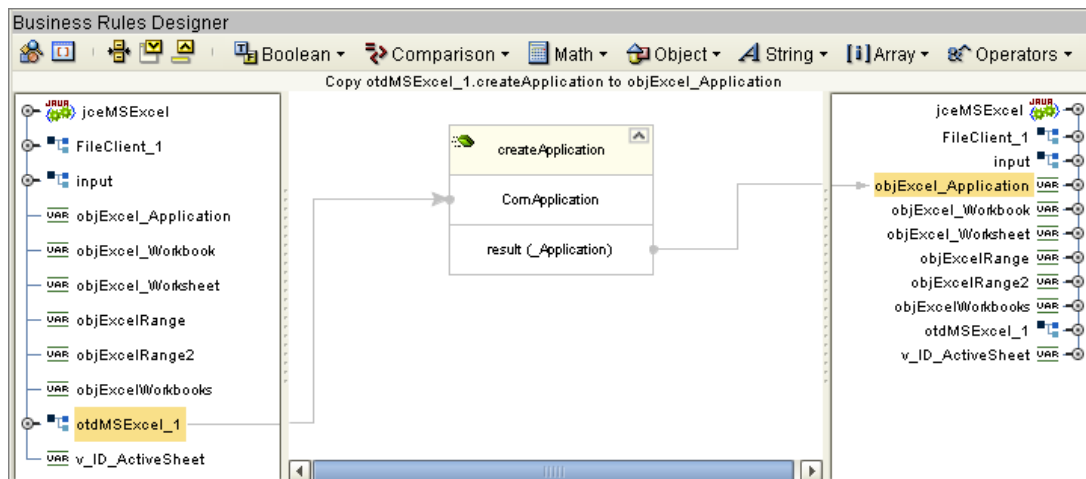
In COM, when you create an instance of a COM object, you get back is an interface for that object. The Create methods create instances of COM objects. When the OTD wizard scans the selected Type Library, it displays the Library’s CoClasses. For each CoClass you select in the wizard (see [Select Classes](#) on page 23), a Create method is created on the OTD. These Create methods instantiate the CoClass.

Example of Create Method

An example of a Create method used in a Collaboration can be seen in the **COMeWay_MSEExcel** sample project’s **jceMSEExcel** Collaboration.

- 1 In the Business Rule, **Copy otdMSEExcel_1.createApplication to objExcel_Application**, the user calls the **createApplication()** method to create an instance of the Excel application (see Figure 5).

Figure 5 Copy otdMSEExcel_1.createApplication to objExcel_Application Business Rule



- 2 From this instance interface, the user is now able to call various methods to access the functionality of the application (for example, the next four Business Rules in the Collaboration copy the application Name, Version to the output file).

Query Methods

Query methods allow you to convert a particular interface to another interface.

In some cases, COM servers declare their parameters to be simply IDispatch, but document that this parameter will return a particular interface in the Type Library. For

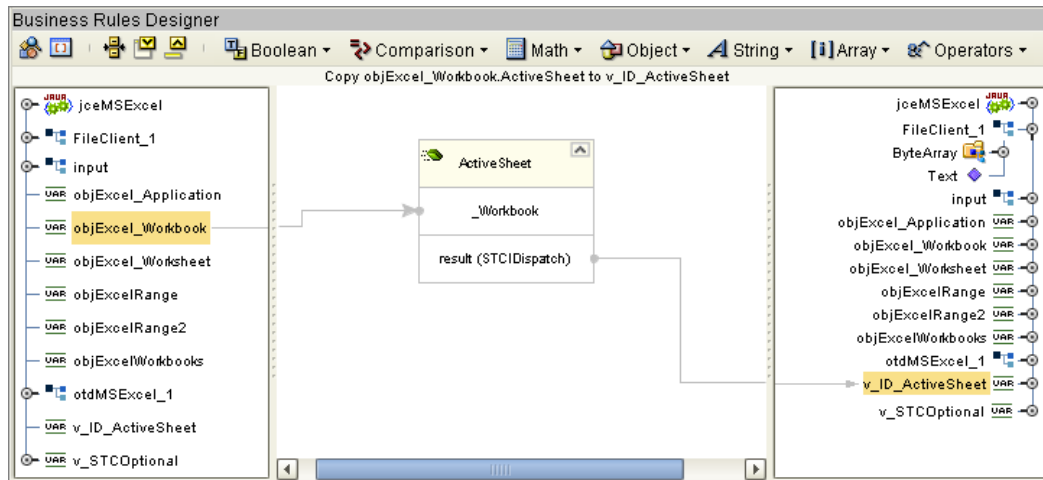
each interface found in the Type Library, a Query method is created on the OTD. You can then pass one of the IDispatch interfaces and convert it to the appropriate interface. Query methods returns the appropriate interfaces with relevant, useful, Java methods.

Example of Query Method

An example of Query method used in a Collaboration can be seen in the COMeWay_MSExcel sample project's jceMSExcel Collaboration Business Rules.

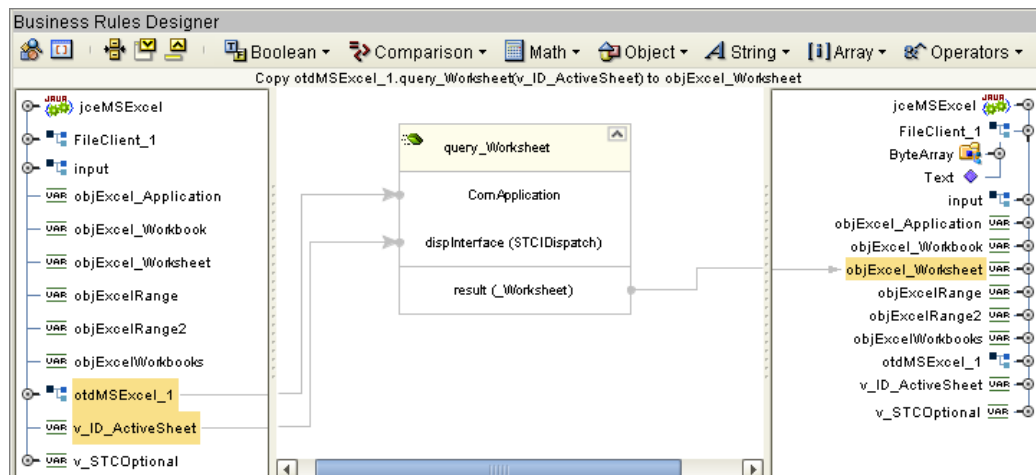
- 1 The user wants to select and copy an active Excel worksheet. In the Business Rule, **Copy objExcel_Workbook.ActiveSheet to v_ID_ActiveSheet**, When the objExcel_Workbook method, **ActiveSheet()** is called, the **STCIDispatch** interface is returned (as seen in Figure 6) providing limited functionality.

Figure 6 Copy objExcel_Workbook.ActiveSheet to v_ID_ActiveSheet Business Rule



- 2 To provide a more useful interface, the user selects the Query method, **query_Worksheet(com.stc.connector.comadapter.comruntime.STCIDispatch dispinterface)** from the **otdMSExcel_1** OTD in the next Business Rule, **Copy otdMSExcel_1.query_Worksheet(v_ID_ActiveSheet) to objExcel_Worksheet**. As a result, the **Worksheet** interface is returned (see [Figure 7 on page 21](#)).

Figure 7 Query Method



- 3 The Worksheet interface contains multiple methods that provide easy access to the COM object's functionality. For example, the user is able to call the **Range** method (in the **Copy objExcel_Worksheet.Range(v_position1, v_position2) to objExcelRange** Business Rule) to specify two cells in the active worksheet.

4.2 Using the COM OTD Wizard

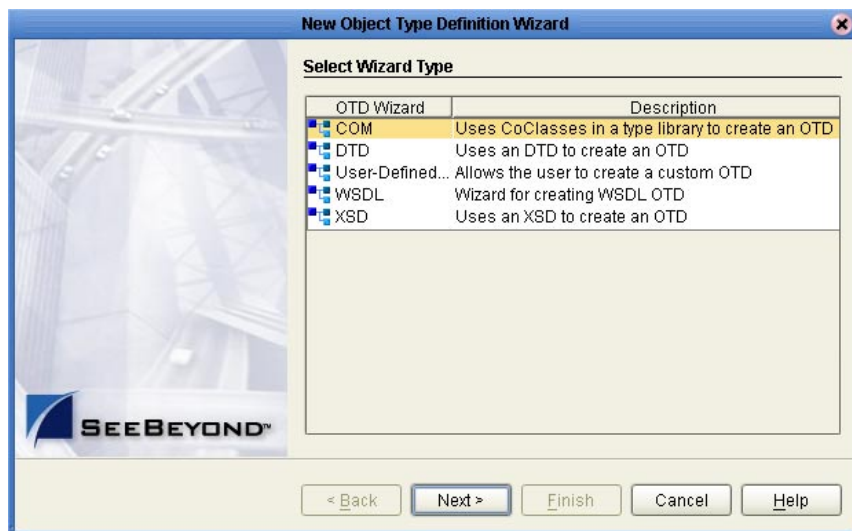
The COM OTD Wizard generates an OTD from a COM automation-compatible component's **Type Library** files. COM Type Library files describe the methods and properties exposed from an automation-compatible component. COM type libraries may have the file extension **.tlb** or **.olb**, however, most components typically embed the type library file in the **DLL**, **OCX**, or **EXE** file that contains the component.

To create an OTD using the COM OTD Wizard do the following:

Select Wizard Type

- 1 From the Project Explorer tree, right click your project and select **New > Object Type Definition** from the shortcut menu.
- 2 From the **Select Wizard Type** window of the **New Object Type Definition Wizard**, select the **COM** Wizard and click **Next**. See [Figure 8 on page 22](#).

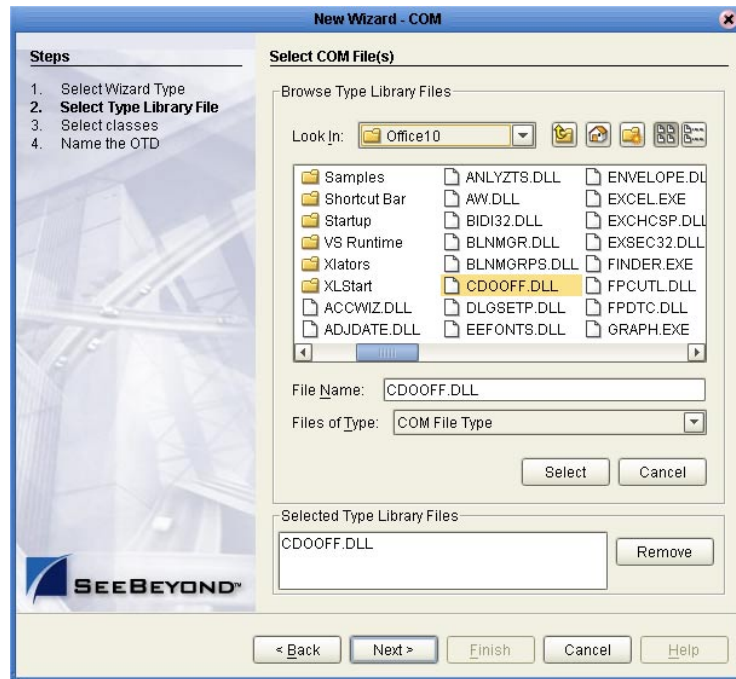
Figure 8 OTD Wizard Selection



Select Type Library File

- 3 Browse to the directory that contains the type library file from which the OTD will be created. You can only select one type library file at a time. Select your type library file and click the **Select** button (see Figure 9). Click **Next**.

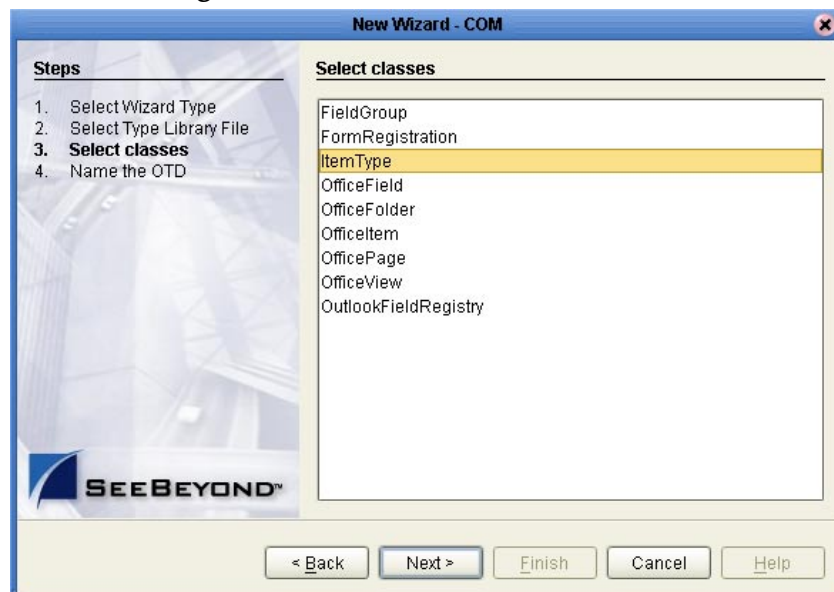
Figure 9 Select the Type Library



Select Classes

- 4 For Step 3 of the wizard, select one or more of the CoClasses from the type library and click **Next** (see Figure 10).

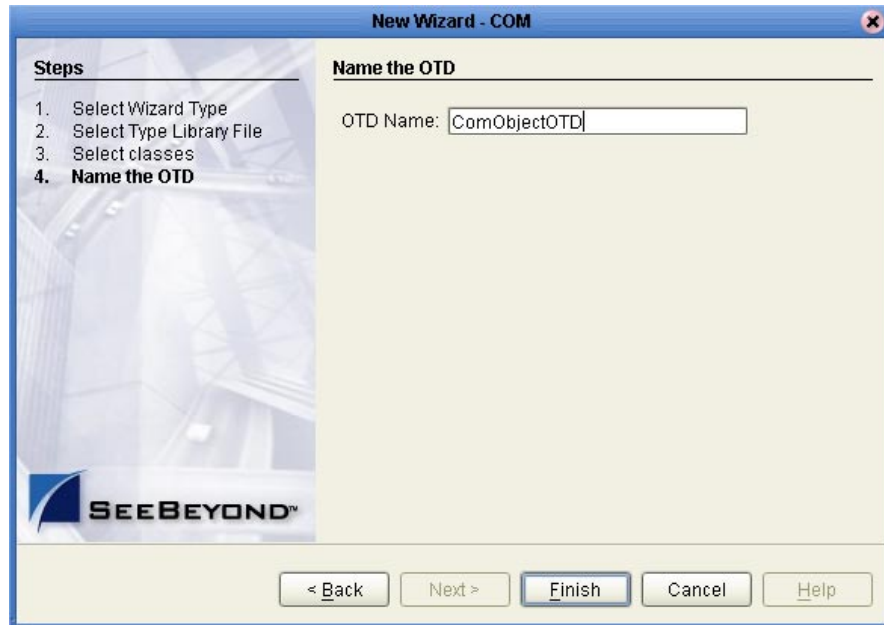
Figure 10 Select Classes



Name the OTD

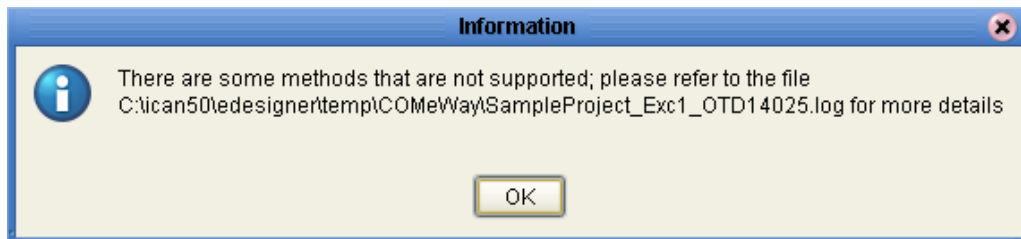
- 5 For Step 4 of the wizard, enter a name for the new OTD in the OTD Name field and click **Finish** (see Figure 11).

Figure 11 Name the OTD



- 6 If any of the selected CoClasses contain a method with an unsupported data type, an **Information** box appears (see Figure 12).

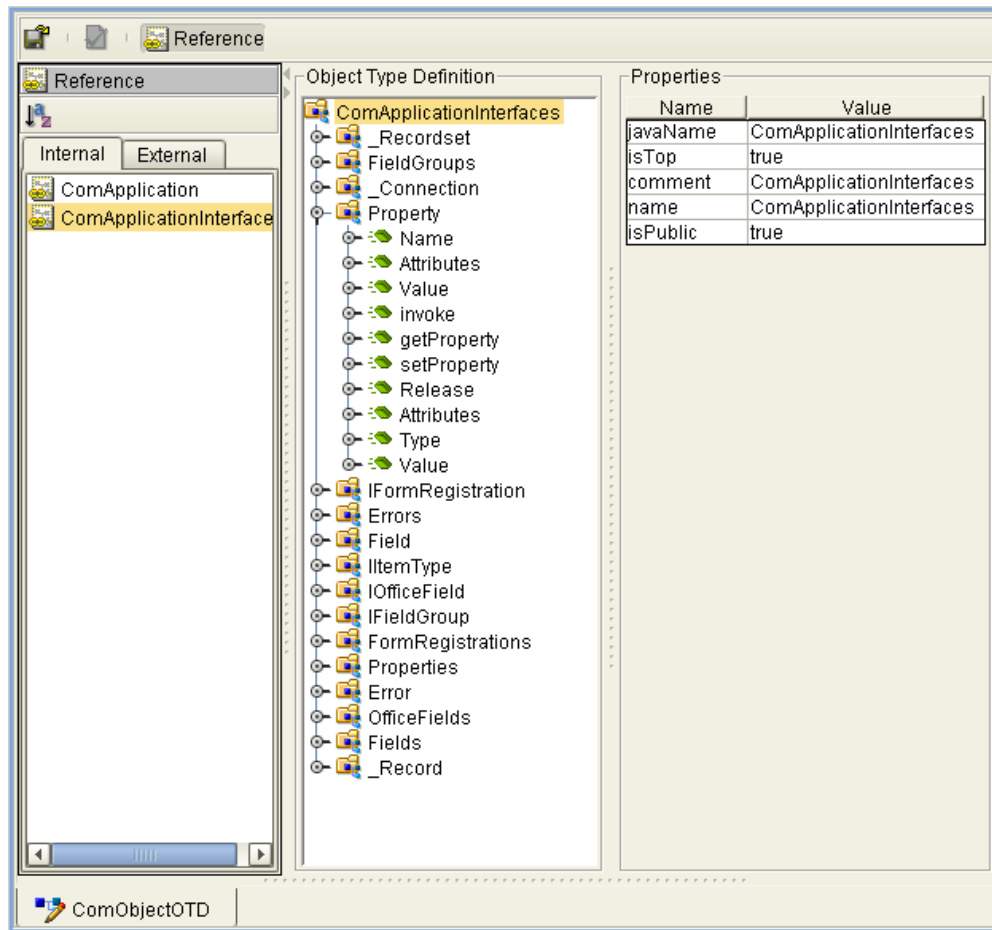
Figure 12 Information - Skipped Methods



The **Information** box indicates that some methods were not created in the OTD, and provides the location of the generated "Skipped Methods" log. This log provides a report of any methods that were skipped when the OTD was created (this information is also written to the IDE log file). Click **OK** to acknowledge and close the **Information** box.

- 7 The OTD Editor appears displaying the new OTD (see [Figure 13 on page 25](#)).

Figure 13 OTD Editor - New OTD



The resulting OTD is now available for use in your Collaborations.

For more information on using the OTD Editor see the *eGate Integrator User's Guide*.

Implementing a COM/DCOM eWay Project

This chapter provides an introduction to the COM/DCOM eWay components and information on 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 SeeBeyond Enterprise Designer. For more information on creating an eGate project see the *eGate Tutorial* and the *eGate Integrator User's Guide*.

What's in This Document

- [COM/DCOM eWay Components](#) on page 26
- [COM/DCOM eWay Considerations](#) on page 28
- [COM/DCOM eWay Sample Projects](#) on page 28
- [Importing a Sample Project](#) on page 30

5.1 COM/DCOM eWay Components

This chapter presents a sample COM/DCOM eWay project created using the same procedures as the sample end-to-end project provided in the *eGate Integrator Tutorial*. The eWay components that are unique to the COM/DCOM eWay include the following:

COM/DCOM eWay Configuration File

The properties file for the COM/DCOM eWay contains properties that are used to connect with a specific external system. These parameters are set using the Properties Editor. For more information about the COM/DCOM eWay properties File and the Properties Editor see [Configuring the COM/DCOM eWay](#) on page 15.

ComApplication OTD

The ComApplication OTD, provided with the eWay, is for advanced use only. It allows for direct interaction with the COM Server without the need to create an OTD. This OTD's functionality is readily available from any of the COM builder based OTDs generated by the COM OTD Wizard. For more information on how to use the ComApplication OTD see the COM/DCOM eWay Javadoc.

COM OTD Wizard

The COM OTD Wizard builds an Object Type Definition (OTD) for COM object classes in a type library (see [COM/DCOM eWay OTD Overview](#) on page 19).

5.2 Supported Data Types

The COM/DCOM eWay supports the following data types:

Table 2 Supported Data Types

VARTYPE	Description	1 and 2 Dimension SAFEARRAY Support
VT_I2 (short)	2 byte signed integer	Yes - 1 and 2
VT_I4 (long) (See note below)	4 byte signed integer	Yes - 1 and 2
VT_R4 (float)	4 byte float	Yes - 1 and 2
VT_R8 (double)	8 byte double	Yes - 1 and 2
VT_DATE (Date)	Standard COM date type	Yes - 1 and 2
VT_BSTR (BSTR)	COM string type	Yes - 1 and 2
VT_BOOL (Boolean)	COM boolean type	Yes - 1 and 2
VT_DISPATCH	COM automation interface	Yes - 1
VT_VARIANT (VARIANT)	COM VARIANT type (which is a tagged union)	No
VT_USERDEFINED	Typedef enum myenum	Yes - 1
VT_USERDEFINED	dispinterface Typename	Yes - 1
VT_USERDEFINED	CoClass Typename	Yes - 1

Note: In COM, at the C/C++ level, VT_I4 is a **long** integer type. For Win32 platforms, this is 4 bytes - unsigned. The Java mapping for a 4 byte integer is type *int*.

5.2.1. SAFEARRAY Constraints

Zero-Based Indexes

Although COM technically allows SAFEARRAYs to use starting indexes other than zero, the COM/DCOM eWay supports zero-based indexes only.

2 Dimensional SAFEARRAYs

Internally, the eWay runtime supports 1 or 2 dimensional SAFEARRAYs. The eWay runtime only supports two dimensional arrays with equal array size in both dimensions. The builder, however, cannot store information about the array dimension in the OTD because the metadata is not available in the type library: that is, the type library relates that the parameter is a SAFEARRAY, but does not relate how many dimensions the array is.

The STCComSafeArray class was created to hold and exchange arrays with the methods of generated COM OTDs that have SAFEARRAY parameters. When using SAFEARRAY parameters, the user must consult the documentation for the component, then ensure that the STCComSafeArray class is created with the type and array dimensions that the component expects.

5.3 COM/DCOM eWay Considerations

The following items must be considered when implementing a COM/DCOM eWay project:

- After an application is finished using an acquired COM object, the **Release()** method must be called to clear the local interface handle and remove the COM object from the server memory.
- The COM/DCOM eWay supports outbound connection only.
- Method names used for a created custom COM object must not match names of final methods on the Java Object class (for example, the **wait()**, **notify()**, and so forth). If the COM object contains these methods it will cause a compile error.
- Characters used in COM Type Library names that are not acceptable in Java (such as dots ".") in the Java namespace will be translated as an underscore. For example: **com.seagull.fileInquiry** is translated as **com_seagull_fileInquiry**.
- The COM/DCOM eWay is a "JCE (Java Collaboration Editor) only" eWay, and does not directly support Business Process Execution Language (BPEL) for Web Services or SeeBeyond's eInsight Manager. To use the COM/DCOM eWay with eInsight requires the eInsight Business Process to first invoke a Java Collaboration. That Java Collaboration can then execute a call to the COM/DCOM eWay. For more information on calling a Java Collaboration from an eInsight Business Process, see the *eInsight Business Process Manager User's Guide*.
- **STCComVariant.getVartype()** returns the VARTYPE of the stored value and is one of the public constants defined in STCComVARTYPE.

5.4 COM/DCOM eWay Sample Projects

Three COM/DCOM eWay sample projects are included on the installation CD-ROM, that demonstrate how the eWay is implemented in a production environment.

- COMeWay_MSWord_Sample.zip
- COMeWay_MSExcel_Sample.zip
- COMeWay_MSDAO_Sample.zip

These sample projects, when imported to your Repository, are nearly complete. They require only that you create the Environments, Deployment Profiles, and configure the eWay properties for your system.

5.4.1. The COMeWay_MSWord Sample Project

The **COMeWay_MSWord** sample project demonstrates the following:

- 1 An input file triggers the eWay to connect to the COM server (If a COM server is not specified, the eWay uses your local host as the Com server).

- 2 The eWay searches the server for a specified application with a specific version number (in this case MS Word, version 10).
- 3 The eWay then creates a new Word document and adds the specified text to the document.
- 4 The eWay then saves the new document to the location specified in the input file.
- 5 The eWay writes the application name and version to the output file and publishes the file to an external directory.

The **COMeWay_MSWord** sample file includes:

- **COMeWay_MSWord_SampleProject.zip**: The sample file that is imported into the Repository.
- **input_MSWord_SaveAsFileName.~in**: The input (trigger) file. This is a text file that contains the location and file name where the testing doc will be saved.
- **MSWord_output1.dat**: The output file with the COM application name and version.
- **output_MSWordTesting.doc**: The testing doc.

For directions on creating the **COMeWay_MSWord** sample manually, see [Creating the COMeWay_MSWord Project](#) on page 31.

5.4.2. The **COMeWay_MSExcel** Sample Project

The **COMeWay_MSExcel** sample project demonstrates the following:

- 1 An input file triggers the eWay to connect to the COM server (If a COM server is not specified, the eWay uses your local host as the Com server). This input file contains the location and name of the Excel file.
- 2 The eWay searches the server for a specified Excel file and opens the file.
- 3 After 2 seconds, the eWay populates the message, "Hello World!", into a range of two cells in the Excel file.
- 4 The eWay copies the value of a cell in the Excel file and writes the value to an output file.
- 5 The eWay closes the Excel file, without Saving, and publishes the output file to an external directory.

The **COMeWay_MSExcel** sample file includes:

- **COMeWay_MSExcel_SampleProject.zip**: The sample file that is imported into the Repository.
- **COMeWay_ExcelTest.xls** -The sample Excel file.
- **in_Excel.~in**: The sample input file.
- **MSExcel_output1.dat**: The sample output file.

For directions on creating the **COMeWay_MSWord** sample manually, see [Creating the COMeWay_MSExcel Sample Project](#) on page 58.

5.4.3. The COMeWay_MSDAO Sample Project

The COMeWay_MSDAO (Data Access Object) sample project demonstrates the following:

- 1 An input file triggers the eWay to connect to the COM server (If a COM server is not specified, the eWay uses your local host as the Com server). This input file contains the location and name of an Access MDB file.
- 2 The eWay searches the server for the specified Access MDB file and opens the file.
- 3 The eWay copies all rows of data from the specified recordset and writes it to an output file.
- 4 The eWay publishes the output file to an external directory.

The COMeWay_MSDAO sample file includes:

- **COMeWay_MSDAO_TestProject.zip**: The sample file that is imported into the Repository.
- **DAOTestDB.mdb** -The sample MDB file.
- **in_DAO.~in**: The sample input file.
- **MSDAO_output1.dat**: The sample output file.

For directions on implementing the COMeWay_MSDAO sample, see [Implementing the COMeWay_MSDAO Sample Project](#) on page 68.

5.5 Importing a Sample Project

To import the 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 Enterprise Manager's Documentation tab. Extract the samples from the Enterprise Manager to a local file. This file should include three .zip files. Extract each of these for the sample .z and any additional sample files.
- 2 Save all unsaved work before importing a sample 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 example, **COMeWay_MSWord_SampleProject.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 53)
 - ♦ Configure the eWays (see [Configuring the eWays](#) on page 54)
 - ♦ Create a **Deployment Profile** (see [Creating and Activating the Deployment Profile](#) on page 56)

5.6 Creating the COMeWay_MSWord Project

The sample projects provided with this eWay require very little setup. The following sections in this chapter are provided to demonstrate how the COMeWay_MSWord sample project's components are created.

5.6.1. Create a Project

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

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

5.6.2 Create a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a project's components.

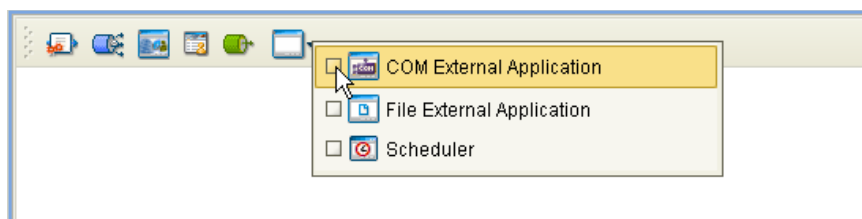
- 1 In Enterprise Explorer's Project Explorer, right-click the new project and select **New > New Connectivity Map** from the shortcut menu.
- 2 The New Connectivity Map appears and a node for the Connectivity Map is added to the Project Explorer tree labeled **CMap1**.

5.6.3. Select the External Applications

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

When creating a Connectivity Map, the eWays are associated with external systems. For example, to establish a connection to an external COM server, you must first select COM/DCOM as an External Application to use in your Connectivity Map (see Figure 14). The COM/DCOM External Application icon is then added to the Connectivity Map toolbar.

Figure 14 Connectivity Map - External Applications



To add the External Applications used with the COMeWay_MSWord project, do the following:

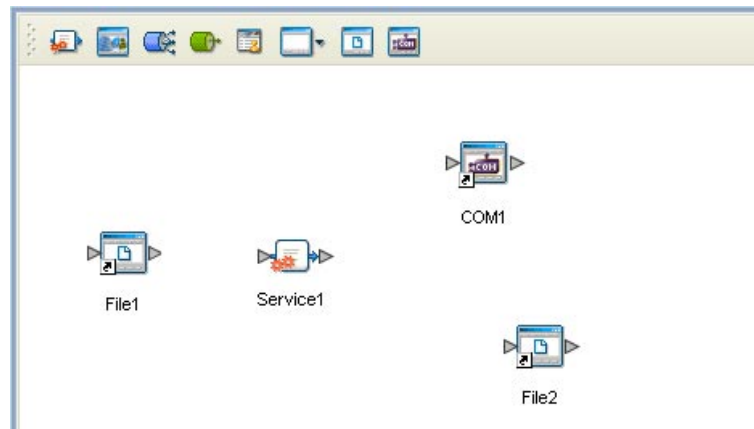
- 1 Click the **External Application** icon on the Connectivity Map toolbar.
- 2 Select the external systems that are necessary for your project (for this sample, **COM/DCOM** and **File**). Icons representing the selected external systems are added to the Connectivity Map toolbar.

5.6.4. Populate the Connectivity Map

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

- 1 For this sample, drag the following components onto the Connectivity Map canvas as displayed in Figure 15:
 - ♦ File External System (2)
 - ♦ Service (A Service is a container for Java Collaborations, Business Processes, eTL processes, and so forth)
 - ♦ COM/DCOM External System

Figure 15 Connectivity Map with Components



- 2 Save your current changes.

5.6.5. Creating the otdMSWord OTD

The COM OTD Wizard generates an OTD from a COM automation-compatible component's **type library**. The COM Type Library file used for this sample is the **MSWORD.OLB** file from Microsoft Word 2002.

To create an OTD using the COM OTD Wizard do the following:

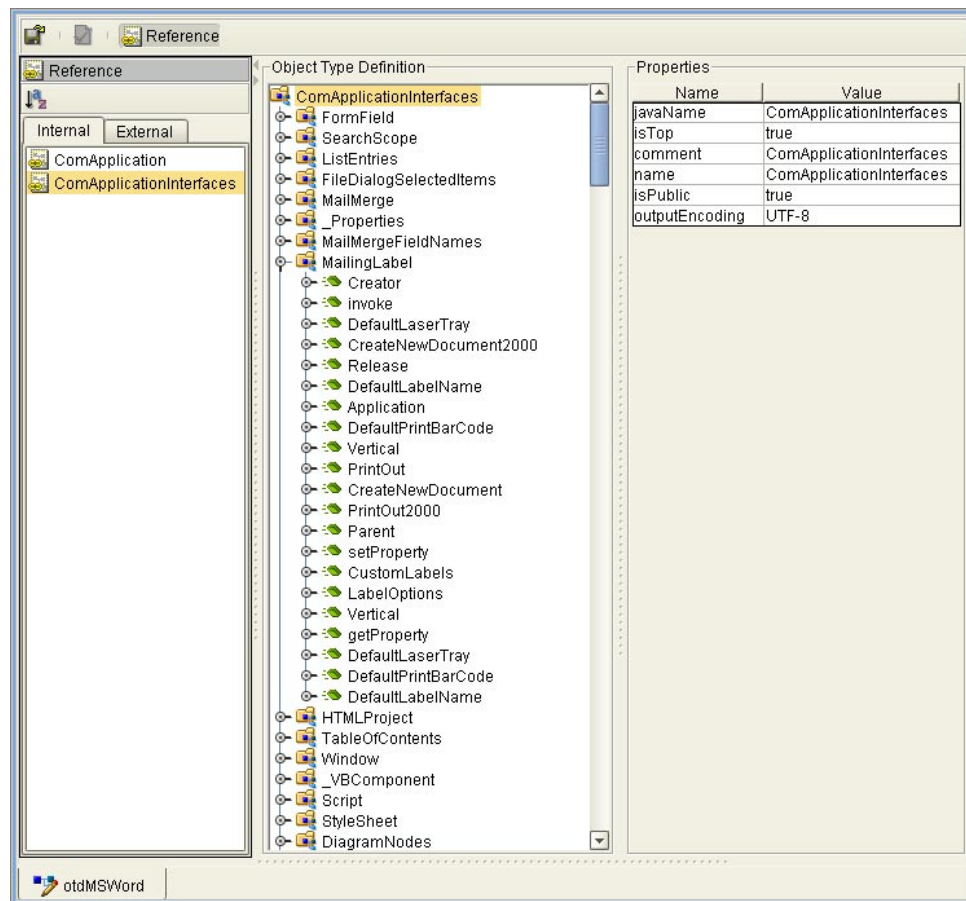
- 1 From the Project Explorer tree, right click **COMeWay_MSWord** and select **New > Object Type Definition** from the shortcut menu.
- 2 From the **Select Wizard Type** window of the **New Object Type Definition Wizard**, select the **COM Wizard** and click **Next**.

- 3 Browse to the directory that contains the type library file, MSWORD.olb. For example:

```
C:\Program Files\Microsoft Office\Office10
```

Select the **MSWORD.olb** file and click the **Select** button, and click **Next**.
- 4 For **Step 3** of the wizard, select the classes you want from the type library (for this sample select all of the classes) and click **Next**.
- 5 For Step 4 of the wizard, enter **otdMSWord** as the name of the OTD, and click **Finish**.
- 6 An **Information** box is displayed indicating that some methods are not supported by the eWay. These methods will be skipped in the created OTD. The **Information** box also gives you the location of a generated “Skipped Methods” log that lists the methods skipped, as well as the issues that caused the method to be excluded.
- 7 The OTD Editor appears displaying the new OTD (see Figure 16).

Figure 16 OTD Editor - otdMSWord OTD



The new **otdMSWord** OTD is added to the Project Explorer tree. The OTD is now available to use in a Collaboration.

For more information on using the COM OTD Wizard, see [Using the COM OTD Wizard](#) on page 22.

5.6.6. Creating the Collaboration Definitions

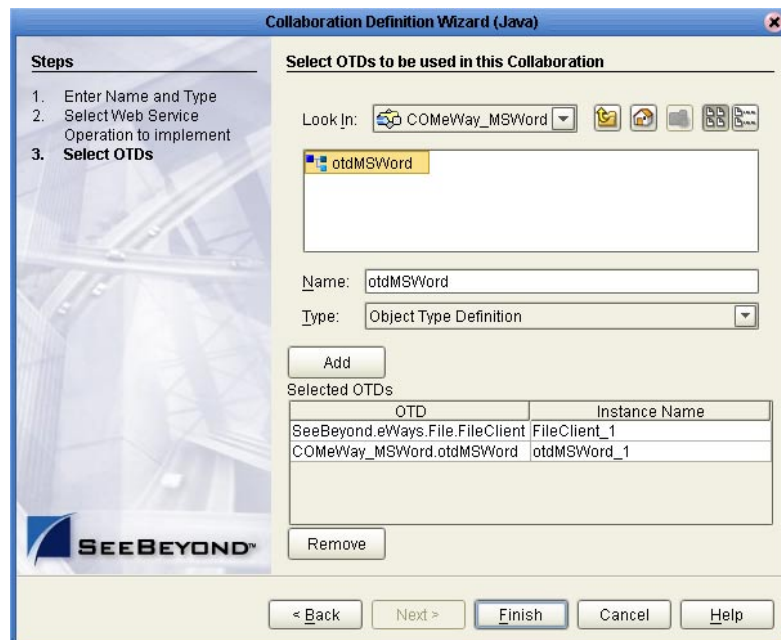
The next step in the sample is to create a Collaboration using the Collaboration Definition Wizard (Java). Once a Collaboration Definition has been created, the Business Rules of the Collaboration can be written using the Collaboration Editor (Java).

The jceMSWord Collaboration (Java)

The **jceMSWord** Collaboration defines transactions from the inbound File eWay to the COM/DCOM eWay and from the COM/DCOM application to the outbound File eWay.

- 1 From the Project Explorer, right-click the **COMeWay_MSWord** Sample project and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
- 2 Enter a Collaboration Definition name (for this sample, **jceMSWord**) and click Next.
- 3 For Step 2 of the wizard, **Select a Web Services Operation**, double-click **SeeBeyond > eWays > File > FileClient > receive** to select the File eWay receive Web service. Click Next.
- 4 For Step 3 of the wizard, from the Select OTDs selection window, double-click **SeeBeyond > eWays > File > FileClient**. The **FileClient** OTD is added to the Selected OTDs field.
- 5 Click the **Up One Level** button to return to the Repository. Double-click **COMeWay_MSWord > otdMSWord**. The **otdMSWord** OTD is added to the Selected OTDs field (see [Figure 17 on page 34](#)).

Figure 17 Collaboration Definition Wizard (Java) - Select OTDs



- 6 Click **Finish**. The Collaboration Editor (Java) appears in the left pane of the Enterprise Designer and the **jceMSWord** Collaboration is added to the Project Explorer tree.

5.6.7. Using the Collaboration Editor (Java)

The next step in the sample is to create the Business Rules of the Collaboration using the Collaboration Editor. The the COMeWay_MSWord sample project uses one Collaboration created in the previous section, **jceMSWord**.

The **jceMSWord** Collaboration contains the Business Rules displayed in [Figure 18 on page 35](#).

Figure 18 jceMSWord Collaboration Business Rules

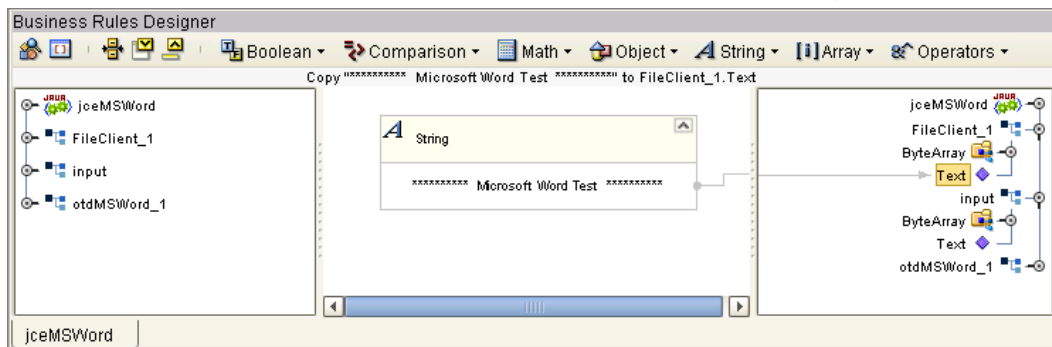


Note: Business Rules are wrapped in Figure 18 for display purposes only.

To create the **jceMSWord** Collaboration Business Rules, do the following:

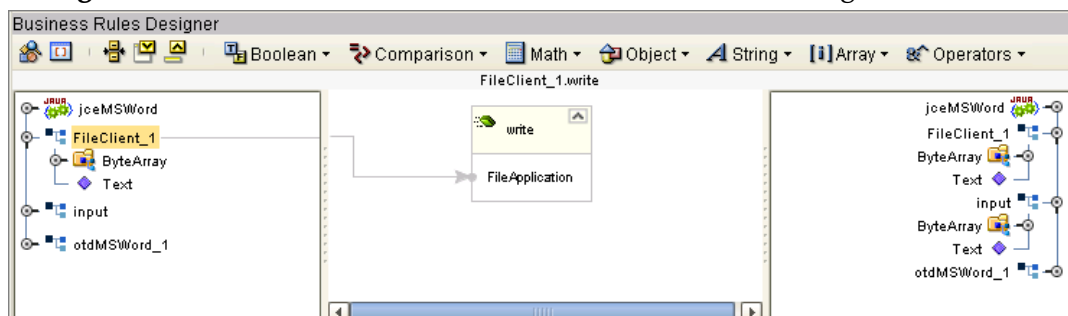
- 1 From the Project Explorer tree, double-click **jceMSWord** to open the Collaboration Editor (Java) to the **jceMSWord** Collaboration.
- 2 The **Copy "***** Microsoft Word Test *****" to FileClient_1.Text** rule, along with the **FileClient_1.write** rule, adds the leading "test" note to the output file. To create this rule, do the following:
 - A From the Business Rules Designer String menu, click **Literal String**. The **String** method box appears. Enter ******* Microsoft Word Test ******* as the value.
 - B Map the ******* Microsoft Word Test ******* output node of the **Literal** method box to **Text**, under the **FileClient1** node in the right pane of the Business Rules Designer. To do this, click on the ******* Microsoft Word Test ******* output node of the **Literal** method box and drag the cursor to **Text**, under the **FileClient1** node in the right pane of the Business Rules Designer (see Figure 19).

Figure 19 Collaboration Editor (Java) - Business Rules Designer



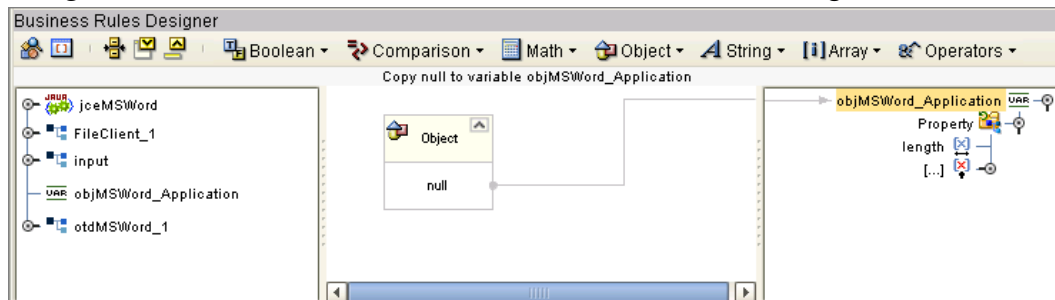
- 3 To create the **FileClient1_Write** rule do the following:
 - A From the Business Rules toolbar, click rule to add a new rule to the Business Rule tree.
 - B Right-click **FileClient1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.
 - C Select **write()** from the method selection window. The **write** method box appears in the Business Rules Designer canvas (see Figure 20).

Figure 20 Collaboration Editor (Java) - Business Rules Designer



- 4 The **Copy null to variable objMSWord_Application** rule, is the first of three declared COM variables created in the **Try** block. These COM objects, placed before the **Try** block, are released by the three “release” rules in the **finally** block. To create this rule, do the following:
 - A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.
 - B Enter **objMSWord_Application** as the variable name.
 - C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears.
 - D Select **_Application** under **All Classes**, and click **OK**.
 - E Click **OK** to close the **Create Variable** dialog box and create the variable.
 - F From the Business Rules Designer Object menu, select **Null**. The **Object** method box appears with **null** as the value.
 - G Map the **null** output node of the **Object** method box to the **objMSWord_Application** field in the right pane of the Business Rules Designer (see Figure 21).

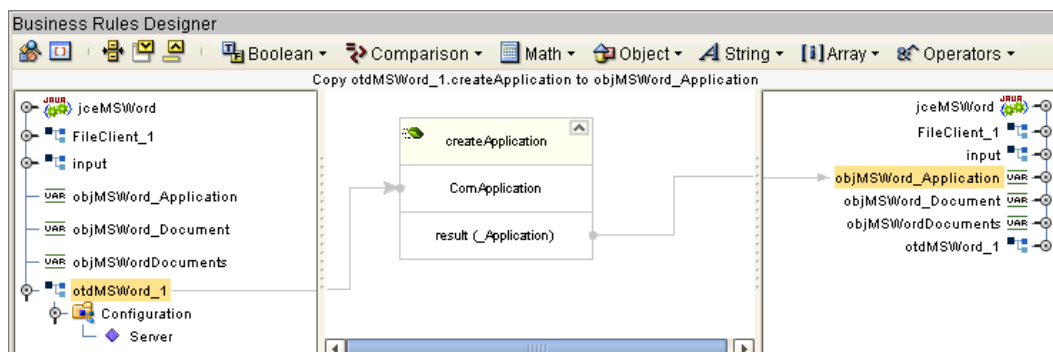
Figure 21 Collaboration Editor (Java) - Business Rules Designer



- 5 To create the **Copy null to variable objMSWordDocuments** rule do the following:
 - A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.
 - B Enter **objMSWordDocuments** as the variable name.
 - C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears.
 - D Select **Documents** under **All Classes**, and click **OK**.
 - E Click **OK** to close the **Create Variable** dialog box and create the variable.
 - F From the Business Rules Designer Object menu, click **Null**. The **Object** method box appears.
 - G Map the **null** output node of the **Object** method box to the **objMSWordDocuments** field in the right pane of the Business Rules Designer.
- 6 To create the **Copy null to variable objMSWord_Document** rule do the following:
 - A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.

- B Enter **objMSWord_Document** as the variable name.
 - C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears.
 - D Select **_Document** under **All Classes**, and click **OK**.
 - E Click **OK** to close the **Create Variable** dialog box and create the variable.
 - F From the Business Rules Designer Object menu, select **Null**. The **Object** method box appears.
 - G The **Literal** method box appears. Map the **null** output node of the **Object** method box to the **objMSWord_Document** field in the right pane of the Business Rules Designer.
- 7 The **Try** block is used to create an exception handling mechanism. The **Try** block encloses sections of code that may throw exceptions and provides a **catch** clause that catches and handles the exceptions. To create the **Try** block, do the following:
- A From the Business Rules toolbar, click **Try**. a Try node is added to the Business Rules tree.
- 8 The **Copy otdMSWord_1.createApplication to objMSWord_Application** rule creates an application object. To create this rule, do the following:
- A From the Business Rules tree, select **rules** under the **Try** block and click **rule** on the Business Rules toolbar to add a new rule.
 - B Right-click **otdMSWord_1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.
 - C Select **createApplication()** from the method selection window. The **createApplication** method box appears in the Business Rules Designer canvas.
 - D Map the **result (_Application)** output node of the **createApplication** method box to the **objMSWord_Application** field in the right pane of the Business Rules Designer (see Figure 22).

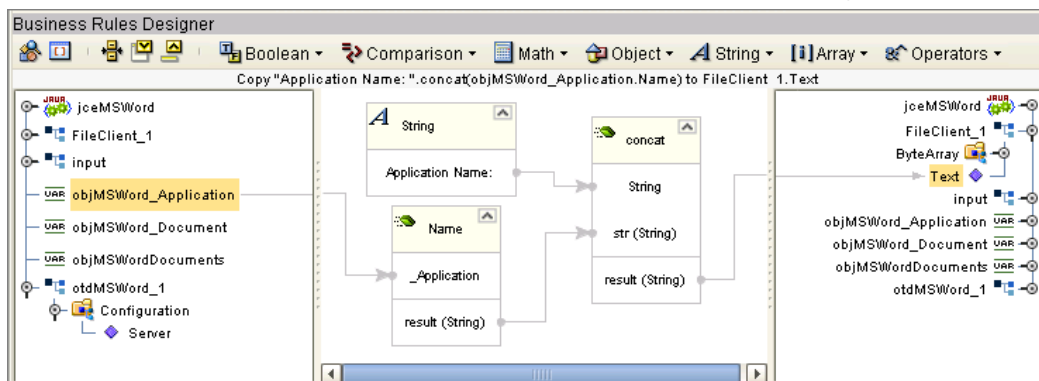
Figure 22 Collaboration Editor (Java) - Business Rules Designer



- 9 The **Copy "Application Name: ".concat(objMSWord_Application.Name) to FileClient_1.Text** rule, along with the **FileClient_1.write** rule, provides the application name and writes it to the output file. To create this rule, do the following:

- A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
- B From the Business Rules Designer String menu, click **Literal String**. The **String** method box appears. Enter **Application Name:** as the value.
- C From the Business Rules Designer String menu, select **concat**. The **concat** method box appears.
- D Right-click the **objMSWord_Application** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The **method selection** window appears.
- E Select **name()** from the method selection window. The **Name** method box appears in the Business Rules Designer canvas.
- F Map the **result (String)** output node of the **Name** method box to the **String** input node of the **concat** method box.
- G Map the **Application Name:** output node of the **String** method box to the **str (String)** input node of the **concat** method box.
- H Map the **result (String)** output node of the **concat** method box to **Text** under **FileClient_1** in the right pane of the Business Rules designer (see Figure 23).

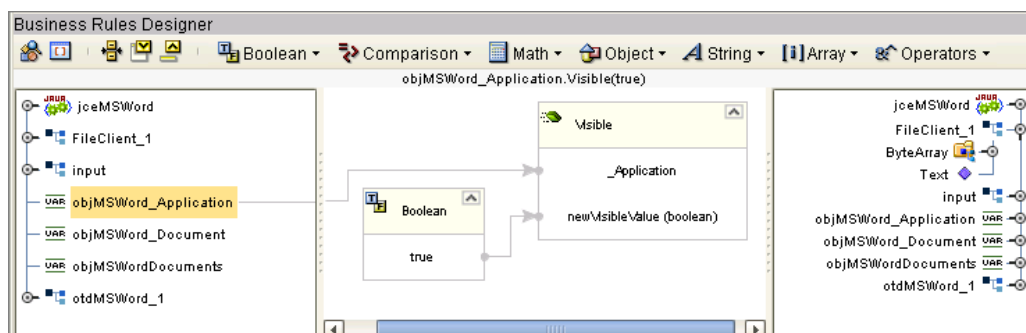
Figure 23 Collaboration Editor (Java) - Business Rules Designer



- 10 To create the **FileClient1_Write** rule, do the following:
 - A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click **FileClient1** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The **method selection** window appears.
 - C Select **write()** from the method selection window. The **write** method box appears in the Business Rules Designer canvas.
- 11 The **Copy "Application Value: ".concat(objMSWord_Application.Version) to FileClient_1.Text** rule, along with the **FileClient_1.write** rule, provides the application version (value) and writes it to the output file. To create this rule, do the following:

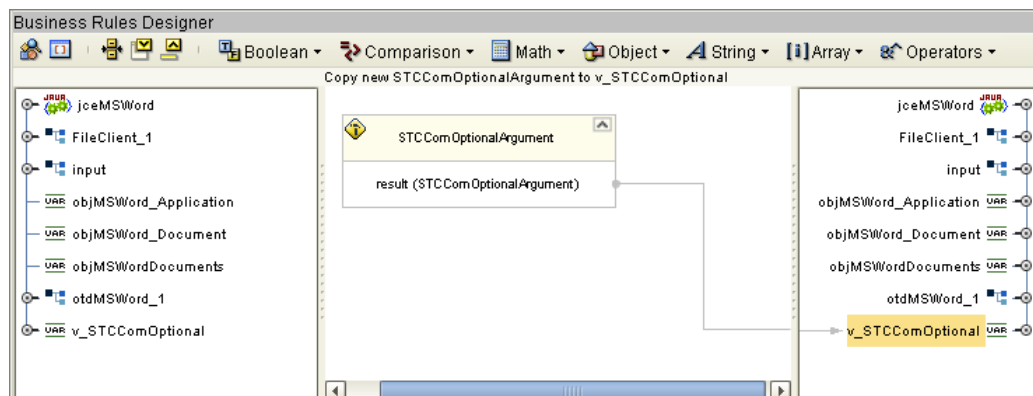
- A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B From the Business Rules Designer String menu, select **Literal String**. The **String** method box appears. Enter **Application Value:** as the value.
 - C From the Business Rules Designer String menu, select **concat**. The **concat** method box appears.
 - D Right-click the **objMSWord_Application** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.
 - E Select **version()** from the method selection window. The **Version** method box appears in the Business Rules Designer canvas.
 - F Map the **result (String)** output node of the **Version** method box to the **String** input node of the **concat** method box.
 - G Map the **Application Value:** output node of the **String** method box to the **str (String)** input node of the **concat** method box.
 - H Map the **result (String)** output node of the **concat** method box to **Text** under **FileClient_1** under the right pane of the Business Rules designer.
- 12 To create the next **FileClient_1.write** rule, follow the steps for **Step 10** on page 39.
- 13 The **objMSWord_Application.Visible(true)** rule allows the user to see the application running. To create this rule, do the following:
- A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click the **objMSWord_Application** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu. The method selection window appears.
 - C Select **visible(boolean newVisibleValue)** from the method selection window. The **Visible** method box appears.
 - D From the Business Rules Designer Boolean menu, select **True**. The **Boolean** method box appears.
 - E Map the **true** output node of the **Boolean** method box to the **newVisibleValue (boolean)** input node of the **Visible** method box (see Figure 24).

Figure 24 Collaboration Editor (Java) - Business Rules Designer



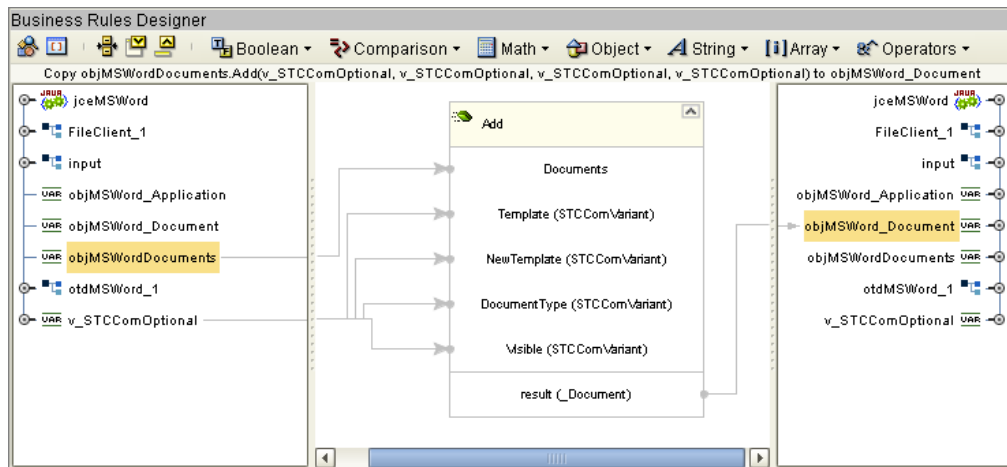
- 14 The **Copy objMSWord_Application.Documents to objMSWordDocuments** rule creates a new Word document. To create this rule, do the following:
 - A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click the **objMSWord_Application** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.
 - C Select **Documents()** from the method selection window. The **Documents** method box appears.
 - D Map the **result (Documents)** output node of the **Documents** method box to the **objMSWordDocuments** field in the right pane of the Business Rules designer.
- 15 The **Create uninitialized variable v_STCComOptional (of type STCComOptionalArgument)** variable creates an optional argument. To create this rule, do the following:
 - A From the Business Rules toolbar, click **local variable**. The **Create Variable** dialog box appears.
 - B Enter **v_STCComOptional** as the variable name.
 - C For Type, select **Class** and click the ellipsis (...) button. The **Find Class** dialog box appears.
 - D Select **STCComOptionalArgument** under **All Classes**, and click **OK**.
 - E Click **OK** to close the **Create Variable** dialog box and create your variable.
- 16 The **Copy new STCComOptionalArgument to v_STCComOptional** rule copies the optional argument to the methods. To create this rule, do the following:
 - A From the Business Rules Designer toolbar, click the **Class Browser** icon. The **Class Browser** dialog box appears.
 - B Select **STCComOptionalArgument** under **All Classes** field and click **OK**. The **STCComOptionalArgument** method box appears.
 - C Map the **result (STCComOptionalArgument)** output node of the **Call Constructor** method box to the **v_STCComOptional** field in the right pane of the Business Rules designer (see Figure 25).

Figure 25 Collaboration Editor (Java) - Business Rules Designer



- 17 The **Copy objMSWordDocuments.Add(v_STCComOptional, v_STCComOptional, v_STCComOptional, v_STCComOptional) to objMSWord_Document** rule creates the new Word document. To create this rule, do the following:
- A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click the **objMSWordDocuments** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.
 - C Select **Add(com.stc.connector.comadapter.STCComVariant Template...)** from the method selection window. The **Add** method box appears.
 - D Map the **v_STCComOptional** field in the left pane of the Business Rules Designer, to each of the following input nodes of the **Add** method box:
 - ♦ Template (STCComVariant)
 - ♦ New Template (STCComVariant)
 - ♦ Document Type (STCComVariant)
 - ♦ Visible (STCComVariant)
 - E Map the **result (_Document)** output node of the **Add** method box to the **objMSWord_Document** field in the right pane of the Business Rules designer (see Figure 26).

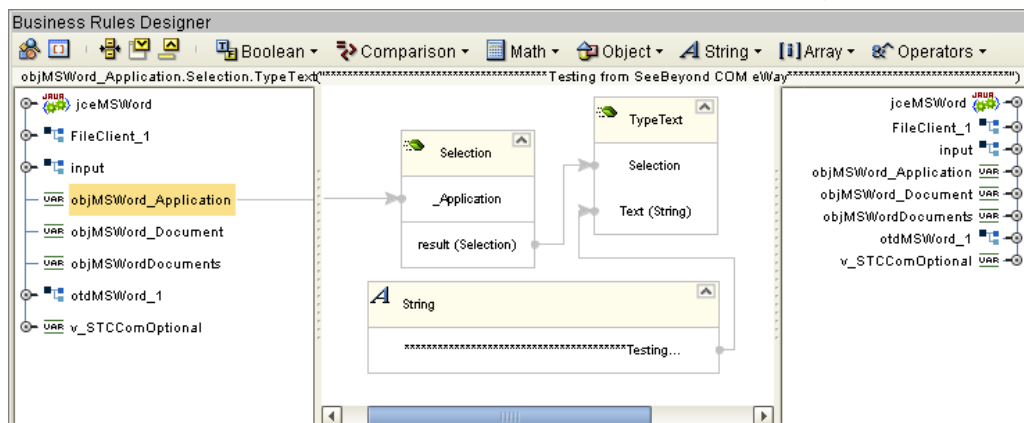
Figure 26 Collaboration Editor (Java) - Business Rules Designer



- 18 The **objMSWord_Application.Selection.TypeText("*****
*****\nTesting from SeeBeyond COM eWay\n*****
*****")** rule adds this text string to the document. To create this rule, do the following:
- A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click the **objMSWord_Application** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

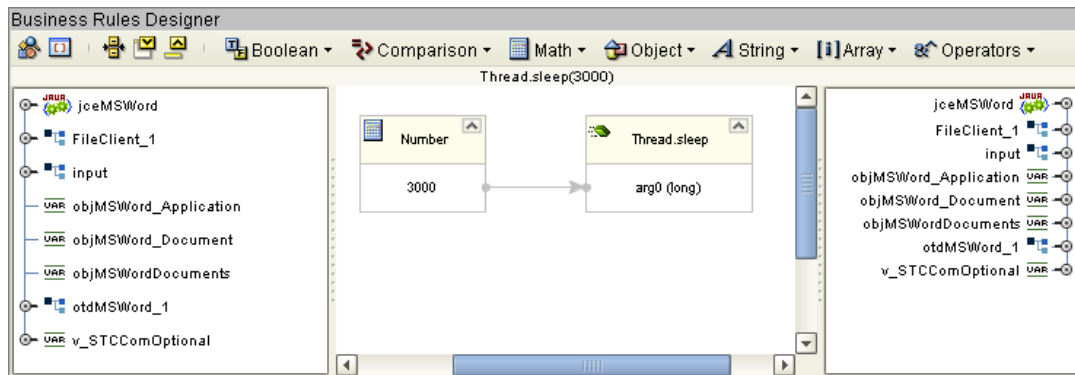
- C Select **Selection()** from the method selection window. The **Selection** method box appears.
- D Right-click the **result (Selection)** output node of the **Selection** method box, and click **Select method to call** from the shortcut menu.
- E Select **TypeText()** from the method selection window. The **TypeText** method box appears.
- F From the Business Rules Designer String menu, select **Literal String**. The **String** method box appears. Enter `*****\n Testing from SeeBeyond COM eWay\n*****` as the value.
- G Map the output node of the **String** method box to the **Text (String)** input node of the **TypeText** method box (see Figure 27).

Figure 27 Collaboration Editor (Java) - Business Rules Designer



- 19 The **lang.Thread.sleep(3000)** rule adds a three second delay so that the user can watch the operation as the document is created. To create this rule, do the following:
 - A From the Business Rules Designer Math menu, select **Literal Number**. The **Number** method box appears. Enter **3000** as the value.
 - B From the Business Rules Designer toolbar, click the **Class Browser** icon. The **Class Browser** dialog box appears.
 - C From the **Class Browser** dialog box, select **Thread** in the **All Classes** field, select **sleep (long arg0)** in the **Thread** field, and click **OK**. The **Thread.sleep** method box appears.
 - D Map the **3000** output node of the **Literal** method box to the **Arg0 (long)** input node of the **Thread.sleep** method box (see [Figure 28 on page 44](#)).

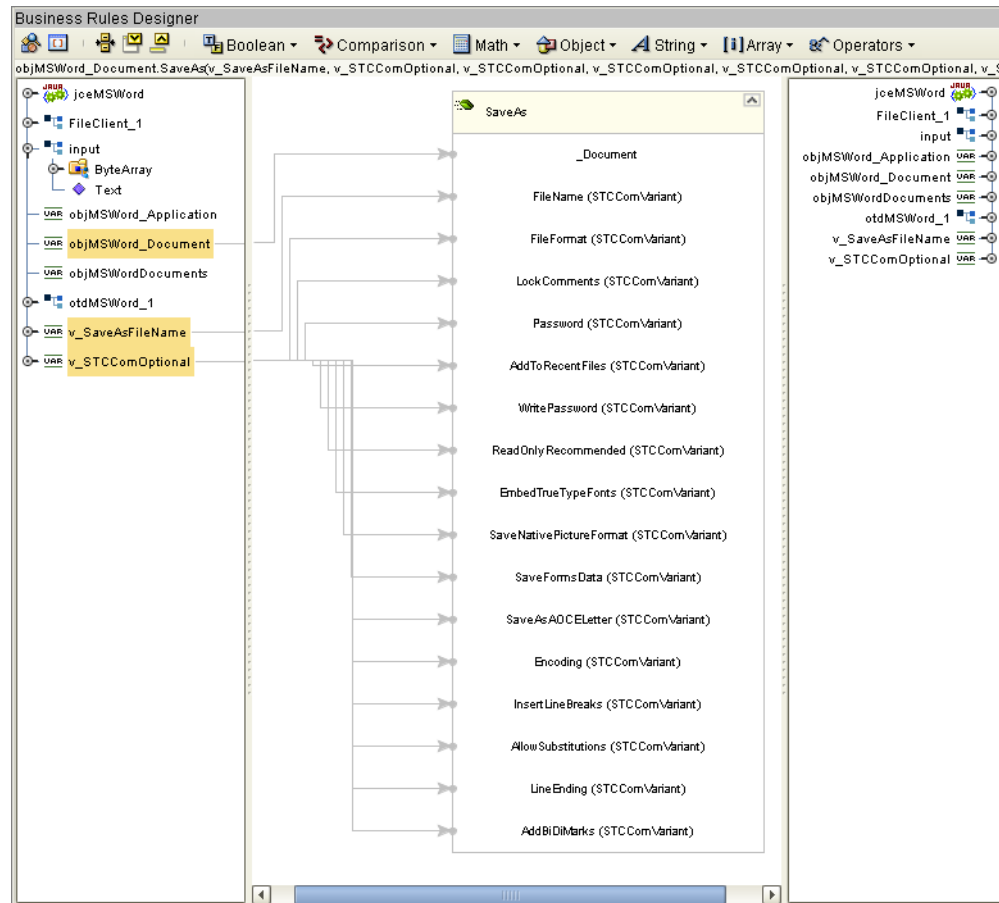
Figure 28 Collaboration Editor (Java) - Business Rules Designer



- 20 The **Create uninitialized variable v_SaveAsFileName** (of type **STCComVariant**) variable creates a variable for the file name and location that uses the name/ location string from the input file. To create this rule, do the following:
 - A From the Business Rules toolbar, click **Local Variable**. The **Create Variable** dialog box appears.
 - B Enter **v_SaveAsFileName** as the variable name.
 - C For Type, select **Class** and click the ellipsis (...) button. The **Class Browser** dialog box appears.
 - D Select **STCComVariant** under **All Classes**, and click **OK**.
 - E Click **OK** to close the **Create Variable** dialog box and create the variable.
- 21 The **Copy new STCComVariant(input.Text) to v_SaveAsFileName** rule copies the file name and location from the input file as the name and save location of the new document. To create this rule, do the following:
 - A From the Business Rules Designer toolbar, click the **Class Browser** icon. The **Class Browser** dialog box appears. Select **STCComVariant** under the **All Classes** and select **STCComVariant (String s)** under **STCComVariant**. Click **OK**. The **STCComVariant** method box appears in the Business Rules Designer canvas.
 - B Map **Text** under **input** in the left pane of the Business Rules Designer, to the **s (String)** input node of the **STCComVariant** method box.
 - C Map the **result (STCComVariant)** output node of the **STCComVariant** method box to the **v_SaveAsFileName** field in the right pane of the Business Rules Designer.
- 22 The **objMSWord_Document.SaveAs(v_SaveAsFileName, v_STCComOptional, v_STCComOptional, ...)** rule provides 15 options for associated arguments to save the new document. **v_SaveAsFileName**, created in step 21, is used for the **FileName** argument. To create this rule, do the following:
 - A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click the **objMSWord_Document** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

- C Select **SaveAs(com.stc.connector.comadapter.comruntime.STCComVariant...)** from the method selection window. The **SaveAs** method box appears.
- D Map the **v_SaveAsFileName** field in the left pane of the Business Rules Designer, to the **FileName (STCComVariant)** input node of the **SaveAs** method box.
- E Map the **v_STCComOptional** field in the left pane of the Business Rules Designer, to each of the following input nodes of the **SaveAs** method box as displayed in **Figure 29 on page 46**.
 - ♦ FileFormat (STCComVariant)
 - ♦ LockComments (STCComVariant)
 - ♦ Password (STCComVariant)
 - ♦ AddToRecentFiles (STCComVariant)
 - ♦ WritePassword (STCComVariant)
 - ♦ ReadOnlyRecommended (STCComVariant)
 - ♦ EmbedTrueTypeFonts (STCComVariant)
 - ♦ SaveNativePictureFormat (STCComVariant)
 - ♦ SaveFormsData (STCComVariant)
 - ♦ SaveAsOAOCELetter (STCComVariant)
 - ♦ Encoding (STCComVariant)
 - ♦ InsertLineBreaks (STCComVariant)
 - ♦ AllowSubstitutions (STCComVariant)
 - ♦ LineEnding (STCComVariant)
 - ♦ AddBiDiMarks (STCComVariant)

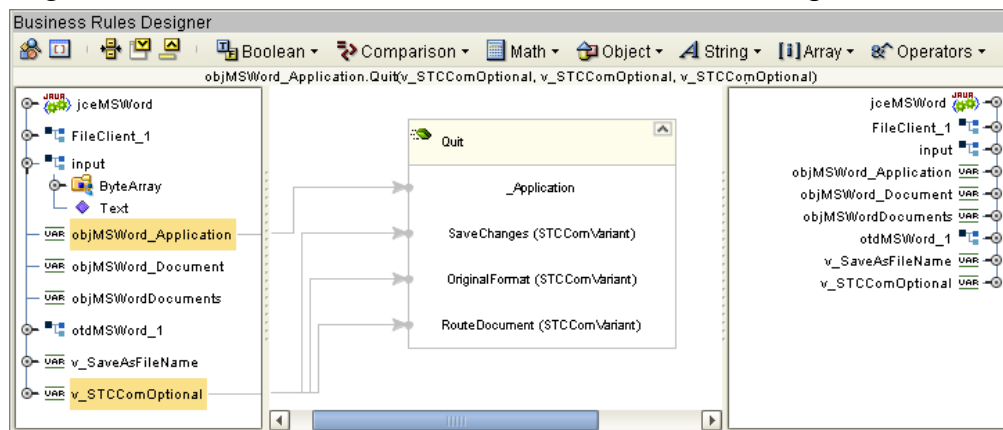
Figure 29 Collaboration Editor (Java) - Business Rules Designer



- 23 The `lang.Thread.sleep(1000)` provides a one second delay to allow the user to observe the operation. To create this rule, do the following:
 - A From the Business Rules Designer Math menu, select **Literal Number**. The **Number** method box appears. Enter **1000** as the value.
 - B From the Business Rules Designer toolbar, click the **Class Browser** icon. The **Class Browser** dialog box appears.
 - C From the **Class Browser** dialog box, select **Thread** in the **All Classes** field, select **sleep (long arg0)** in the **Thread** field, and click **OK**. The **Thread.sleep** method box appears.
 - D Map the **1000** output node of the **Literal** method box to the **Arg0 (long)** input node of the **Thread.sleep** method box
- 24 The `objMSWord_Application.Quit(v_STCComOptional, v_STCComOptional, v_STCComOptional)` rule ends the operation and closes the document. To create this rule, do the following:
 - A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.
 - B Right-click the `objMSWord_Application` field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

- C Select **Quit(com.stc.connector.comadapter.comruntime.STCComVariant SaveChanges, com.stc.connector.comadapter.comruntime.STCComVariant RouteDocument)** from the method selection window. The **Quit** method box appears.
- D Map the **v_STCComOptional** field in the left pane of the Business Rules Designer, to each of the following input nodes of the **Quit** method box as displayed in Figure 30).
 - ♦ SaveChanges (STCComVariant)
 - ♦ OriginalFormat (STCComVariant)
 - ♦ RouteDocument (STCComVariant)

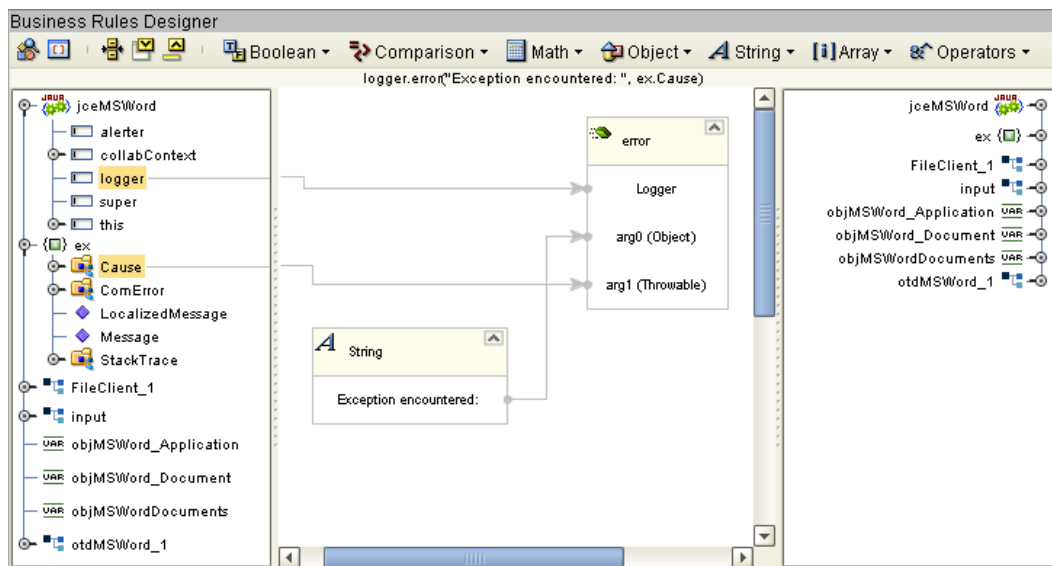
Figure 30 Collaboration Editor (Java) - Business Rules Designer



- 25 The **STCComException catch block** provides error handling for the Collaboration. To create this rule, do the following:
 - A From the Business Rules pane, collapse all of the rules under the **Try** on the Business Rules tree. To do this, right-click **Try** on the Business Rules tree, and select **Collapse All Rules** from the shortcut menu.
 - B From the Business Rules tree, right-click **Try** and select **catch** from the shortcut menu. The **Create Exception Variable** dialog box appears.
 - C From the **Create Exception Variable** dialog box, enter **STCComException** as the **Name**.
 - D Select **Class** as the type and click the ellipsis (...) button. The **Class Browser** dialog box appears. Select **STCComException** under **All Classes** and click **OK**.
 - E Click **OK** to close the **Create Exception Variable** dialog box.
- 26 The **logger.error("Exception encountered: ", ex.Cause)** rule under **catch**, catches an error if one is thrown in the **Try** block, and writes the error to the log file. To create this rule, do the following:
 - A From the Business Rules tree, expand **catch** and select **rules** under the **catch**.
 - B Click **rule** on the Business Rules toolbar to add a rule to the Business Rules tree.

- C Right-click the **logger** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.
- D Select **error(object arg0, Throwable arg1)** from the method selection window. The **error** method box appears.
- E From the Business Rules Designer String menu, select **Literal String**. The **String** method box appears. Enter **Exception encountered:** as the value.
- F Map the **Exception encountered:** output node of the String method box, to the **arg0 (Object)** input node of the **error** method box.
- G Map **Cause**, under the **ex** field in the left pane of the Business Rules Designer, to the **arg1 (Throwable)** input node of the **error** method box (see Figure 31).

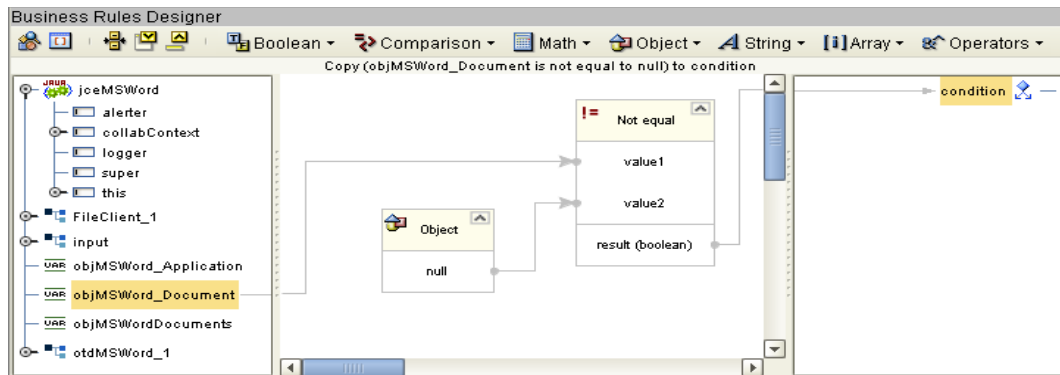
Figure 31 Collaboration Editor (Java) - Business Rules Designer



- 27 The **If objMSWord_Document is not equal to null** rule, along with the **then** rule, **objMSWord_Document.Release** (under **finally**), releases the local interface handle and removes the **objMSWord_Document** COM object from the server's task manager list. To create this rule, do the following:
 - A Select the **finally** node on the Business Rules tree.
 - B Click the **if** icon on the Business Rules toolbar to add a new **if** statement to the Business Rules tree.
 - C From the Business Rules Designer Comparison menu, select **Settings**. The **Method Palette** dialog box appears. Click the Comparison tab, and select the **!= not equal tool**. Click Close. The new tool is added to the Comparison menu.
 - D From the Business Rules Designer Comparison menu, select **!= not equal**. The **!= not equal** method box is added to the canvas.
 - E From the Business Rules Designer Object menu, select **Null**. The **Object** dialog box appears.

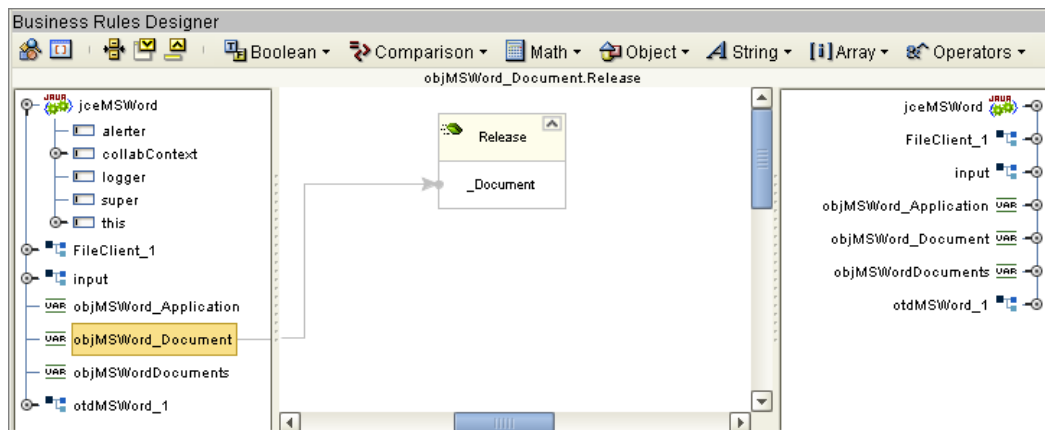
- F Map **objMSWord_Document** in the left pane of the Business Rules Designer to the **value1** input node of the != not equal method box.
- G Map the **null** output node of the Object method box to the **value2** input node of the != not equal method box.
- H Map the **result (boolean)** output node of the != not equal method box to the **condition** node in the right pane of the Business Rules Designer (see Figure 32).

Figure 32 Collaboration Editor (Java) - Business Rules Designer



- 28 To create the **then** statement rule, **objMSWord_Document.Release**, under the **If objMSWord_Document is not equal to null** rule, do the following:
 - A Select the **then** under the **If objMSWord_Document is not equal to null** rule on the Business Rules tree.
 - B Click **rule** on the Business Rules toolbar to add a rule under the **then** statement.
 - C Right-click **objMSWord_Document** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.
 - D Select **Release()** from the method selection window. The **Release** method box appears (see Figure 33).

Figure 33 Collaboration Editor (Java) - Business Rules Designer



- 29 The **If objMSWordDocuments is not equal to null** rule, along with the **then** rule, **objMSWordDocuments.Release** (under **finally**), releases the local interface handle

and removes the **objMSWordDocuments** COM object from the server's task manager list. To create this rule, do the following:

- A Click the **if** icon on the Business Rules toolbar to add a new **if** statement to the Business Rules tree.
 - B From the Business Rules Designer Comparison menu, select **! = not equal**. The **! = not equal** method box is added to the canvas.
 - C From the Business Rules Designer Object menu, select **Null**. The **Object** dialog box appears.
 - D Map **objMSWordDocuments** in the left pane of the Business Rules Designer to the **value1** input node of the **! = not equal** method box.
 - E Map the **null** output node of the Object method box to the **value2** input node of the **! = not equal** method box.
 - F Map the **result (boolean)** output node of the **! = not equal** method box to the **condition** node in the right pane of the Business Rules Designer
- 30 To create the **then** statement rule, **objMSWordDocuments.Release**, under the **If objMSWordDocuments is not equal to null** rule, do the following:
- A Select the **then** under the **If objMSWordDocuments is not equal to null** rule on the Business Rules tree.
 - B Click **rule** on the Business Rules toolbar to add a rule under the **then** statement.
 - C Right-click **objMSWordDocuments** in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.
 - D Select **Release()** from the method selection window. The **Release** method box appears
- 31 The **If objMSWord_Application is not equal to null** rule, along with the **then** rule, **objMSWord_Application.Release** (under **finally**), releases the local interface handle and removes the **objMSWord_Application** COM object from the server's task manager list. To create this rule, do the following:
- A Click the **if** icon on the Business Rules toolbar to add a new **if** statement to the Business Rules tree.
 - B From the Business Rules Designer Comparison menu, select **! = not equal**. The **! = not equal** method box is added to the canvas.
 - C From the Business Rules Designer Object menu, select **Null**. The **Object** dialog box appears.
 - D Map **objMSWord_Application** in the left pane of the Business Rules Designer to the **value1** input node of the **! = not equal** method box.
 - E Map the **null** output node of the Object method box to the **value2** input node of the **! = not equal** method box.
 - F Map the **result (boolean)** output node of the **! = not equal** method box to the **condition** node in the right pane of the Business Rules Designer
- 32 To create the **then** rule **objMSWord_Application.Release** under the **If objMSWord_Application is not equal to null** rule, do the following:

- A Select the **then** under the **If objMSWord_Application is not equal to null** rule on the Business Rules tree.
 - B Click **rule** on the Business Rules toolbar to add a rule under the **then** statement.
 - C Right-click **objMSWord_Application** in the left pane of the Business Rules Designer, and click **Select a method to call** from the shortcut menu.
 - D Select **Release()** from the method selection window. The **Release** method box appears
- 33 The **Copy "***** Microsoft Word Test Completed *****"** to **FileClient_1.Text** rule, along with the **FileClient_1.write** rule, adds the "completed" text string to the output file. To create this rule, do the following:
- A Select the **receive** method at the top of the Business Rules tree.
 - B From the Business Rules toolbar, click **rule**. An empty rule is added at the bottom of the Business Rules tree.
 - C From the Business Rules Designer String menu, select **Literal String**. The **String** method box appears. Enter ******* Microsoft Word Test Completed ******* as the value.
 - D Map the ******* Microsoft Word Test Completed ******* output node of the **Literal** method box to **Text** under **FileClient_1** in the right pane of the Business Rules Designer.
- 34 To create the next **FileClient_1.write** rule, follow the steps for **Step 10** on page 39.
- 35 Save your current changes to the Repository.

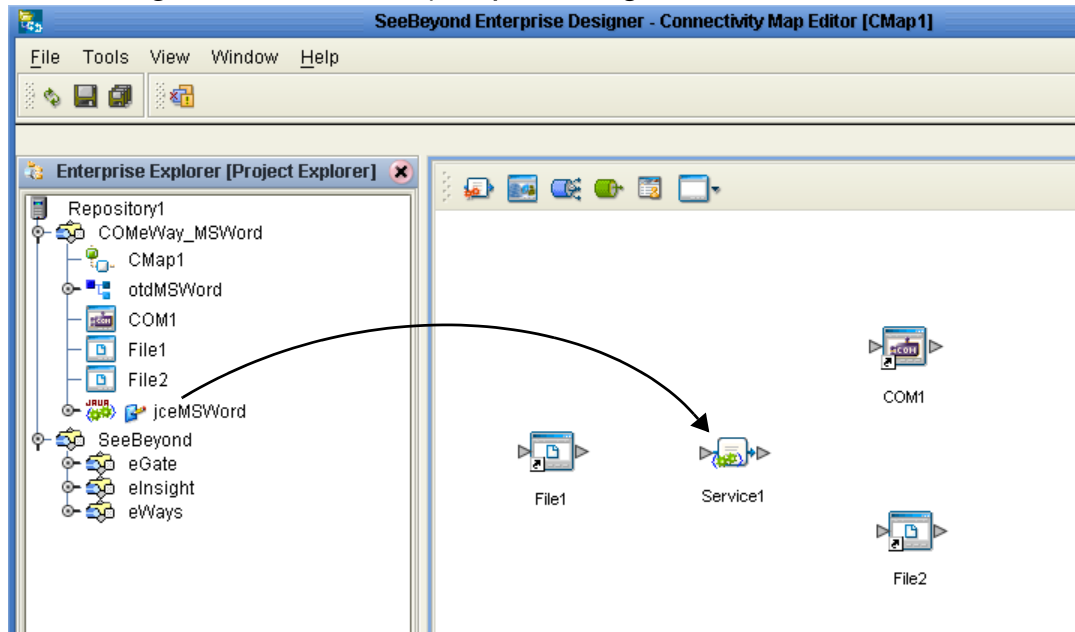
Note: For more information on creating Business Rules using the Collaboration Editor See the eGate Integrator User's Guide.

5.6.8. Creating Collaboration Bindings

After the Collaboration has been written, the components are associated and bindings are created in the Connectivity Map.

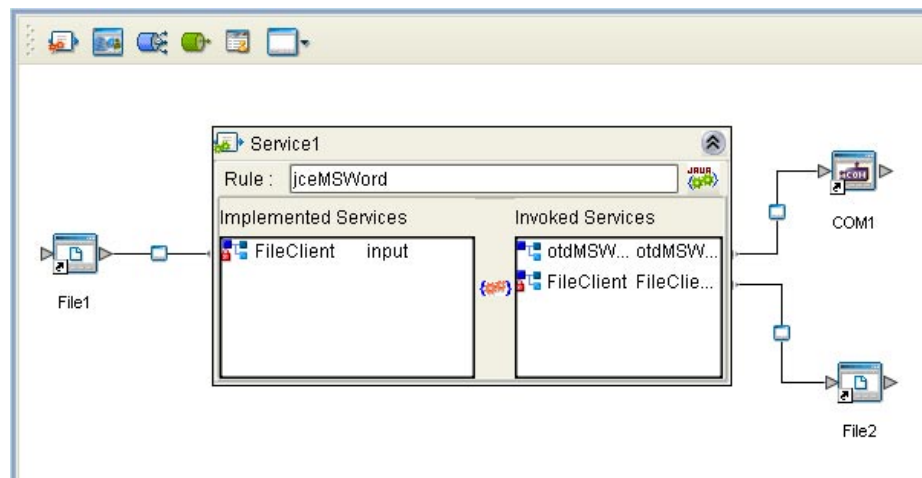
- 1 From the Project Explorer tree, double-click **CMap1** to display the Connectivity Map.
- 2 Drag and drop the **jceMSWord** Collaboration from the Project Explorer tree to **Service1** in the Connectivity Map. If the Collaboration is successfully associated, the Service icon's "gears" change from red to green (see Figure 34).

Figure 34 Connectivity Map - Binding the Collaborations



- 3 Double-click **Service1** in the Connectivity Map. The **Service1** dialog box appears.
- 4 From the **Service1** dialog box, map **FileClient input** (under Implemented Services) to the **File1** inbound External Application, by clicking on **FileClient input** in the Service1 dialog box and dragging the cursor to the **File1** application.
- 5 From the **Service1** dialog box, map **otdMSWord_1, otdMSWord** (under Invoked Services) to the **COM1** External Application.
- 6 From the **Service1** dialog box, map **FileClient_1 FileClient** (under Invoked Services) to the **File2** outbound External Application (see Figure 35).

Figure 35 Connectivity Map - Connecting the Project's Components



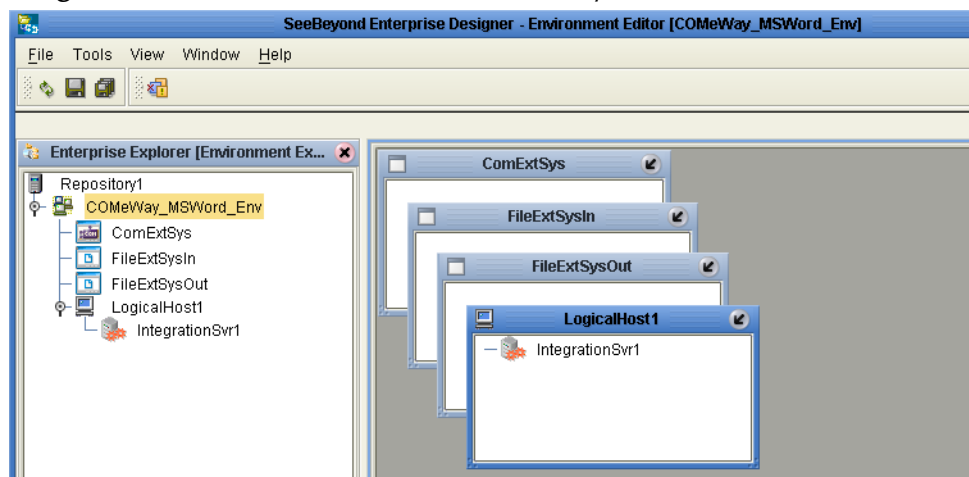
- 7 Minimize the **Service1** dialog box, and save your current changes to the Repository.

5.6.9. 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 **COMeWay_MSWord_Env**.
- 4 Right-click **COMeWay_MSWord_Env** and select **New COM/DCOM External System**. Name the External System **ComExtSys**. **ComExtSys** is added to the Environment Editor.
- 5 Right-click **COMeWay_MSWord_Env** and select **New File External System**. From the **Create an External System** dialog box, enter **FileExtSysIn** as the name and select **Inbound File eWay** as the type. Click **OK**. **FileExtSysIn** is added to the Environment Editor.
- 6 Right-click **COMeWay_MSWord_Env** and select **New File External System** again. Enter **FileExtSysOut** as the name and select **Outbound File eWay** as the type. **FileExtSysOut** is added to the Environment Editor.
- 7 Right-click **COMeWay_MSWord_Env** and select **New Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.
- 8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server** from the shortcut menu. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under **LogicalHost1** (see [Figure 36 on page 53](#)).

Figure 36 Environment Editor - COMeWay_MSWord_Env



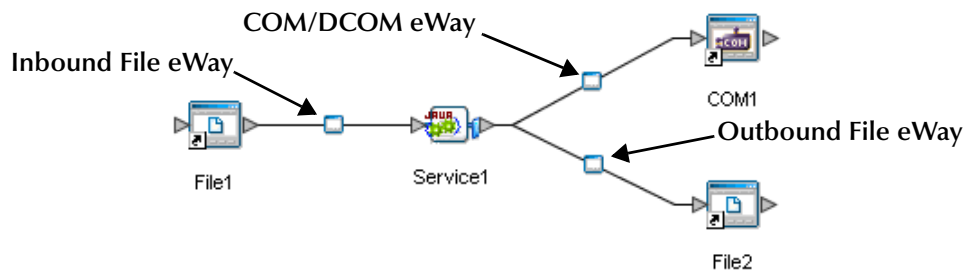
- 9 At this point, upload the JAR files (see [Adding JAR files to the Integration Server Classpath](#) on page 13)

- 10 Save your current changes to the Repository.

5.6.10. Configuring the eWays

The **COMeWay_MSWord** sample project uses three eWays, each represented in the Connectivity Map as a node between an External Application and the Collaboration (see Figure 37). The eWays facilitate communication and movement of data between the External Applications and the eGate system.

Figure 37 eWays



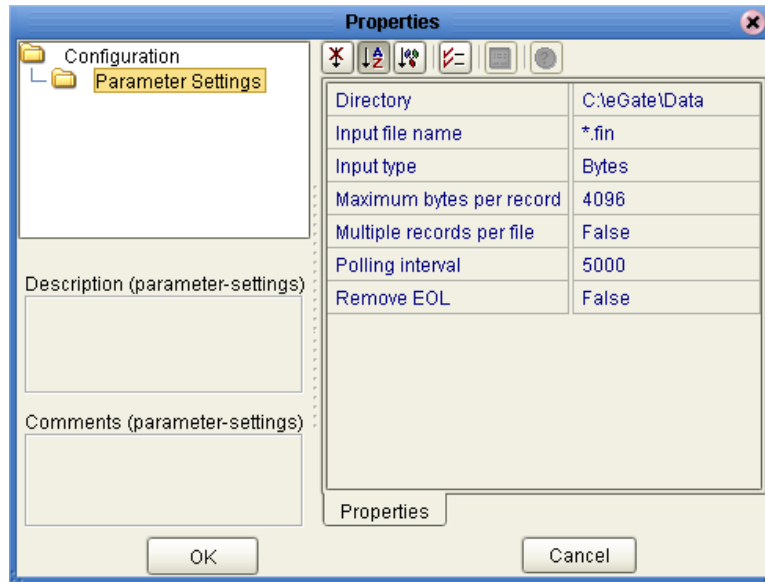
Configuring the File eWays

File eWays only possess Connectivity Map properties. These are accessed by double-clicking the eWay node in the Connectivity map.

Inbound File eWay (File1) Properties

- 1 Double-click the inbound File eWay (see Figure 37). The Templates dialog box appears. Select **Inbound File eWay** in the Templates dialog box and click **OK**.
- 2 The Properties Editor opens to the inbound File eWay properties. Modify the properties for your system as displayed in Figure 38.

Figure 38 Properties Editor - Inbound File eWay (File1) Properties



- 3 Click **OK**. The properties settings are saved for the eWay.

Outbound File eWay (File2) Properties

- 1 Double-click the outbound File eWay located between **Service1** and the outbound **File2** External Application. Select **Outbound File eWay** from the Templates dialog box. The Properties Editor appears.
- 2 Modify the outbound File eWay configuration for your system, including the settings in Table 3, and click **OK**.

Table 3 Outbound File eWay Settings

Outbound eWay Configuration Parameters	
Add EOL	True
Directory	C:\eGate\Data
Multiple records per file	True
Output file name	output%.dat

Configuring the COM/DCOM eWay

The COM/DCOM eWay properties are set from Environment Explorer only (the COM.DCOM eWay does not possess Connectivity Map Properties).

Modifying the COM/DCOM eWay Environment Explorer Properties

- 1 From the **Environment Explorer** tree, right-click the COM/DCOM External System (**COMExtSys** in this sample), and select **Properties**. The Properties Editor opens to the COM/DCOM eWay properties.

- 2 Modify the **Server** property. If this property is left blank, the value defaults to your logical host.
- 3 Click OK to save your settings and close the editor.
- 4 For more information on the COM/DCOM eWay configuration properties see [Configuring the COM/DCOM eWay](#) on page 15.

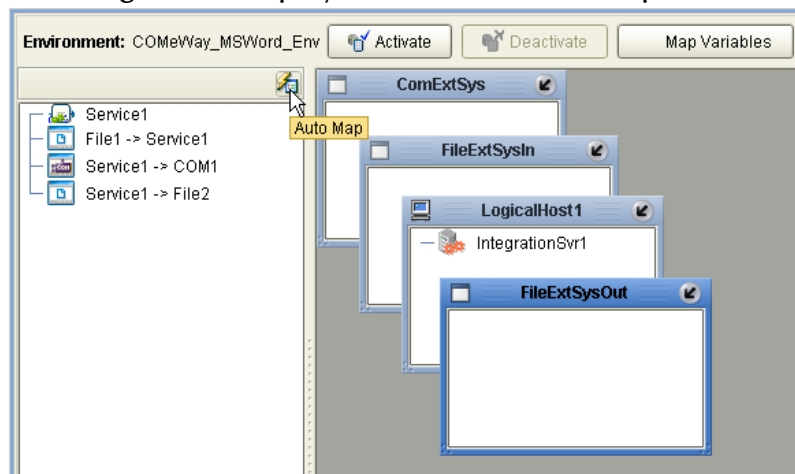
5.6.11. Creating and Activating the Deployment Profile

Deployment Profiles are specific instances of a project in a particular Environment. A Deployment Profile is created using the Enterprise Designer's Deployment Editor.

To create the **COMeWay_MSWord** sample Deployment Profile do the following:

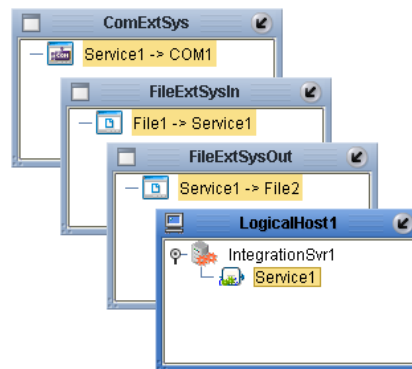
- 1 From the Enterprise Explorer's Project Explorer, right-click the project (**COMeWay_MSWord**) and select **New > Deployment Profile**.
- 2 From the **Create Deployment Profile for COMeWay_MSWord** dialog box, enter a name for the Deployment Profile (for this sample **COMeWay_MSWord_DP**), and select **COMeWay_MSWord_Env** as the Environment. Click **OK**.
- 3 Click the **Auto Map** icon as displayed in [Figure 39 on page 56](#). The projects components are automatically mapped to their system window as seen in [Figure 40 on page 57](#). If any of the project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8.

Figure 39 Deployment Profile - Auto Map



- 4 From the left pane of the Deployment Editor, drag **Service1 -> COM1** to the **ComExtSys** window.
- 5 From the left pane of the Deployment Editor, drag **File1 -> Service1** to the **FileExtSysIn** window.
- 6 From the left pane of the Deployment Editor, drag **Service1 -> File2** to the **FileExtSysOut** window.
- 7 Drag **Service1** to **IntegrationSvr1** in the **LogicalHost1** window (see Figure 40).

Figure 40 Deployment Profile



- 8 Click **Activate**, then save the changes to the Repository.

5.6.12. Running the Project

Designate the Input and Output Directories

Before starting the eWay, create the input and output directories. For this sample, input and output share the same directory: **C:\eGate\Data**. You can designate any directories as the input and output directories, but the File eWay properties must be configured to use those directories before running the eWay. For more information on configuring these properties see [Configuring the eWays](#) on page 54.

Starting the Project

The following directions assume that the Enterprise Designer was downloaded to C:\ican50. If this is not the case, replace that location in the following directions with the appropriate location.

- 1 From the Enterprise Manager Downloads tab, double-click **Logical Host - for win32**.
- 2 Extract the file to the **C:\ican50\LogicalHost1** directory. You must specify the **LogicalHost1** directory for it to be created.
- 3 Navigate to the **C:\ican50\LogicalHost1\bootstrap\config** directory and open the **logical-host.properties** file using Notepad™.
- 4 Enter and save the following information in the appropriate fields:
 - ♦ Logical Host root directory: **C:\ican50\LogicalHost1**
 - ♦ Repository URL: **http://localhost:port number/repository name**
 - ♦ Repository user name and password: **Your user name and password**
 - ♦ Logical Host Environment name **COMeWay_MSWord_Env**
 - ♦ Logical Host name: **Localhost**

Save your changes to **logical-host.properties** and close the file.

- 5 Run the **bootstrap.bat** file located in the **C:\ican50\LogicalHost1\bootstrap\bin** directory.
- 6 Copy the sample input data file to the input directory, with the correct extension.

5.7 Creating the COMeWay_MSEExcel Sample Project

The following sections in this chapter describe how the COMeWay_MSEExcel sample project's components are created.

5.7.1. Create a Project

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

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

5.7.2 Create a Connectivity Map

The Connectivity Map provides a canvas for assembling and configuring a project's components.

- 1 In Enterprise Explorer's Project Explorer, right-click the new project and select **New > 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**.

5.7.3. Select the External Applications

To add the External Applications used with the COMeWay_MSEExcel project, do the following:

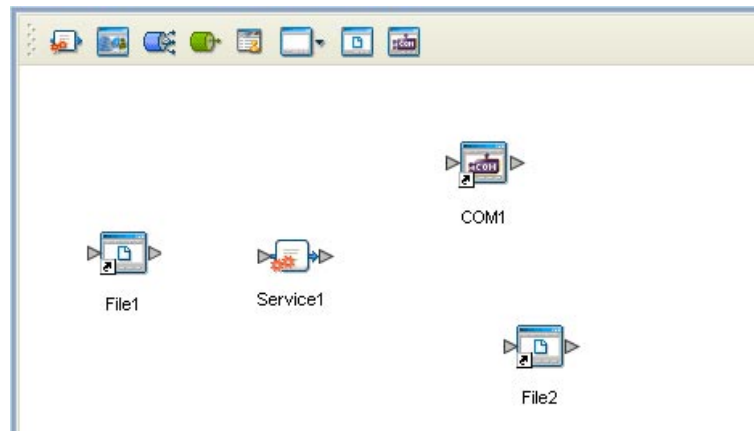
- 1 Click the **External Application** icon on the Connectivity Map toolbar.
- 2 Select the external systems that are necessary for your project (for this sample, **COM/DCOM** and **File**). Icons representing the selected external systems are added to the Connectivity Map toolbar.

5.7.4. Populate the Connectivity Map

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

- 1 For this sample, drag the following components onto the Connectivity Map canvas as displayed in [Figure 41 on page 59](#):
 - ♦ File External System (2)
 - ♦ Service (A Service is a container for Java Collaborations, Business Processes, eTL processes, and so forth.)
 - ♦ COM/DCOM External System

Figure 41 Connectivity Map with Components



- 2 Save your current changes.

5.7.5. Creating the otdMSEExcel OTD

The COM OTD Wizard generates an OTD from a COM automation-compatible component's **type library**. The COM Type Library file used for this sample is the **EXCEL.EXE** file from Microsoft Excel 2002.

To create an OTD using the COM OTD Wizard do the following:

- 1 From the Project Explorer tree, right click **COMeWay_MSEExcel** and select **New > Object Type Definition** from the shortcut menu.
- 2 From the **Select Wizard Type** window of the **New Object Type Definition Wizard**, select the **COM Wizard** and click **Next**.
- 3 Browse to the directory that contains the type library file, **EXCEL.EXE**. For example:

```
C:\Program Files\Microsoft Office\Office10
```

Select the **EXCEL.EXE** file and click the **Select** button, and click **Next**.
- 4 For **Step 3** of the wizard, select the classes you want from the type library (for this sample select all of the classes) and click **Next**.
- 5 For Step 4 of the wizard, enter **otdMSEExcel** as the name of the OTD, and click **Finish**.
- 6 An **Information** box is displayed indicating that some methods are not supported by the eWay. These methods will be skipped in the created OTD. The **Information** box also gives you the location of a generated "Skipped Methods" log that lists the methods skipped, as well as the issues that caused the method to be excluded.
- 7 The OTD Editor appears displaying the new OTD. The new **otdMSEExcel** OTD is added to the Project Explorer tree. The OTD is now available to use in a Collaboration.

For more information on using the COM OTD Wizard, see [Using the COM OTD Wizard](#) on page 22.

5.7.6. Creating the Collaboration Definitions

The next step in the sample is to create a Collaboration using the Collaboration Definition Wizard (Java). Once a Collaboration Definition has been created, the Business Rules of the Collaboration can be written using the Collaboration Editor (Java).

The jceMSEExcel Collaboration (Java)

The **jceMSEExcel** Collaboration defines transactions from the inbound File eWay to the COM/DCOM eWay and from the COM/DCOM application to the outbound File eWay.

- 1 From the Project Explorer, right-click the **COMeWay_MSEExcel** Sample project and select **New > Collaboration Definition (Java)** from the shortcut menu. The **Collaboration Definition Wizard (Java)** appears.
- 2 Enter a Collaboration Definition name (for this sample, **jceMSEExcel**) and click **Next**.
- 3 For Step 2 of the wizard, **Select a Web Services Operation**, double-click **SeeBeyond > eWays > File > FileClient > receive** to select the File eWay receive Web service. Click **Next**.
- 4 For Step 3 of the wizard, from the Select OTDs selection window, double-click **SeeBeyond > eWays > File > FileClient**. The **FileClient** OTD is added to the Selected OTDs field.
- 5 Click the **Up One Level** button to return to the Repository. Double-click **COMeWay_MSEExcel > otdMSEExcel**. The **otdMSEExcel** OTD is added to the Selected OTDs field.
- 6 Click **Finish**. The Collaboration Editor (Java) appears in the left pane of the Enterprise Designer and the **jceMSEExcel** Collaboration is added to the Project Explorer tree.

5.7.7. Using the Collaboration Editor (Java)

The next step in the sample is to create the Business Rules of the Collaboration using the Collaboration Editor. The the COMeWay_MSEExcel sample project uses one Collaboration created in the previous section, `jceMSEExcel`.

The `jceMSEExcel` Collaboration contains the Business Rules displayed in Figure 42.

Figure 42 `jceMSEExcel` Collaboration Business Rules



Note: Business Rules are wrapped in Figure 42 for display purposes only.

The Collaboration Editor (Java) allows you to create Business Rules using the **Business Rules Designer**, a graphical, drag-and-drop, interface. In addition, if a user is familiar with Java code, the editor also provides a **Java Source Editor**, which allows you to enter Java code directly, or display the code created when using the **Business Rules Designer** to create Business Rules. The completed Business Rules are displayed in the **Business Rules** pane of the editor (as displayed in [Figure 42 on page 61](#)).

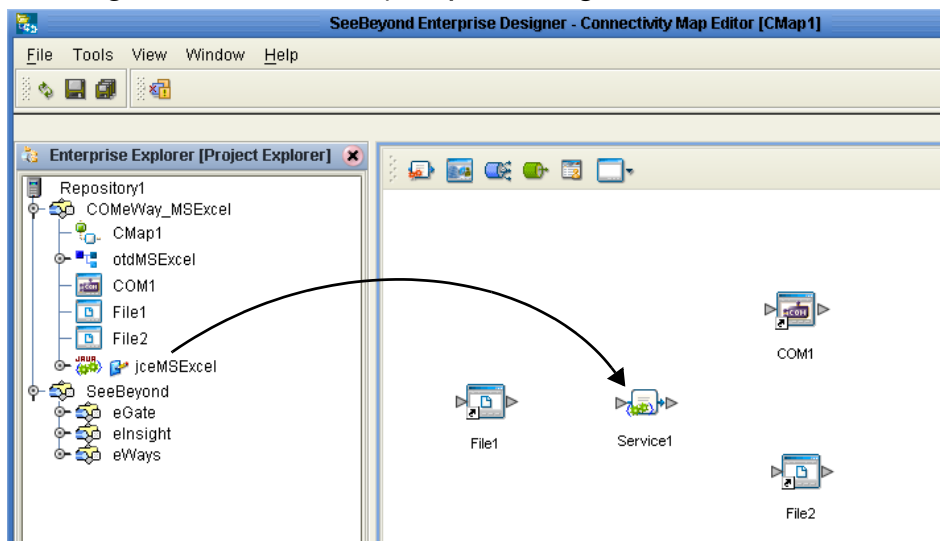
For more information on Creating Collaboration Business Rules see the *eGate Integrator User's Guide*.

5.7.8. Creating Collaboration Bindings

After the Collaboration has been written, the components are associated and bindings are created in the Connectivity Map.

- 1 From the Project Explorer tree, double-click **CMap1** to display the Connectivity Map.
- 2 Drag and drop the **jceMSEExcel** Collaboration from the Project Explorer tree to **Service1** in the Connectivity Map. If the Collaboration is successfully associated, the Service icon's "gears" change from red to green (see Figure 43).

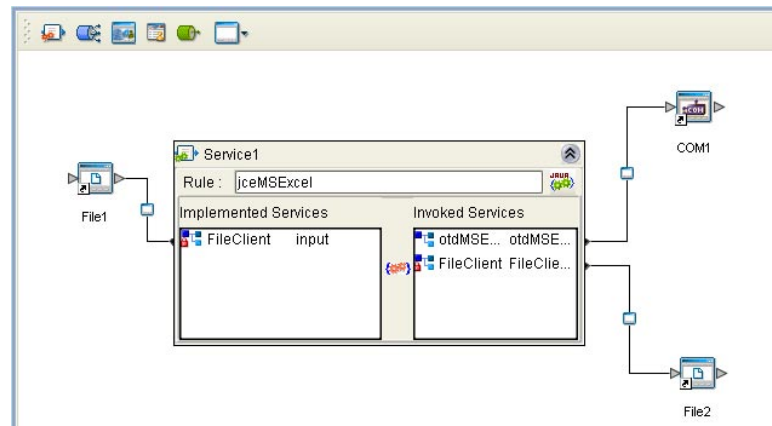
Figure 43 Connectivity Map - Binding the Collaborations



- 3 Double-click **Service1** in the Connectivity Map. The **Service1** dialog box appears.
- 4 From the **Service1** dialog box, map **FileClient input** (under Implemented Services) to the **File1** inbound External Application, by clicking on **FileClient input** in the Service1 dialog box and dragging the cursor to the **File1** application.
- 5 From the **Service1** dialog box, map **otdMSEExcel_1, otdMSEExcel** (under Invoked Services) to the **COM1** External Application.

- 6 From the **Service1** dialog box, map **FileClient_1 FileClient** (under Invoked Services) to the **File2** outbound External Application (see Figure 44).

Figure 44 Connectivity Map - Connecting the Project's Components



- 7 Minimize the **Service1** dialog box, and save your current changes to the Repository.

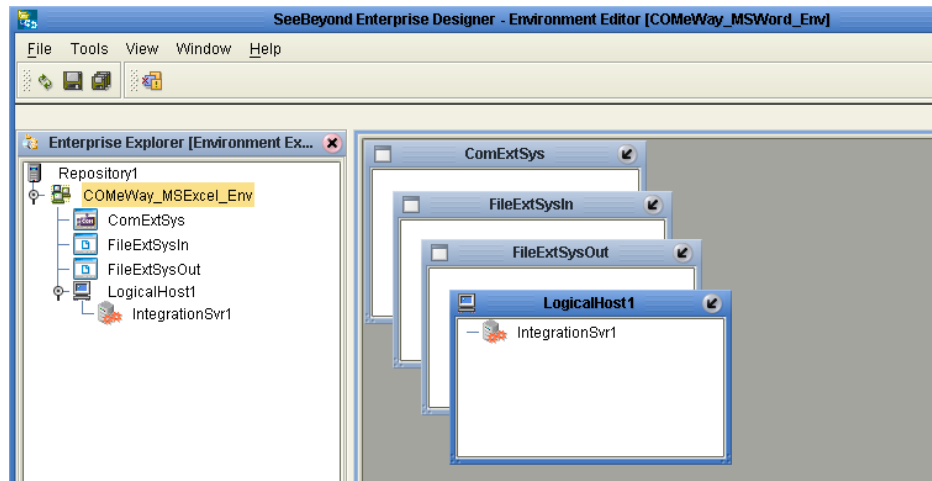
5.7.9. Creating an Environment

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 **COMeWay_MSEExcel_Env**.
- 4 Right-click **COMeWay_MSEExcel_Env** and select **New COM/DCOM External System**. Name the External System **ComExtSys**. **ComExtSys** is added to the Environment Editor.
- 5 Right-click **COMeWay_MSEExcel_Env** and select **New File External System**. From the **Create an External System** dialog box, enter **FileExtSysIn** as the name and select **Inbound File eWay** as the type. Click **OK**. **FileExtSysIn** is added to the Environment Editor.
- 6 Right-click **COMeWay_MSEExcel_Env** and select **New File External System** again. Enter **FileExtSysOut** as the name and select **Outbound File eWay** as the type. **FileExtSysOut** is added to the Environment Editor.
- 7 Right-click **COMeWay_MSEExcel_Env** and select **New Logical Host**. The **LogicalHost1** box is added to the Environment and **LogicalHost1** is added to the Environment Editor tree.
- 8 From the Environment Explorer tree, right-click **LogicalHost1** and select **New SeeBeyond Integration Server** from the shortcut menu. A new Integration Server

(**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1 (see Figure 45).

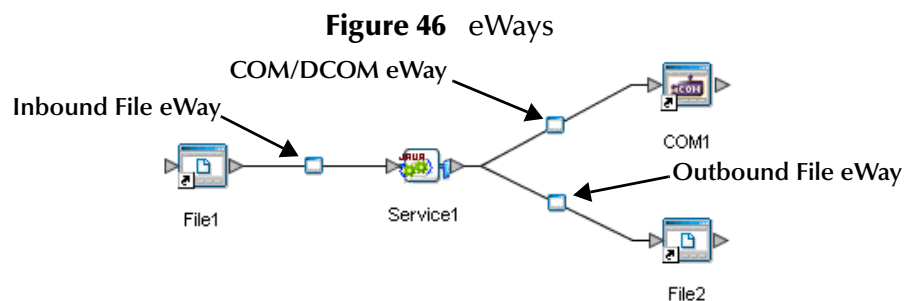
Figure 45 Environment Editor - COMeWay_MSExcel_Env



- 9 At this point, upload the JAR files (see [Adding JAR files to the Integration Server Classpath](#) on page 13)
- 10 Save your current changes to the Repository.

5.7.10. Configuring the eWays

The **COMeWay_MSExcel** sample project uses three eWays, each represented in the Connectivity Map as a node between an External Application and the Collaboration (see Figure 46). The eWays facilitate communication and movement of data between the External Applications and the eGate system.



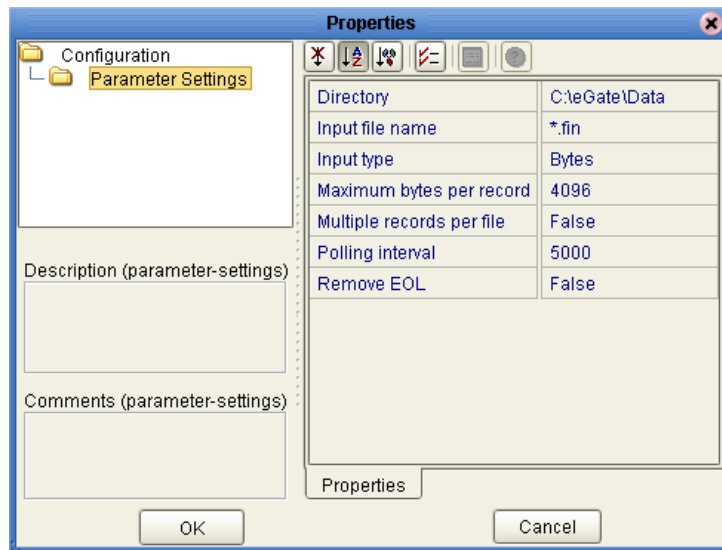
Configuring the File eWays

File eWays only possess Connectivity Map properties. These are accessed by double-clicking the eWay node in the Connectivity map.

Inbound File eWay (File1) Properties

- 1 Double-click the inbound File eWay (see Figure 46). The Templates dialog box appears. Select **Inbound File eWay** in the Templates dialog box and click **OK**.
- 2 The Properties Editor opens to the inbound File eWay properties. Modify the properties for your system as displayed in Figure 38.

Figure 47 Properties Editor - Inbound File eWay (File1) Properties



- 3 Click **OK**. The properties settings are saved for the eWay.

Outbound File eWay (File2) Properties

- 1 Double-click the outbound File eWay located between **Service1** and the outbound **File2** External Application. Select **Outbound File eWay** from the Templates dialog box. The Properties Editor appears.
- 2 Modify the outbound File eWay configuration for your system, including the settings in Table 3, and click **OK**.

Table 4 Outbound File eWay Settings

Outbound eWay Configuration Parameters	
Add EOL	True
Directory	C:\eGate\Data
Multiple records per file	True
Output file name	MSEExcel_output%d.dat

Configuring the COM/DCOM eWay

The COM/DCOM eWay properties are set from Environment Explorer only (the COM.DCOM eWay does not possess Connectivity Map Properties).

Modifying the COM/DCOM eWay Environment Explorer Properties

- 1 From the **Environment Explorer** tree, right-click the COM/DCOM External System (**COMExtSys** in this sample), and select **Properties**. The Properties Editor opens to the COM/DCOM eWay properties.
- 2 Modify the **Server** property. If this property is left blank, the value defaults to your logical host. Click **OK** to save your settings and close the editor.
- 3 For more information on the COM/DCOM eWay configuration properties see [Configuring the COM/DCOM eWay](#) on page 15.

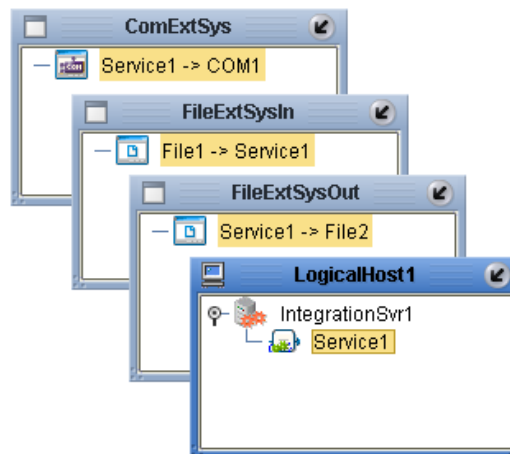
5.7.11. Creating and Activating the Deployment Profile

Deployment Profiles are specific instances of a project in a particular Environment. A Deployment Profile is created using the Enterprise Designer's Deployment Editor.

To create the **COMeWay_MSEExcel** sample Deployment Profile do the following:

- 1 From the Enterprise Explorer's Project Explorer, right-click the project (**COMeWay_MSEExcel**) and select **New > Deployment Profile**.
- 2 From the **Create Deployment Profile for COMeWay_MSEExcel** dialog box, enter a name for the Deployment Profile (for this sample **COMeWay_MSEExcel_DP**), and select **COMeWay_MSEExcel_Env** as the Environment. Click **OK**.
- 3 From the Deployment Editor toolbar, click the **Auto Map** icon. The projects components are automatically mapped to their system window as seen in Figure 48. If any of the project components are not mapped automatically after Auto Map is used, those component can be mapped manually by following the appropriate steps below. Once all components are mapped, proceed to step 8.
- 4 From the left pane of the Deployment Editor, drag **Service1 -> COM1** to the **ComExtSys** window.
- 5 From the left pane of the Deployment Editor, drag **File1 -> Service1** to the **FileExtSysIn** window.
- 6 From the left pane of the Deployment Editor, drag **Service1 -> File2** to the **FileExtSysOut** window.
- 7 Drag **Service1** to **IntegrationSvr1** in the **LogicalHost1** window (see Figure 48).

Figure 48 Deployment Profile



- 8 Click **Activate**, then save the changes to the Repository.

5.7.12. Running the Project

Designate the Input and Output Directories

Before starting the eWay, create the input and output directories. For this sample, these files are created as follows

- Input directory: **C:\eGate\Data**
- Output directory: **MSEExcel_output%d.dat**.

You can designate any directories as the input and output directories, but the File eWay properties must be configured to use those directories before running the eWay. For more information on configuring these properties see [Configuring the eWays](#) on page 64.

Starting the Project

The following directions assume that the Enterprise Designer was downloaded to C:\ican50. If this is not the case, replace that location in the following directions with the appropriate location.

- 1 From the Enterprise Manager Downloads tab, double-click **Logical Host - for win32**.
- 2 Extract the file to the **C:\ican50\LogicalHost2** directory. You must specify the **LogicalHost2** directory for it to be created.
- 3 Navigate to the **C:\ican50\LogicalHost2\bootstrap\config** directory and open the **logical-host.properties** file using Notepad™.
- 4 Enter and save the following information in the appropriate fields:
 - ♦ Logical Host root directory: **C:\ican50\LogicalHost2**
 - ♦ Repository URL: **http://localhost:port number/repository name**
 - ♦ Repository user name and password: *Your user name and password*
 - ♦ Logical Host Environment name **COMeWay_MSEExcel_Env**
 - ♦ Logical Host name: **Localhost**

Save your changes to **logical-host.properties** and close the file.

- 5 Run the **bootstrap.bat** file located in the **C:\ican50\LogicalHost2\bootstrap\bin** directory.
- 6 Copy the sample input data file to the input directory, with the correct extension.

5.8 Implementing the COMeWay_MSDAO Sample Project

The **COMeWay_MSDAO** (Data Access Object) sample project demonstrates how the eWay, using an OTD created from the file, **dao360.dll**, can access and modify a specified MSAccess MDB file.

5.8.1. Preparing the Imported COMeWay_MSDAO Sample Project

To run the **COMeWay_MSDAO** sample project, do the following:

- 1 Import the sample to the Repository (see [Importing a Sample Project](#) on page 30).
- 2 Create an Environment similar to those created for the other sample projects (see [Creating an Environment](#) on page 53).
- 3 Configure the eWay properties (see [Configuring the eWays](#) on page 54). Edit the input and output directories as suitable to your specific system.
- 4 Create a Deployment Profile similar to those created for the other sample projects (see [Creating and Activating the Deployment Profile](#) on page 56)

5.8.2. Running the Project

Designate the Input and Output Directories

Before starting the eWay, create the input and output directories. For this sample, these files are created as follows

- Input directory: **C:\eGate\Data**
- Output directory: **MSDAO_output%d.dat**.

You can designate any directories as the input and output directories, but the File eWay properties must be configured to use those directories before running the eWay. For more information on configuring these properties see [Configuring the eWays](#) on page 54.

Starting the Project

The following directions assume that the Enterprise Designer was downloaded to C:\ican50. If this is not the case, replace that location in the following directions with the appropriate location.

- 1 From the Enterprise Manager Downloads tab, double-click **Logical Host - for win32**.
- 2 Extract the file to the **C:\ican50\LogicalHost3** directory. You must specify the **LogicalHost3** directory for it to be created.
- 3 Navigate to the **C:\ican50\LogicalHost3\bootstrap\config** directory and open the logical-host.properties file using Notepad™.
- 4 Enter and save the following information in the appropriate fields:
 - ♦ Logical Host root directory: **C:\ican50\LogicalHost3**

- ♦ Repository URL: **http://localhost;port number/repository name**
- ♦ Repository user name and password: **Your user name and password**
- ♦ Logical Host Environment name **COMeWay_MSDAO_Env** (or the name that you specified for the Environment).
- ♦ Logical Host name: **Localhost**

Save your changes to **logical-host.properties** and close the file.

- 5 Run the **bootstrap.bat** file located in the **C:\ican50\LogicalHost3\bootstrap\bin** directory.
- 6 Copy the sample input data file to the input directory, with the correct extension.

Java Methods and Classes for the COM/DCOM eWay

This chapter provides an overview of the Java classes and methods contained in the COM/DCOM eWay. These methods are used to extend the functionality of the eWay.

What's in This Document

- [COM/DCOM eWay Classes](#) on page 70
- [COM/DCOM Javadoc](#) on page 71
- [COM/DCOM Runtime Exceptions](#) on page 71

6.1 COM/DCOM eWay Classes

The COM/DCOM eWay exposes a number of Java methods to extend the functionality of the eWay. These methods are contained in the following class:

- **ComApplication:** Implements ConnectionAssociation.
- **ComApplicationException:** Exception type that can be thrown from methods on the COM AppConn interfaces.
- **ComConfiguration:** Used to access the configuration parameters given by the user.
- **STCComException:** Exception type thrown from methods on the STCDispatchDriver.
- **STCComOptionalArgument:** An extension of the STCComVariant with the appropriate error code.
- **STCComSafeArray:** Used to exchange arrays with the methods of generated COM OTDs.
- **STCComVariant:** Wraps the concept of the COM VARIANT type.
- **STCComVARTYPE:** Defines the VARTYPEs used internally in the implementation.
- **STCHResult:** Wraps the COM HRESULT type.
- **STCIDispatch:** Extends STCIUnknown.
- **BooleanRef:** Used in cases where the method parameter is an [in, out] or [out] type.

- **DoubleRef:** Used in cases where the method parameter is an [in, out] or [out] type.
- **FloatRef:** Used in cases where the method parameter is an [in, out] or [out] type.
- **IntRef:** Used in cases where the method parameter is an [in, out] or [out] type.
- **ShortRef:** Used in cases where the method parameter is an [in, out] or [out] type.
- **STCIDispatchRef:** Used in cases where the method parameter is an [in, out] or [out] type.
- **StringRef:** Used in cases where the method parameter is an [in, out] or [out] type.

6.1.1. COM/DCOM Javadoc

The COM/DCOM eWay Javadoc is an API document in HTML format that provides information on the various methods available with the COM/DCOM eWay. The Javadoc is accessed by selecting and uploading **COMeWayDocs.sar** from the **ADMIN** tab of the Enterprise Manager. The Javadoc can then be accessed from the **Documentation** tab of the Enterprise Manager.

To access the full Javadoc, extract the Javadoc to an easily accessible folder, and double click the **index.html** file.

6.1.2. COM/DCOM Runtime Exceptions

The COM/DCOM eWay can be conceptually divided into two layers. The low-level JNI code that wraps the COM STCIDispatch interface and the high-level OTD code that is generated by the builder.

At the lower level, all methods accessed via the IDispatch interface return an HRESULT error code. In C Programming Language, this type is a long. In Java, it is an integer, but it is wrapped by the STCHresult java class. In general, a value of zero indicates success, greater than zero indicates a warning, and less than zero indicates an error.

To provide the user with more control over this type of situation, an exception type, STCComException, has been added to the low level JNI code. This exception class is derived from the java.lang.RuntimeException. If a method fails on the component (or if the creation of the component fails) an exception of this type is thrown. The exception is passed up through the builder generated OTD code (because the OTD code does not catch it), up to the Collaboration code where the user can catch the exception if desired. Access to the underlying HRESULT is provided. The getMessage method has been overridden and provides a brief contextual string indicating the operation that failed.

Index

A

alerting and logging 18
Auto Map 56, 66

B

bindings 51, 62
BPEL
for COM/DCOM 28
Business Process Execution Language
for COM/DCOM 28

C

Cobol Copybook Converter
sample project 58
CoClasses 19, 20
selecting 23
Collaboration
Editor
Java 34, 60
COM/DCOM
description 6
Javadoc 71
COM/DCOM eWay
Java classes 70
comewayruntime.dll 13
comewayruntime.dll
installing 12, 13
Connectivity Map 31, 58
populating 32, 58
conventions, document 8
create methods 19, 20

D

data types
supported 27
Deployment Profile
Auto Map 56, 66
document
conventions 9
document conventions 8
dynamic configuration 17

E

eInsight
with COM/DCOM 28
Environment 53, 63
exceptions
runtime 71
External Applications 31, 58

H

heap size
adjusting heap memory size 14

I

implementation 26
Information dialog box
unsupported data types 24
installing
COM/DCOM eWay 11

J

Java
Collaboration
using the editor 34, 60
Collaboration Definitions 34, 60
Collaboration Editor 35, 61
Java classes 70
Javadoc 71

L

late binding 7
logging 18

M

methods 70
not supported
skipped methods log 24

O

Object Type Definitions 19
COM OTD Wizard 19
generated 19
operating systems
supported 10
Options Setup
dialog box 14
organization of information 7
OTD wizard

Index

- selecting a type library 23
- selecting CoClasses 23
- using 22

OutOfMemoryError

- increase heap size 14

P

project

- creating 31, 58

properties 54, 64

- comments in Properties Sheet 17
- configuring the COM/DCOM eWay 15
- descriptions in Properties Sheet 17
- editor
 - using 16
- Environment Explorer 15
- overview 15
- Properties Sheet 16

Properties Editor 16

- comments pane 17
- description pane 17

Properties Sheet

- overview 16

Q

query method 21

query methods 19, 20

R

Release() method requirements 20, 28

running the project 57, 67, 68

runtime exceptions 71

Runtime Java API

- adding 13

Runtime JNI bridge DL

- adding 13

S

SAFEARRAY

- 1 or 2 dimensional 27
- constraints 27
- zero-based indexes 27

SAFEARRAYs

- 27
- zero-based indexes 27

sample projects

- descriptions 28, 29
- importing 30

samples

Java Collaboration samples 26

Select Wizard Type 22

skipped methods log 24

stccomruntime.api.jar

- installing 11, 13

stccomruntime.impl.jar

- installing 11, 13

system requirements 10

- external 11

T

type library

- selecting 23

V

VT_BOOL 27

VT_BSTR 27

VT_DATE 27

VT_DISPATCH 27

VT_I2 27

VT_I4 27

VT_R4 27

VT_R8 27

VT_VARIANT 27

W

writing conventions 9

Z

zero-based indexes 27