SUN SEEBEYOND

eWAY™ COM/DCOM ADAPTER USER'S GUIDE

Release 5.1.2

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries. U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties. Sun, Sun Microsystems, the Sun logo, Java, Sun Java Composite Application Platform Suite, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM, eWay, and JMS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés. Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains listés à l'adresse http://www.sun.com/patents et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays. L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties. Sun, Sun Microsystems, le logo Sun, Java, Sun Java Composite Application Platform Suite, Sun, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM et eWay sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd. Ce produit est couvert à la législation américaine en matière de contrôle des exportations et peut être soumis à la règlementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

Part Number: 819-7402-10

Version 20061004084521

Contents

Chapter 1

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

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 Sun SeeBeyond eWay™ COM/DCOM Adapter

The Sun SeeBeyond eWay™ COM/DCOM Adapter (referred to as the COM/DCOM eWay through-out this document) allows the Sun Java Composite Application Platform 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**.

The COM/DCOM eWay does not support eInsight™ Business Processes.

1.3 What's New in This Release

The Sun SeeBeyond eWay COM/DCOM Adapter includes the following changes and new features:

New for Version 5.1.2

- This is a maintenance release. No new features.

New for Version 5.1.1

- This is a maintenance release. No new features.

New for Version 5.1.0

- Version Control: An enhanced version control system allows you to effectively manage changes to the eWay components.
- Multiple Drag-and-Drop Component Mapping from the Deployment Editor: The Deployment Editor now allows you to select multiple components from the Editor's component pane, and drop them into your Environment component.
- Support for Runtime LDAP Configuration: eWay configuration properties now support LDAP URL values.
- Relaunchable OTDs: Allows you to modify existing OTDs from the OTD Wizard.
- Connectivity Map Generator: Generates and links your Project's Connectivity Map components using a Collaboration or Business Process.

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

1.4 What's in This Document

This guide includes the following chapters:

- **Chapter 1 "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.

1.4.1 COM/DCOM eWay Javadoc

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

1.4.2 Scope of the Document

This user's guide provides a description of the 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 Java Composite Application Platform Suite system. This person must also understand any operating systems on which the Java Composite Application Platform Suite will be installed, and must be thoroughly familiar with Windows-style GUI operations.

1.4.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

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

1.5 Sun Microsystems, Inc. Web Site

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

http://www.sun.com

1.6 Documentation Feedback

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

CAPS_docsfeedback@sun.com

Chapter 2

Installing the COM/DCOM eWay

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

What's in This Document

2.1 COM/DCOM eWay System Requirements

The COM/DCOM eWay Readme contains the latest information on:

- Supported Operating Systems
- System Requirements
- External System Requirements
- The COM/DCOM eWay Readme is uploaded with the eWay's documentation file (COMeWayDocs.sar) and can be accessed from the Documentation tab of the Sun Java Integrator Suite Installer. Refer to the COM/DCOM eWay Readme for the latest requirements before installing the COM/DCOM eWay.

2.2 Installing the COM/DCOM eWay

The Sun Java Composite Application Platform Suite Installer, a web-based application, is used to select and upload eWays and add-on files during the installation process. The following section describes how to install the components required for this eWay.

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

2.2.1 Installing the eWay on a JavaCAPS Supported System

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

1. From the Sun Java Composite Application Platform Suite Installer's **Select Sun Java Composite Application Platform Suite Products to Install** table (Administration tab), expand the **eWay** option.
2. Select the products for your Sun Java Composite Application Platform Suite and include the following:
   - **FileeWay** (the File eWay is used by most sample Projects)
   - **COMeWay**

   To upload the COM/DCOM eWay User's Guide, Help file, Javadoc, Readme, and sample Projects, select the following:
   - **COMeWayDocs**
3. Once you have selected all of your products, click **Next** in the top-right or bottom-right corner of the **Select Sun Java Composite Application Platform Suite Products to Install** box.
4. From the **Selecting Files to Install** box, locate and select your first product's SAR file. Once you have selected the SAR file, click **Next**. Your next selected product appears. Follow this procedure for each of your selected products. The **Installation Status** window appears and installation begins after the last SAR file has been selected.
5. Once your product's installation is finished, continue installing the Sun Java Composite Application Platform Suite as instructed in the *Sun Java Composite Application Platform Suite Installation Guide.*

Adding the eWay to an Existing Suite Installation

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

1. Complete steps 1 through 4 above.
2. Once your product's installation is finished, open the Enterprise Designer and select **Update Center** from the Tools menu. The Update Center Wizard appears.
3. For Step 1 of the wizard, simply click **Next**.
4. For Step 2 of the wizard, click the **Add All** button to move all installable files to the **Include in Install** field, then click **Next**.
5. For Step 3 of the wizard, wait for the modules to download, then click **Next**.
6. The wizard's Step 4 window displays the installed modules. Review the installed modules and click **Finish.**
7. When prompted, restart the IDE (Integrated Development Environment) to complete the installation.

2.2.2 Installing eWay Enterprise Manager plug-ins

The **Sun SeeBeyond Enterprise Manager** is a Web-based interface that allows you to monitor and manage your Sun Java Composite Application Platform Suite applications. The Enterprise Manager requires an eWay specific "plug-in" for each of your installed eWays. These plug-ins enable the Enterprise Manager to target specific alert codes for each eWay type, as well as to start and stop the inbound eWays.

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

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

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

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

COM/DCOM eWay Alert Codes

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

To View the eWay Alert Codes

1. Add the eWay Enterprise Manager plug-in for this eWay.
2. From the Enterprise Manager's **Explorer** toolbar, click the **Configuration** icon.

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

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

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

Table 2 COM/DCOM eWay Alert Codes

Alert Code	Description	User Action
COM-CREATECOMOBECTJFAILED000003	Failed to create COM object instance with CLSID {0}	The alert my be caused by one of the following: ▪ COM Server may not be installed. Verify the COM server is running.>Verify that parameters are correct.
COM-NOMOREACCESSHANDLES000002	No more concurrent access handles available	Insufficient resources to run Collaboration. ▪ Consider "load-balancing" by deploying you Project on multiple hosts. ▪ Try re-deploying your Project. ▪ Verify that your configuration parameters are valid.
COM-UNABLETOGETCOMRUNTIME000001	Failed acquire COM runtime environment	▪ The Runtime JNI and DLL may not be available. Verify that Runtime JNI and DLL are installed in the correct directory (see **"Download and Install Additional COM/DCOM eWay Files" on page 14**).
COM -UNKNOWNAPPNAME000004	Unrecognized application name {0}	This alert code is reserved for future enhancement.

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

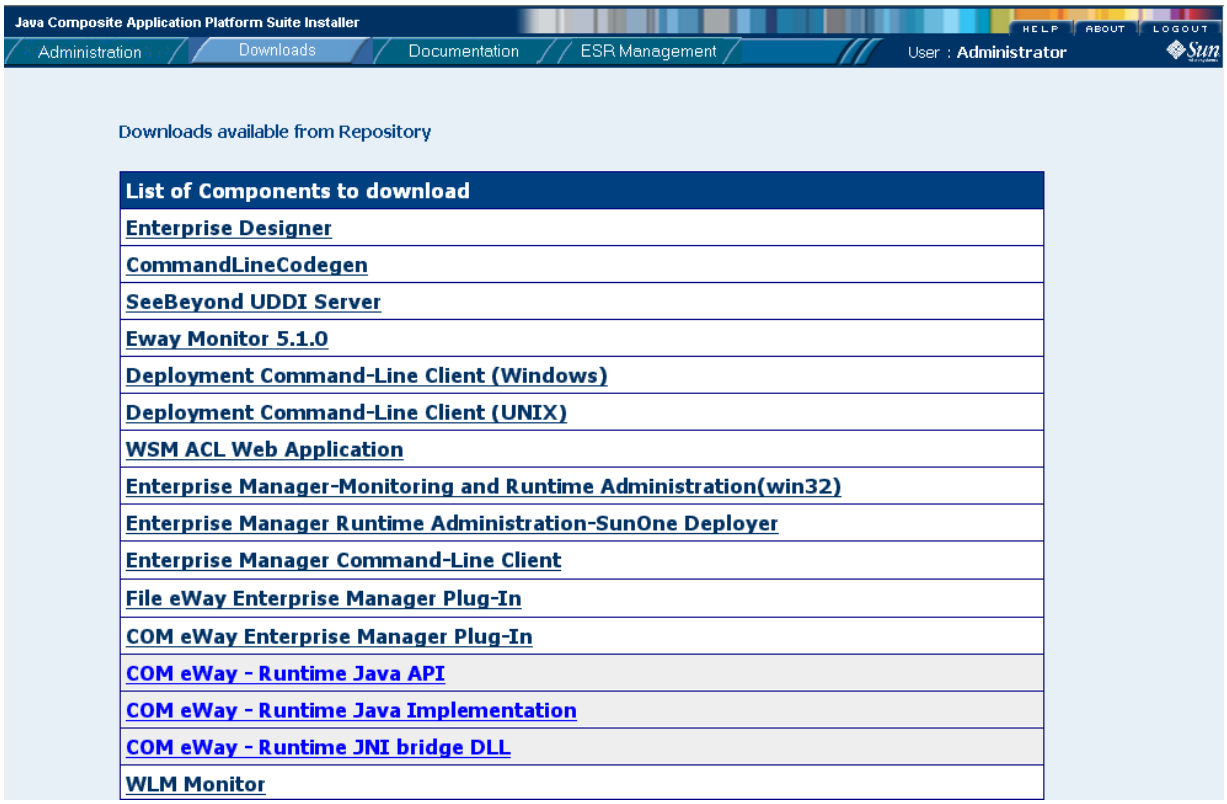
2.2.3 Download and Install Additional COM/DCOM eWay Files

The COM/DCOM eWay requires three additional files to be downloaded and applied.

1 Click on the Sun Java Composite Application Platform Suite Installer's **DOWNLOADS** tab. The Component list, as displayed in **Figure 1 on page 14**, 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 on page 14**).

Figure 1 Sun Java Composite Application Platform Suite Installer - DOWNLOADS



2 Download all three components to a local directory. To add the JAR files to the Integration Server classpath see **"Adding JAR files to the Integration Server" on page 15**. To add the DLL file to the PATH for the Logical Host process, see **"Adding the DLL file to the Path for the Logical Host Process" on page 15**

3 Continue installing the eGate Integrator as instructed in the *Sun Java Composite Application Platform Suite Installation Guide*.

Adding JAR files to the Integration Server

Prior to building your COM/DCOM eWay Project, copy the **Runtime Java API** (stccomruntime.api.jar) and **Runtime Java Implementation** (stccomruntime.impl.jar), downloaded from the Sun Java Composite Application Platform Suite Installer, to the Sun Java Composite Application Platform Suite in the following location:

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

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

These file must be copied manually.

Adding the DLL file to the Path for the Logical Host Process

The runtime JNI bridge DLL file, **comewayruntime.dll**, must be added to the PATH for the Logical Host process before running a COM/DCOM Project. To make sure that the correct file is added to the proper PATH location, do the following before launching your Project:

1. Download the runtime JNI bridge DLL file, **comewayruntime.dll**, to a temporary directory.
2. Copy **comewayruntime.dll** to the following location:

```
C:\winnt\system32
```

An alternative to this procedure is to add the JNI bridge DLL file to the library path using the **Integration Server Administration** console and specifying the location of the downloaded **comewayruntime.dll** in the library path. For more information about the Integration Server Administration Configuration Agent, see the Sun SeeBeyond eGate™ Integrator System Administration Guide.

After Installation

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

2.2.4 Editing the LogicalHost server.policy File

The LogicalHost requires additional code (an additional permission) added to the server.policy, located in the following directory:

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

where <*JavaCAPS51*> is the Sun Java Composite Application Platform Suite install directory.

This is required before you create a domain. If this additional permission is not added, the eWay will throw the following exception at runtime:

```
Error executing business rule: access denied
(java.lang.RuntimePermission shutdownHooks)
```

To add this additional code to the server.policy file, do the following:

1 Open the server.policy file with a text editor.

2 Find the following statement:

```
// Basic set of required permissions granted to all remaining code
grant {
```

Add the following of code to the section under the above statement:

```
// ICAN COM eWay
permission java.lang.RuntimePermission "shutdownHooks";
```

3 Save your changes to the server.policy file.

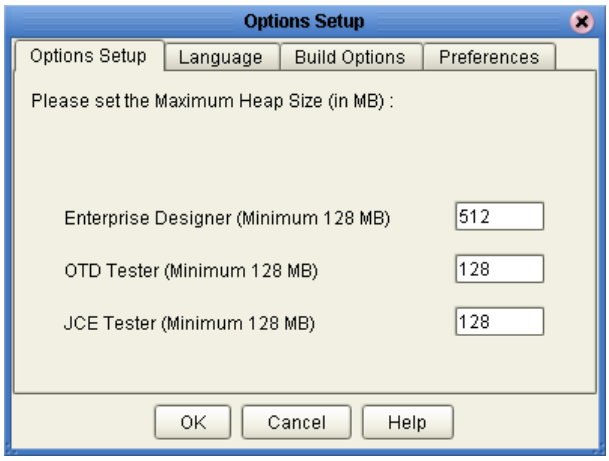
2.2.5 Increasing the Enterprise Designer Heap Size

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.

Figure 2 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 ***<JavaCAPS51>*\edesigner\bin** (where *<JavaCAPS51> is the Sun Java Composite Application Platform Suite install directory). Open **heapSize.bat** with a text editor and change the heap size setting to **512**. Save the file, and restart the Enterprise Designer.*

2.3 ICAN 5.0 Project Migration Procedures

This section describes how to transfer your current ICAN 5.0 Projects to Sun Java Composite Application Platform Suite, version 5.1.2. Only Projects developed on ICAN 5.0.2 and above can be migrated successfully to the Sun Java Composite Application Platform Suite. To migrate your ICAN 5.0 Projects, do the following:

Export the Project

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

Install Sun Java Composite Application Platform Suite

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

Import the Project

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

Deploy the Project

13. You must create a new Deployment Profile for each of your imported Projects. When you export a Project, the Project's components are automatically *"checked in"*

to Version Control to write-protect each component. These protected components appear in the Explorer tree with a red padlock in the bottom-left corner of each icon. Before you can deploy the imported Project, the Project's components must first be *"checked out"* of Version Control from both the Project Explorer and the Environment Explorer. To *"check out"* all of the Project's components, do the following:

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

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

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

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

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

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

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

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

15 Deploy your Projects.

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

Chapter 3

Configuring the eWay Properties

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

What's in This Document

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.

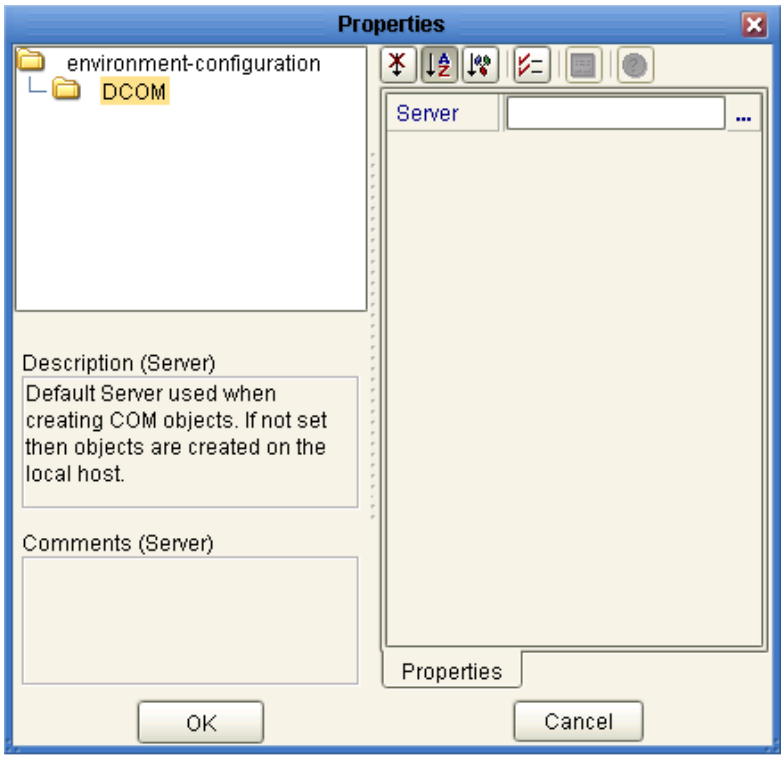
3.1.2 Using the Properties Editor

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 3.

Figure 3 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 **DCOM** section of the COM/DCOM Environment properties contains the top-level parameter displayed in Table 3.

Table 3 Environment Properties - DCOM

Name	Description	Required Value
Server	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).	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 21** 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.

Chapter 4

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

4.1 Object Type Definitions

The basic functionality of eGate is to handle messages. It does this by means of Collaborations. To operate by way of messages, 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, 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

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

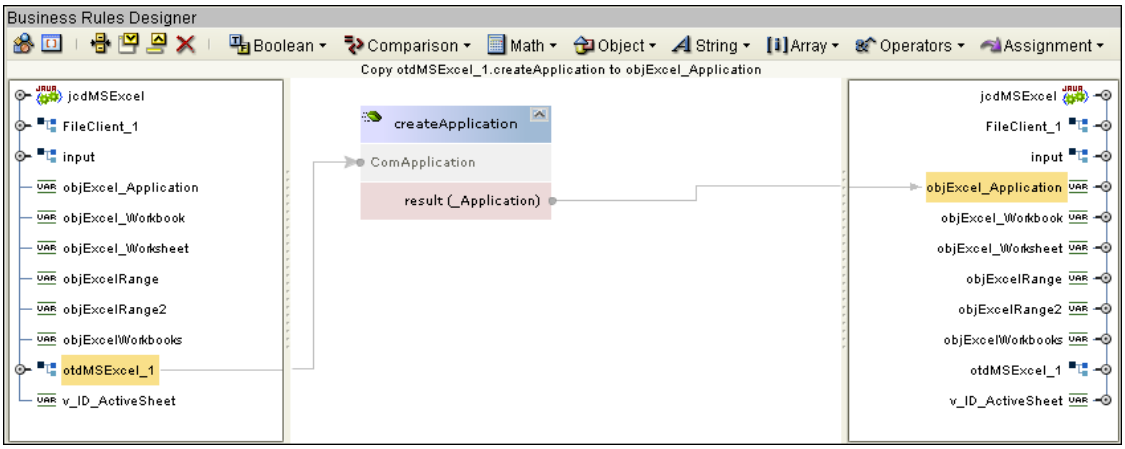
In COM, when you create an instance of a COM object, you get back is an interface for that object. The Create methods create instances of COM objects. When the OTD wizard scans the selected Type Library, it displays the Library's CoClasses. For each CoClass you select in the wizard (see **"Select Classes" on page 26**), 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 **jcdCOMeWay_MSExcel** sample Project's **jcdMSExcel** Collaboration.

1 In the Business Rule, **Copy otdMSExcel_1.createApplication to objExcel_Application**, the user calls the **createApplication()** method to create an instance of the Excel application (see Figure 4).

Figure 4 Copy otdMSExcel_1.createApplication to objExcel_Application Business Rule



2 From this instance interface, the user in 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

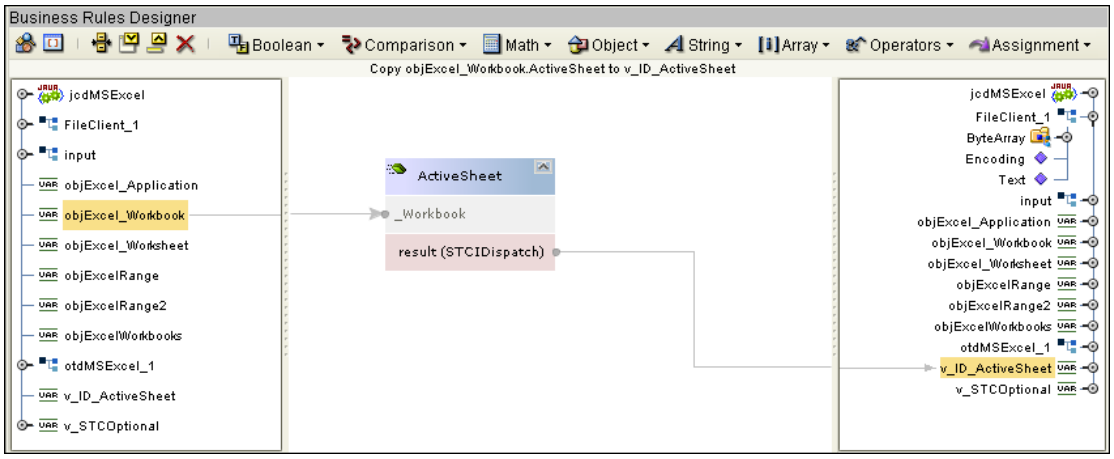
each interface found in the Type Library, a Query method is created on the OTD. You can then pass one of the IDispatch interfaces and convert it to the appropriate interface. Query methods returns the appropriate interfaces with relevant, useful, Java methods.

Example of Query Method

An example of Query method used in a Collaboration can be seen in the **jcdCOMeWay_MSExcel** sample Project's **jcdMSExcel** Collaboration Business Rules.

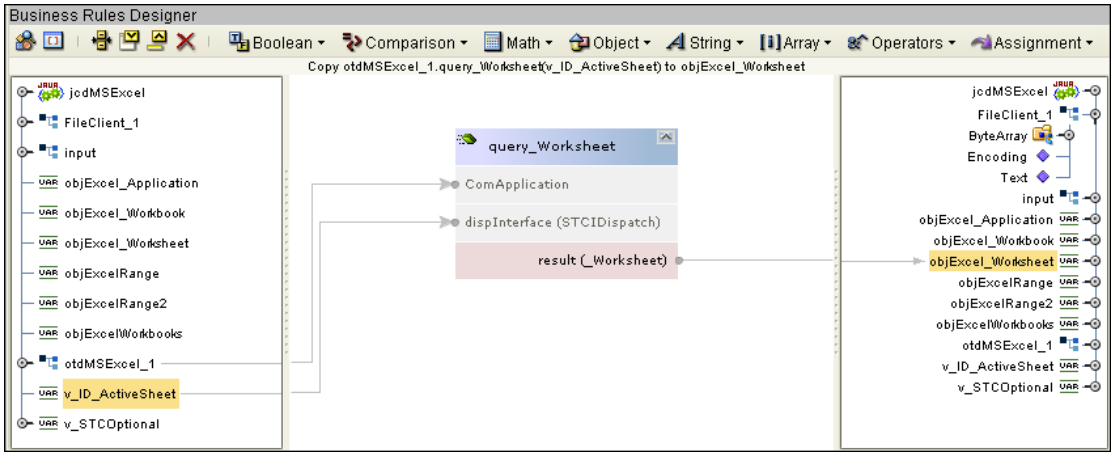
1 The user wants to select and copy an active Excel worksheet. In the Business Rule, **Copy objExcel_Workbook.ActiveSheet to v_ID_ActiveSheet**, When the objExcel_Workbook method, **ActiveSheet()** is called, the **STCIDsipatch** interface is returned (as seen in Figure 5) providing limited functionality.

Figure 5 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 6).

Figure 6 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

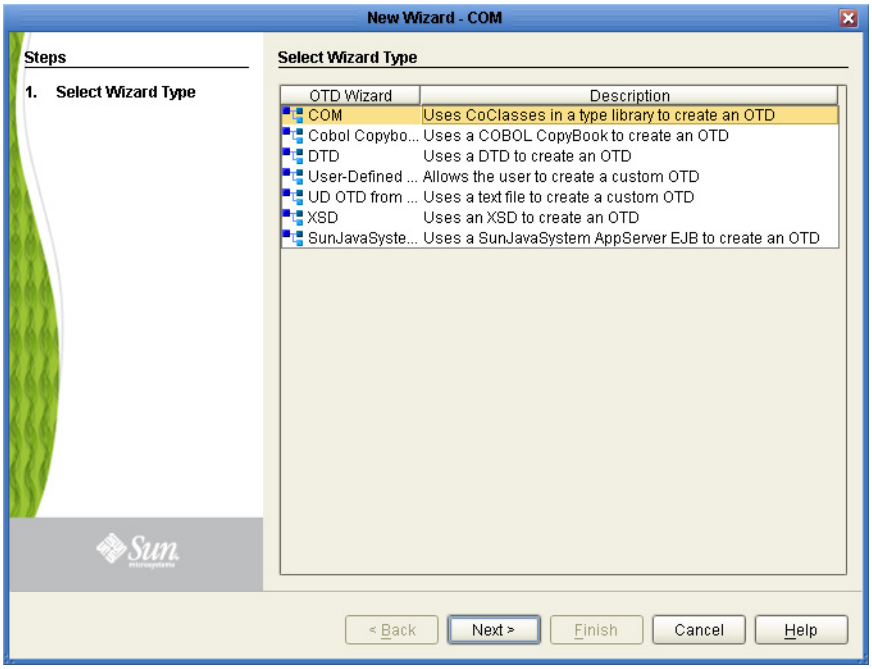
The COM OTD Wizard generates an OTD from a COM automation-compatible component's **Type Library** files. COM Type Library files describe the methods and properties exposed from an automation-compatible component. COM type libraries may have the file extension **.tlb** or **.olb**, however, most components 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 7 on page 25**.

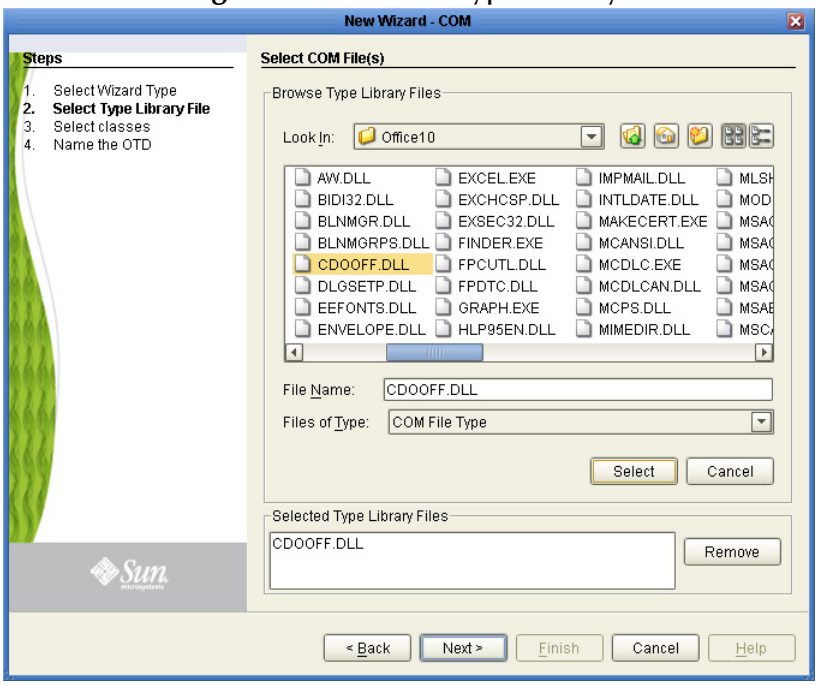
Figure 7 OTD Wizard Selection



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 8). Click **Next**.

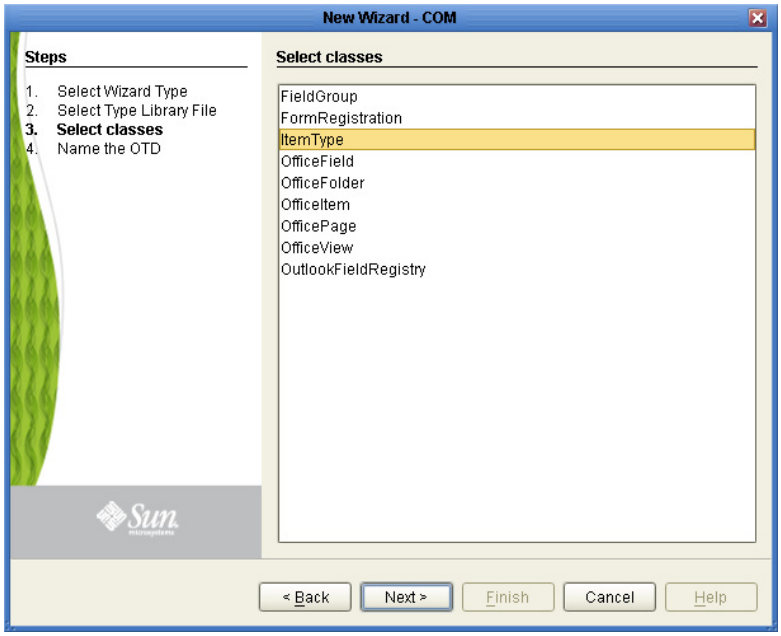
Figure 8 Select the Type Library



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 9 on page 26**).

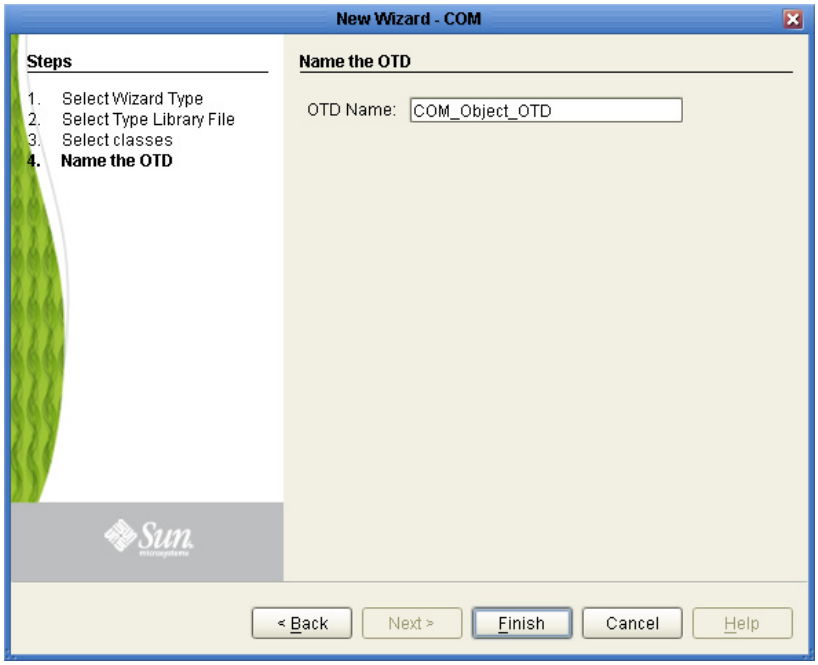
Figure 9 Select Classes



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 10).

Figure 10 Name the OTD



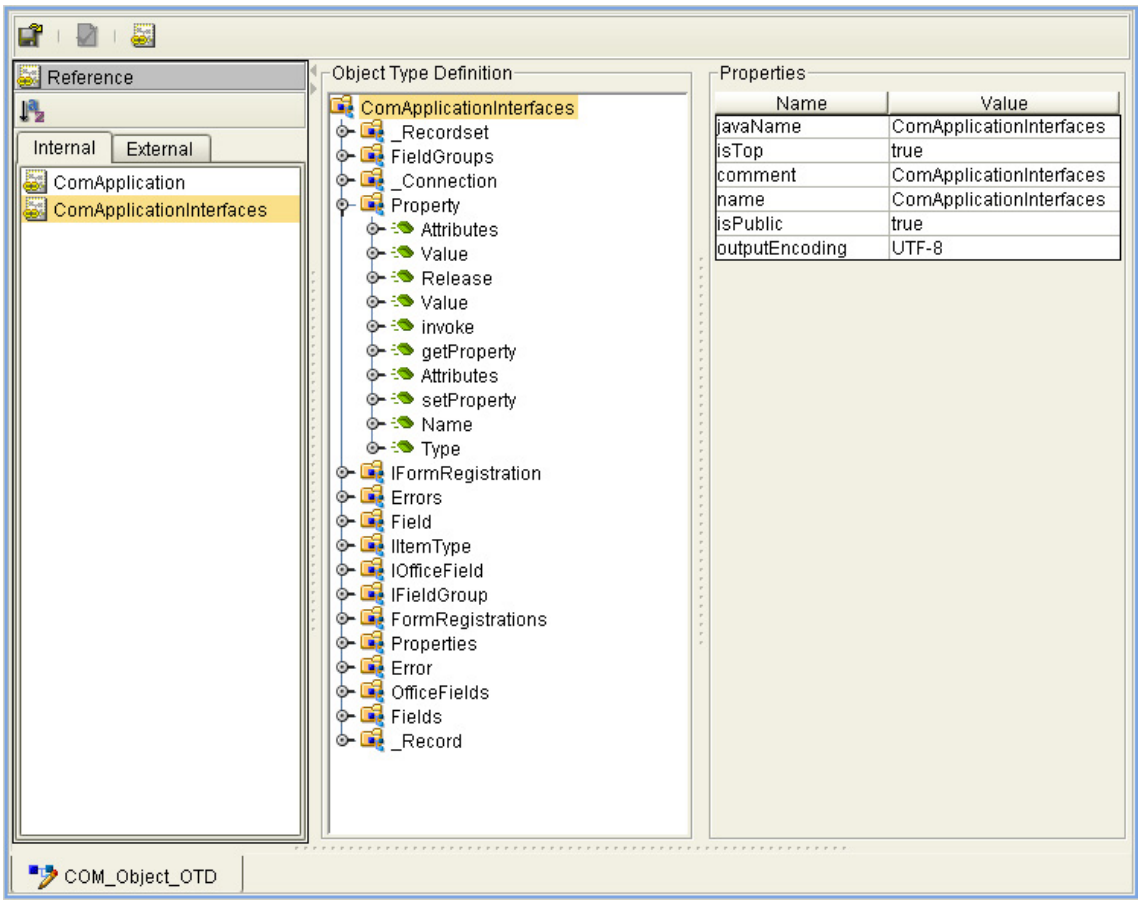
6 If any of the selected CoClasses contain a method with an unsupported data type, an **Information** box appears (see Figure 11).

Figure 11 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 12 on page 28**).

Figure 12 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 *Sun SeeBeyond eGate™ Integrator User's Guide*.

4.3 Relaunching OTDs

A single OTD can consist of many lines of metadata. When a change to the metadata occurs in an OTD, it does not have to be recreated from scratch. Using the **Relaunch** function allows the OTD to be rebuilt and saved under the same name, then relaunched back to the same java collaboration or BPEL.

To Relaunch an Existing OTD

1. In the Enterprise Explorer, right-click on the OTD. From the submenu, click **Relaunch**. The Select Files Wizard opens.
2. Enter the File Name (or Browse and Select) that you wish to be relaunched and click **Next**.
3. Continue with the Wizard as described when creating the OTD.

4 Click the **Finish** button to save the changes.

When relaunching an OTD, an existing collaboration will not be affected if:

- New columns are added
- Deleted columns are not used in the original collaboration

Note: *Validation will fail if existing collaborations are not modified when columns are renamed or deleted.*

Chapter 5

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 Enterprise Designer. For more information on creating an eGate Project see the *Sun SeeBeyond eGate™ Tutorial* and the *Sun SeeBeyond eGate™ Integrator User's Guide*.

What's in This Document

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 *Sun SeeBeyond 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 19**.

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 22**).

5.2 Supported Data Types

The COM/DCOM eWay supports the following data types:

Table 4 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 allows SAFEARRAYs to use starting indexes other than zero, Java supports zero-based indexes only. The COM/DCOM eWay runtime re-maps the received arrays as "0-based" so that they can be used in Java. Information regarding the original starting index for the array is lost. The integration server log file enters a warning message with details about any re-mapping performed and the array indices.

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 created custom COM objects must not match any names of the final methods on the Java Object class (for example, the **wait()**, **notify()**, and so forth). If the COM object contains these methods it will generate a compile error.
- Characters used in COM Type Library names, that are not acceptable in Java, such as dots (“**.**”) in the Java namespace, are translated as underscores. For example: **com.seagull.fileInquiry** is translated `as` **com_seagull_fileInquiry**.
- The COM/DCOM eWay is a “JCD (Java Collaboration Definition) only” eWay, and does not directly support Business Process Execution Language (BPEL) for Web Services or the Sun Java Composite Application Platform Suite’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 Project

The COM/DCOM eWay includes one sample Project, **prjCOM_JCD.zip**, that demonstrates how the eWay is implemented in a production environment.

The sample Project, when imported to your Repository, is nearly complete. It only requires that you create the Environments, Deployment Profiles, Domains, and configure the eWay properties for your system (Environments must be configured to work with your specific system).

The **prjCOM_JCD** sample project demonstrates the following operations:

MSWord Sample

The **MSWord** sample operation demonstrates the following:

1. An input file triggers the eWay to connect to the COM server (the eWay uses your local host as the Com server). This input file contains the location and name of the Word file.
2. The eWay searches the server for a specified application with a specific version number.
3. The eWay then creates a new Word document and adds the specified text to the document.
4. The new document appears. After a few seconds the document is saved and closed.
5. The eWay writes the application name and version to the output file and publishes the file to an external directory.

The **prjCOM_JCD** Project includes the following files for the MSWord sample operation:

- inputWord.fin.~in: The input (trigger) file. This is a text file that contains the location and file name where the testing doc will be saved.
- **COMeWay_WordTest.doc**: The testing doc.

MSExcel Sample

The **MSExcel** sample operation demonstrates the following:

1. An input file triggers the eWay to connect to the COM server (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 selects the specified Excel file from the input directory 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 **prjCOM_JCD** Project includes the following files for the MSExcel sample operation:

- **inputExcel.fin.~in**: The sample input file.
- MDB used to create the otdMSExcel OTD.

MSDAO Sample

The **MSDAO** (Data Access Object) sample operation demonstrates the following:

1. An input file triggers the eWay to connect to the COM server (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 **prjCOM_JCD** Project includes the following files for the MSDAO sample operation:

- **inputMSDAO.fin.~in**: The sample input file.
- **DAOTestDB.mdb**: MDB used to create the otdMSDAO360 OTD.

5.5 Importing a Sample Project

The Sample eWay Project is included as part of the eWay's documentation installation. To import a sample eWay Project to the Enterprise Designer do the following:

1 The sample files are uploaded with the eWay's documentation SAR file and downloaded from the Sun Java Composite Application Platform Suite Installer's Documentation tab. The **prjCOM_JCD.zip** file contains the sample Project ZIP file and sample data. Extract the samples to a local file.

2 Save all unsaved work before importing a Project.

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

4 Browse to the directory that contains the sample Project zip file. Select the sample file (for example, **prjCOM_JCD.zip**) and click **Import**. After the sample Project is successfully imported, click **Close** or select another Project to import.

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

- Create an Environment (see **"Create the Environment" on page 63**)
- Configure the eWays for your specific system (see **"Configuring the eWay Properties" on page 19**)
- Create a Deployment Profile (see **"Creating the Deployment Profile" on page 66**)
- Create and start a domain (see **"Creating and Starting a Domain" on page 67**)
- Build and deploy the Project (see **"Building and Deploying the Project" on page 68**)

5.6 Creating the prjCOM_JCD Project

The sample Projects provided with this eWay require very little setup. The following sections in this chapter are provided to demonstrate how the **prjCOM_JCD** sample Project's components are created manually.

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, **prjCOM_JCD**).

5.6.2 Creating the OTDs

The prjCOM_Sample_JCD Project uses three OTDs created using the COM OTD Wizard.

Create 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.

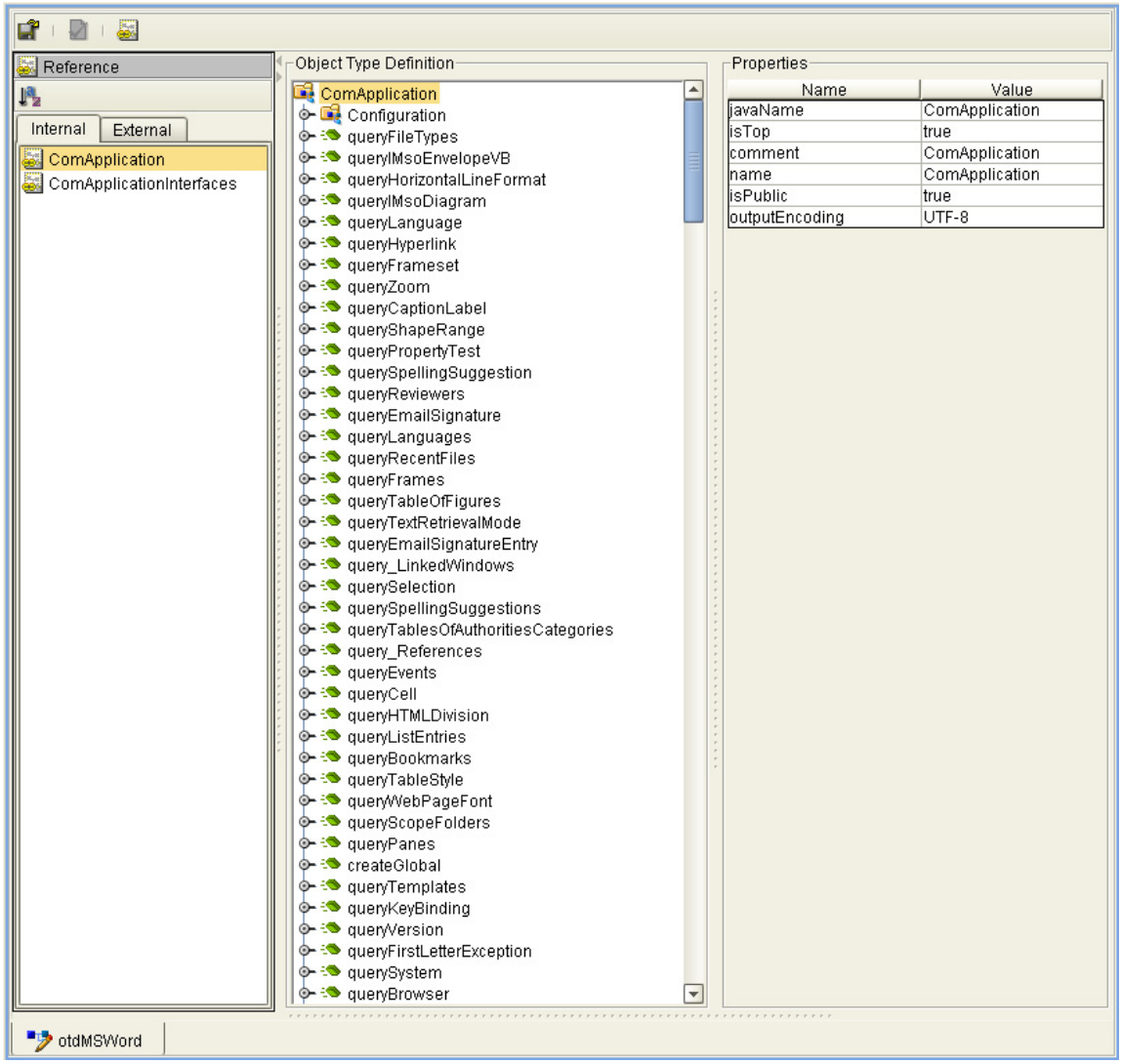
To create an OTD using the COM OTD Wizard do the following:

1. From the Project Explorer tree, right click **prjCOM_Sample_JCD** 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, 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 13 on page 36**).

Figure 13 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.

Create the otdMSExcel OTD

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 **prjCOM_Sample_JCD** 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 **otdMSExcel** 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.

The OTD Editor appears displaying the new **OTD.** The new **otdMSExcel** OTD is added to the Project Explorer tree. The OTD is now available to use in a Collaboration.

Create the otdMSDAO360 OTD

The COM Type Library file used for this sample is the DAOTestDB.mdb provided with the sample. To create an OTD using the COM OTD Wizard do the following:

1 From the Project Explorer tree, right click **prjCOM_Sample_JCD** 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 where you extracted your sample Project files and select the **DAOTestDB.mdb** file. 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 **otdMSDAO360l** 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.

The OTD Editor appears displaying the new **OTD.** The new **otdMSDAO360l** 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 25**.

5.6.3 Creating the Collaboration Definitions

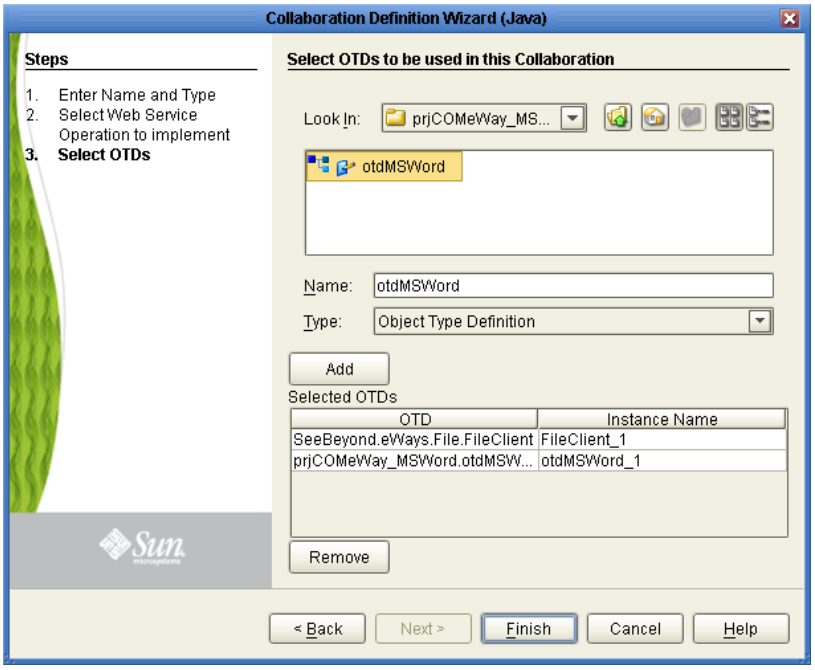
The next step in the sample is to create a Collaboration using the Collaboration Definition Wizard (Java). Once a Collaboration Definition has been created, the Business Rules of the Collaboration are written using the Collaboration Editor (Java).

Create the jcdMSWord Collaboration (Java)

The **jcdMSWord** 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 **prjCOM_Sample_JCD** 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, **jcdMSWord**) and click Next.
3. For Step 2 of the wizard, **Select a Web Services Operation**, double-click **Sun 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 **Sun 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 **prjCOM_Sample_JCD** > **otdMSWord**. The **otdMSWord** OTD is added to the Selected OTDs field (see **Figure 14 on page 38**).

Figure 14 Collaboration Definition Wizard (Java) - Select OTDs



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

5.6.4 Create the jcdMSWord Collaboration Business Rules

The next step in the sample is to create the Business Rules of the Collaboration using the Collaboration Editor. The the prjCOM_Sample_JCD sample Project uses one Collaboration created in the previous section, **jcdMSWord**.

The **jcdMSWord** Collaboration contains the Business Rules displayed in **Figure 15 on page 39**.

Figure 15 jcdMSWord Collaboration Business Rules

Note: *Business Rules are wrapped in Figure 15 for display purposes only.*

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

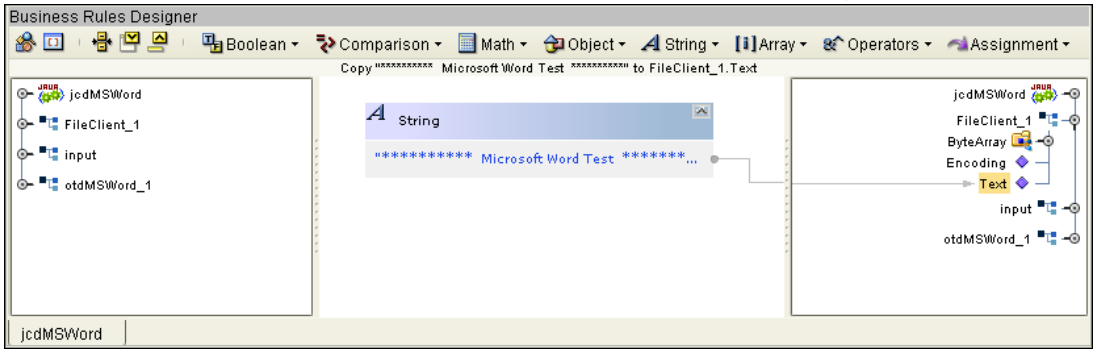
1 From the Project Explorer tree, double-click **jcdMSWord to open the** Collaboration Editor (Java) to the **jcdMSWord** Collaboration.

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. Create this rule:

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 16).

Figure 16 Collaboration Editor (Java) - Business Rules Designer



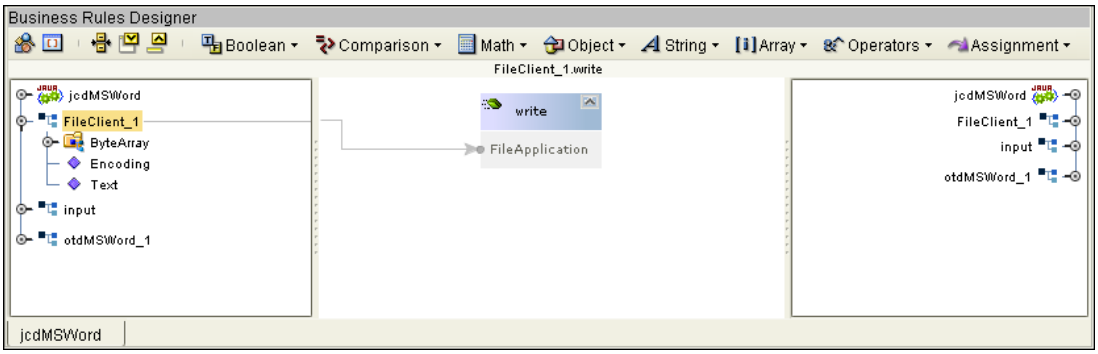
3 Create the **FileClient1_Write** rule:

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 17).

Figure 17 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** bock, are released by the three "release" rules in the **finally** block. Create this rule:

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.

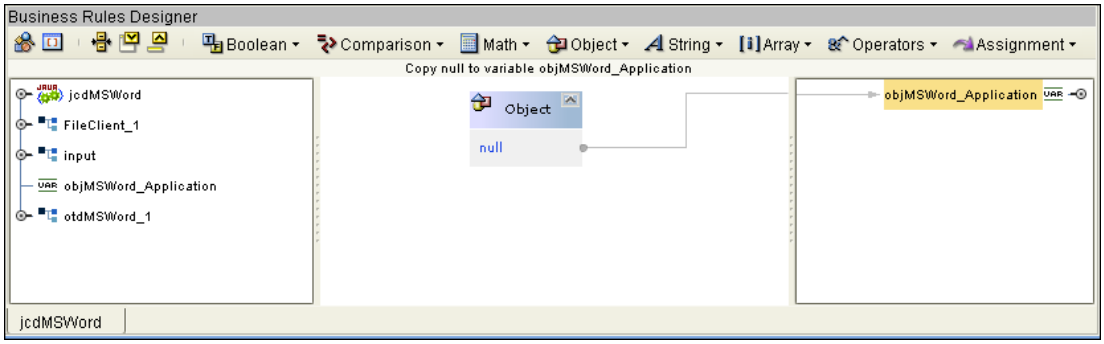
D Select **_Application** under **All Classes**, and click **OK**.

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 18).

Figure 18 Collaboration Editor (Java) - Business Rules Designer



5 Create the **Copy null to variable objMSWordDocuments** rule:

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 Create the **Copy null to variable objMSWord_Document** rule:

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. Create the **Try** block:

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. Create this rule:

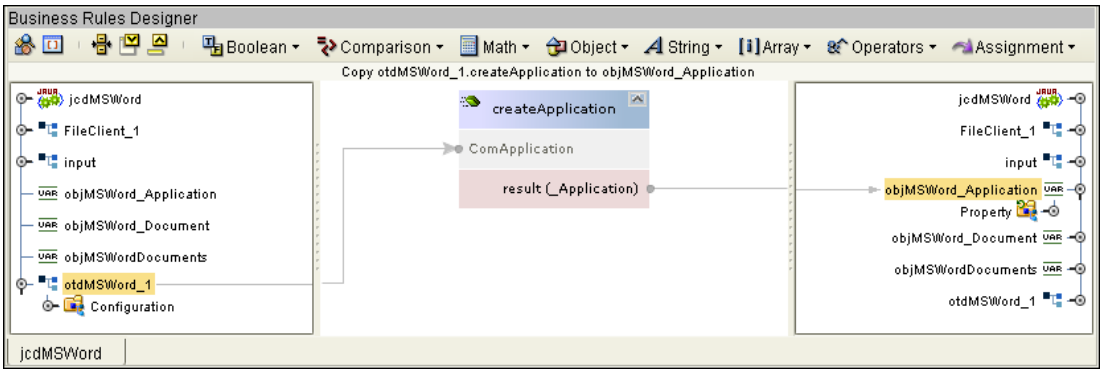
A From the Business Rules tree, select **rules** under the **Try** block and click **rule** on the Business Rules toolbar to add a new rule.

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 19).

Figure 19 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. Create this rule:

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 **name()** from the method selection window. The **Name** method box appears in the Business Rules Designer canvas.

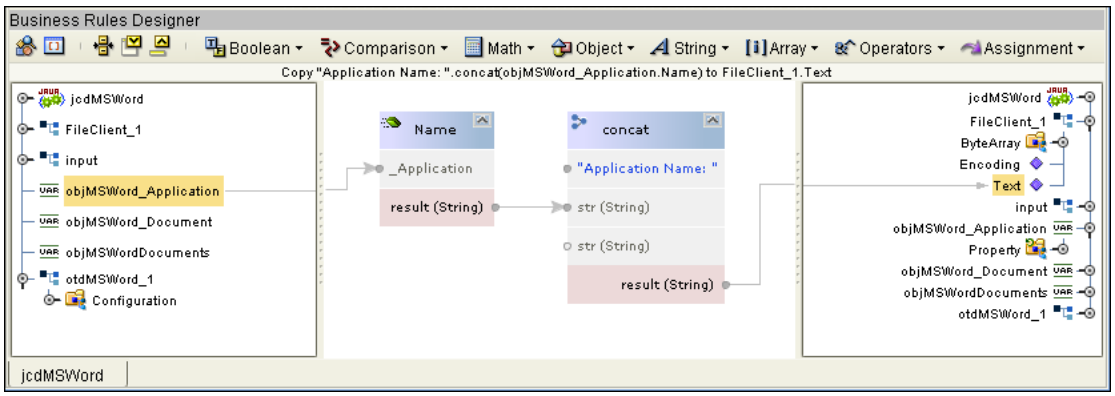
D From the Business Rules Designer String menu, select **concat**. The **concat** method box appears.

E Right-click the String input node of the **Concat** method box and select **Add Literal** from the shortcut menu. Enter **Application Name:** as the **Literal** value.

F Map the **result (String)** output node of the **Name** method box to the **str (String)** input node of the **concat** method box.

G 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 20).

Figure 20 Collaboration Editor (Java) - Business Rules Designer



10 Create the **FileClient1_Write** rule:

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. Create this rule:

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 **version()** from the method selection window. The **Version** method box appears in the Business Rules Designer canvas.

D From the Business Rules Designer String menu, select **concat**. The **concat** method box appears.

E Right-click the String input node of the **Concat** method box and select **Add Literal** from the shortcut menu. Enter **Application Name:** as the **Literal** value.

F Map the **result (String)** output node of the **Version** method box to the **str (String)** input node of the **concat** method box.

G 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 20).

12 Create the next **FileClient_1.write** rule, by following the steps for **Step 10** on page 43.

13 The **objMSWord_Application.Visible(true)** rule allows the user to see the application running. Create this rule:

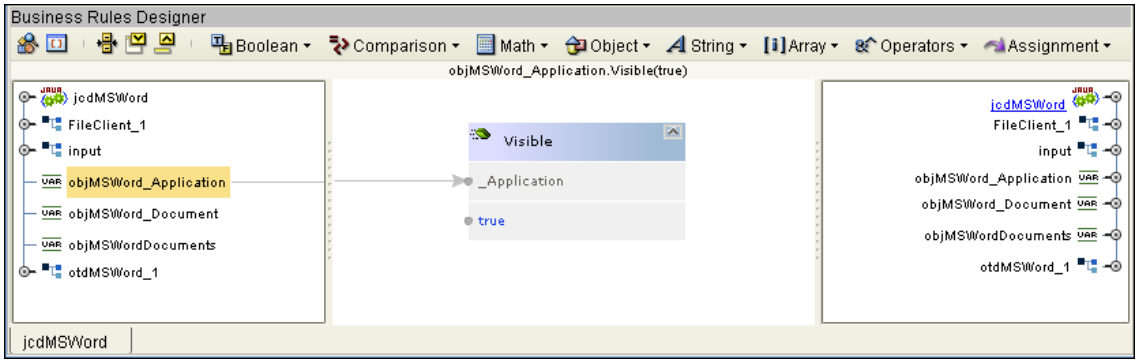
A Click **rule** on the Business Rules toolbar to add a new rule to the Business Rules tree.

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 Right-click the newVisibleValue (boolean) input node of the Visible method box. Select true as the Boolean value (see Figure 21).

Figure 21 Collaboration Editor (Java) - Business Rules Designer



14 The **Copy objMSWord_Application.Documents to objMSWordDocuments** rule creates a new Word document. Create this rule:

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. Create this rule:

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.

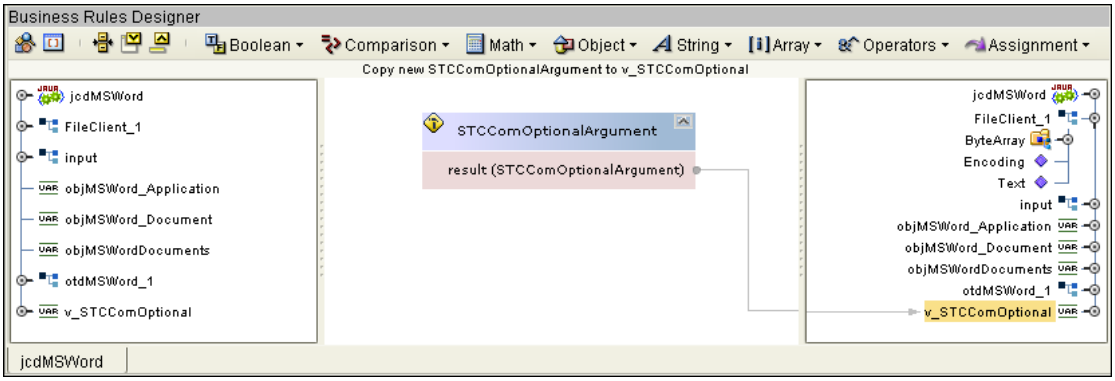
16 The **Copy new STCComOptionalArgument to v_STCComOptional** rule copies the optional argument to the methods. Create this rule:

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 22).

Figure 22 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. Create this rule:

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.

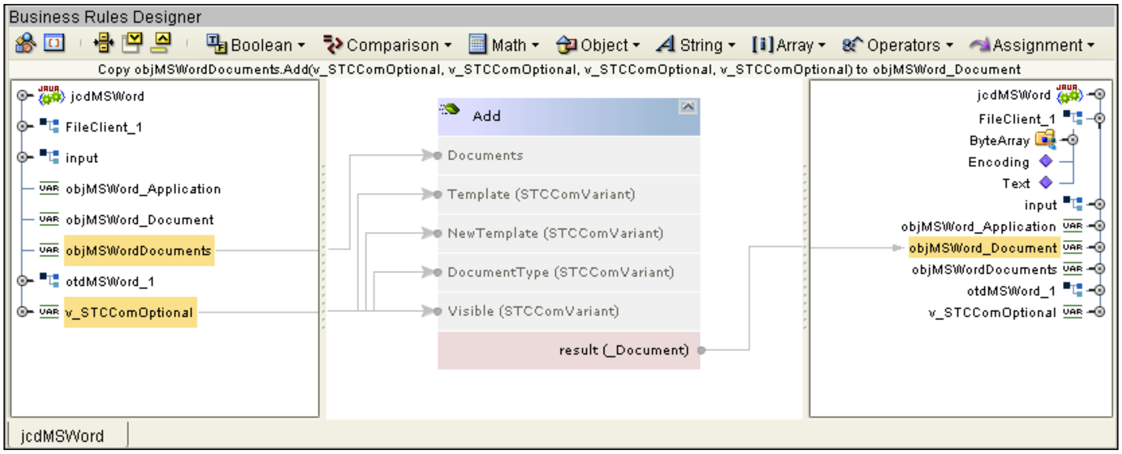
C Select **Add(com.stc.connector.comadapter.STCComVariant Template...)** from the method selection window. The **Add** method box appears.

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 23).

Figure 23 Collaboration Editor (Java) - Business Rules Designer



18 The **objMSWord_Application.Selection.TypeText("*************************
***************\nTesting from Sun SeeBeyond COM
eWay\n*************************
***************")** rule adds this text string to the document. Create this rule:

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.

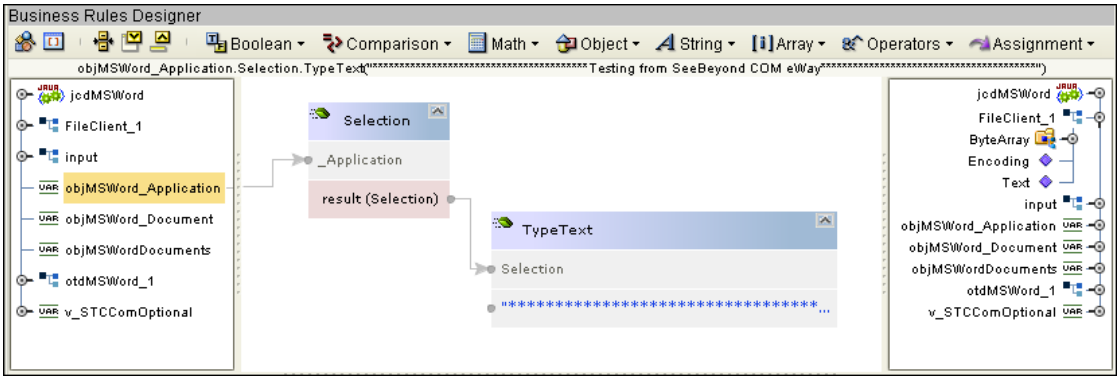
D Right-click the **result (Selection)** output node of the **Selection** method box, and click **Select method to call** from the shortcut menu.

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 Sun SeeBeyond COM eWay\n** as the value.

G Map the output node of the **String** method box to the **Selection** input node of the TextType method box (see Figure 24).

Figure 24 Collaboration Editor (Java) - Business Rules Designer



19 The **Thread.sleep(3000)** rule adds a three second delay so that the user can watch the operation as the document is created. Create this rule:

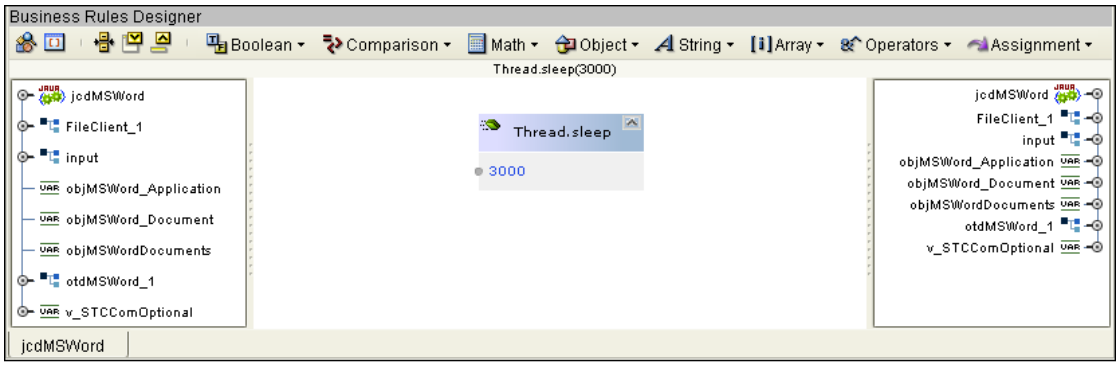
A Click **rule** on the Business Rules toolbar to add a new rule.

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 Double-click the **arg0 (long)** input node of the **Thread.sleep** method box and enter **3000** as the value. (see **Figure 25 on page 48**).

Figure 25 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. Create this rule:

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. Create this rule:

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.

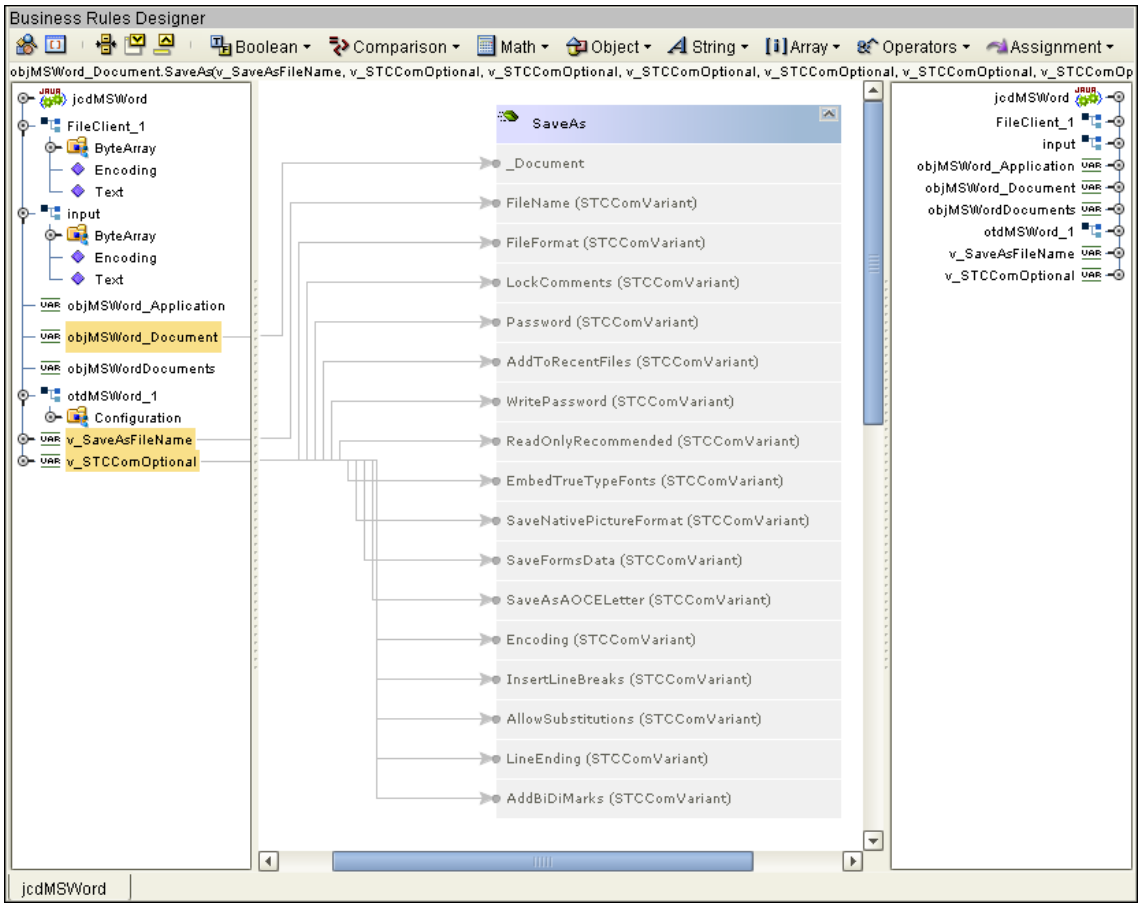
D Map the **v_SaveAsFileName** field in the left pane of the Business Rules Designer, to the **FileName (STCComVariant)** input node of the **SaveAs** method box.

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 26 on page 50**.

- 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 26 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. Create this rule:

A Click **rule** on the Business Rules toolbar to add a new rule.

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 Double-click the **arg0 (long)** input node of the **Thread.sleep** method box and enter **1000** as the value.

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.

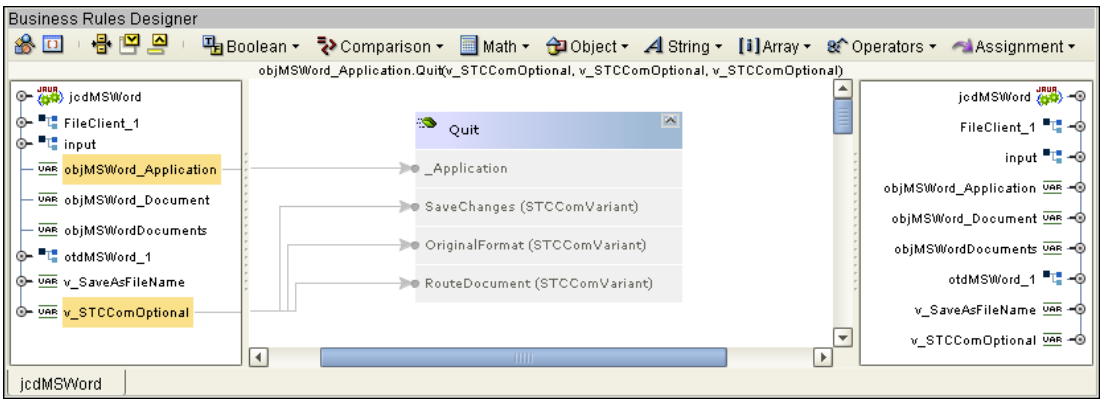
B Right-click the **objMSWord_Application** field in the left pane of the Business Rules Designer, and click **Select method to call** from the shortcut menu.

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 27).

- SaveChanges (STCComVariant)
- OriginalFormat (STCComVariant)
- RouteDocument (STCComVariant)

Figure 27 Collaboration Editor (Java) - Business Rules Designer



25 The **STCComException Catch block** provides error handling for the Collaboration. Create this rule:

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 **ex** as the **Name**.

D For **Type**, click the **Class** field's ellipsis (...) button. The **Class Browser** dialog box appears.

E From the Class Browser dialog box, select **STCComException** under **All Classes**, select **STCComException(int hr, String sOp)** under **STCComException**, and click **Select**.

F Click OK to close the **Create Exception Variable** dialog box.

26 The **logger.error("Exception encountered: ", new Throwable(ex.Message))** rule under **catch**, catches an error if one is thrown in the **Try** block, and writes the error to the log file. Create this rule:

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.

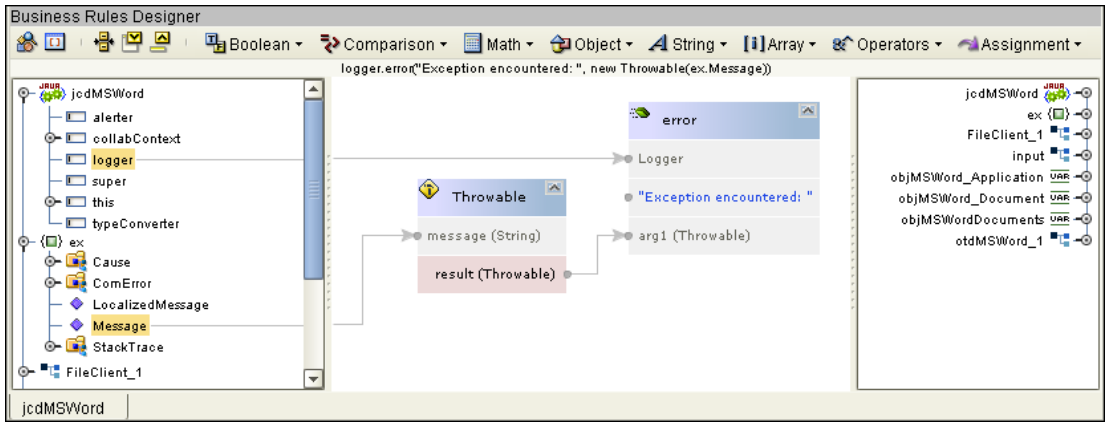
D Select **error(object arg0, Throwable arg1)** from the method selection window. The **error** method box appears.

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 28).

Figure 28 Collaboration Editor (Java) - Business Rules Designer



27 The **If** Statement, the **condition: 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. Create the **If** Statement and the **condition: objMSWord_Document is not equal to null** rule:

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. Select the **condition**.

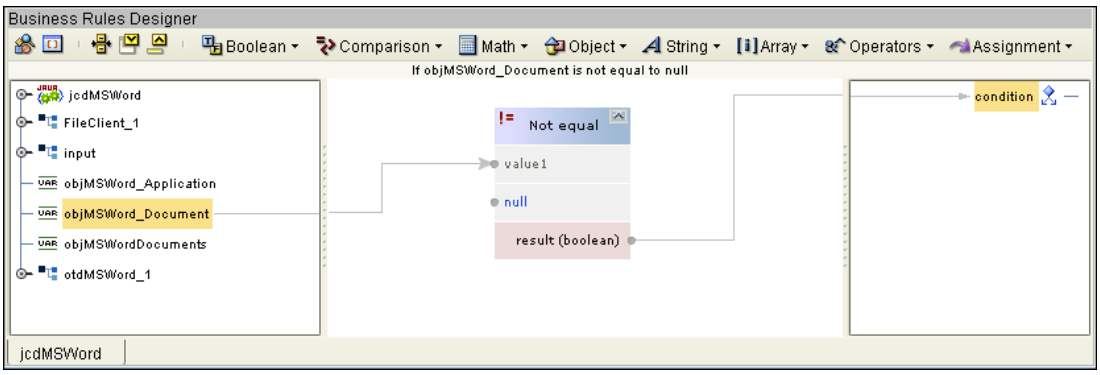
C From the Business Rules Designer Comparison menu, select **! = not equal**. The **Not equal** method box is added to the canvas.

D Map **objMSWord_Document** in the left pane of the Business Rules Designer to the **value1** input node of the **Not equal** method box.

E Right-click the **value2** input node of the **Not equal** method box and select **Add null Literal** from the shortcut menu. The value is changed to **null**.

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 (see Figure 29).

Figure 29 Collaboration Editor (Java) - Business Rules Designer



28 Create the **then** statement rule, **objMSWord_Document.Release,** under the **If objMSWord_Document is not equal to null** rule:

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.

29 The **If** Statement, the **condition: 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. Create the **If** Statement and the **condition: objMSWordDocuments is not equal to null** rule:

A Click the **if** icon on the Business Rules toolbar to add a new **if** statement to the Business Rules tree. Select the **condition**.

B From the Business Rules Designer Comparison menu, select **! = not equal**. The **Not equal** method box is added to the canvas.

C Map **objMSWordDocuments** in the left pane of the Business Rules Designer to the **value1** input node of the **Not equal** method box.

D Right-click the **value2** input node of the **Not equal** method box and select **Add null Literal** from the shortcut menu. The value is changed to **null**.

E 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** Statement, the **condition: 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. Create the **If** Statement and the **condition: objMSWord_Application is not equal to null** rule:

A Click the **if** icon on the Business Rules toolbar to add a new **if** statement to the Business Rules tree. Select the **condition**.

B From the Business Rules Designer Comparison menu, select **! = not equal**. The **Not equal** method box is added to the canvas.

C Map **objMSWord_Application** in the left pane of the Business Rules Designer to the **value1** input node of the **Not equal** method box.

D Right-click the **value2** input node of the **Not equal** method box and select **Add null Literal** from the shortcut menu. The value is changed to **null**.

E 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 43.

35 Save your current changes to the Repository.

Create the jcdMSExcel Collaboration (Java)

The **jcdMSExcel** 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 **prjCOM_Sample_JCD** 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, **jcdMSExcel**) and click Next.
3. For Step 2 of the wizard, **Select a Web Services Operation**, double-click **Sun 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 **Sun 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 **prjCOM_Sample_JCD** > **otdMSExcel**. The **otdMSExcel** 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 **jcdMSExcel** Collaboration is added to the Project Explorer tree.

Create the jcdMSExcel Collaboration Business Rules

The next step is to create the Business Rules of the **jcdMSExcel** Collaboration using the Collaboration Editor. The jcdMSExcel Collaboration is provided with the prjCom_Sample_JCD Project. If you are recreating this sample you can copy and paste the Collaboration into your project, or open the Collaboration and use it as an example. The **jcdMSExcel** Collaboration contains the Business Rules displayed in Figure 30.

Figure 30 jcdMSExcel Collaboration Business Rules

Note: *Business Rules are wrapped in Figure 30 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. If you are 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 using the **Business Rules Designer**.

Create the jcdMSDAO Collaboration (Java)

The **jcdMSDAO** 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 **prjCOM_Sample_JCD** 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, **jcdMSDAO**) and click Next.
3. For Step 2 of the wizard, **Select a Web Services Operation**, double-click **Sun 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 **Sun 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 **prjCOM_Sample_JCD** > **otdMSDAO360**. The **otdMSDAO360_1** 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 **jcdMSDAO** Collaboration is added to the Project Explorer tree.

Create the jcdMSDAO Collaboration Business Rules

The next step in the sample is to create the Business Rules of the **jcdMSDAO** Collaboration using the Collaboration Editor. The jcdMSDAO Collaboration is provided with the prjCom_Sample_JCD Project. If you are recreating this sample you can copy and paste the Collaboration into your project, or open the Collaboration and use it as an example.

The **jcdMSDAO** Collaboration contains the Business Rules displayed in **Figure 31 on page 58**.

Figure 31 jcdMSDAO Collaboration Business Rules

For more information on creating Business Rules using the Collaboration Editor See *the Sun SeeBeyond eGate™ Integrator User's Guide*.

5.6.5 Create a Connectivity Map

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

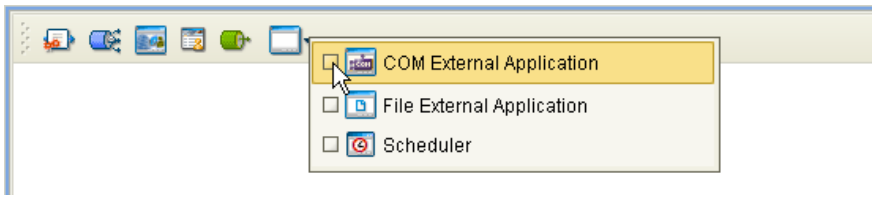
1. In Enterprise Explorer's Project Explorer, right-click the new Project and select **New > New Connectivity Map** from the shortcut menu.
2. The new Connectivity Map appears and a node for the Connectivity Map is added to the Project Explorer tree, labeled **CMap1**. Rename the Connectivity Map to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

5.6.6 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 32). The COM/DCOM External Application icon is then added to the Connectivity Map toolbar.

Figure 32 Connectivity Map - External Applications



To add the External Applications used with the cmMSWord sample, do the following:

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

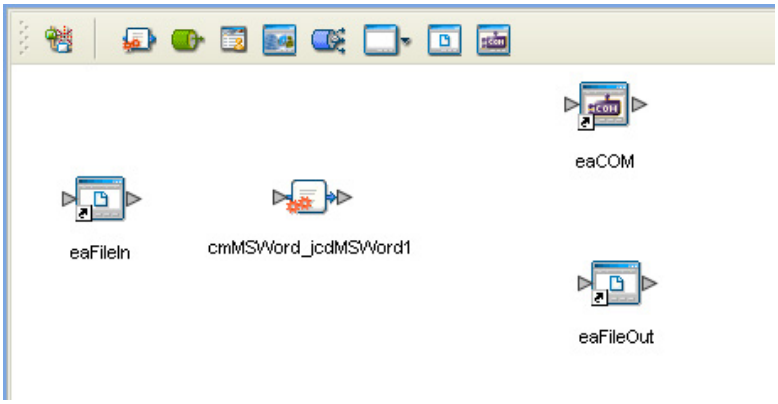
5.6.7 Populate the Connectivity Map

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 33 on page 60**:

- File External Application (2)
- Service (A Service is a container for Java Collaborations, Business Processes, eTL processes, and so forth)
- COM/DCOM External Application

Figure 33 cmMSWord Connectivity Map with Components



2 Rename the components as follows:

- File1 to **eaFileIn**
- cmMSWord_Service1 to **jcdMSWord1**
- COM1 to **eaCOM**
- File2 to **eaFileOut**

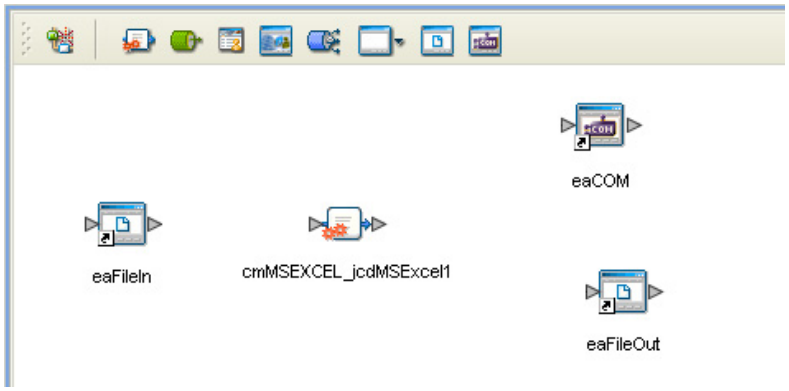
3 Using the same procedures you used to create the cmMSWord Connectivity Map, create the cmMSEXCEL and MSDAO Connectivity Maps.

cmMSEXCEL

The cmMSEXCEL Connectivity Map contains the following components, as displayed in Figure 34:

- File External Application (2): rename to **eaFileIn** and **eaFileOut**
- Service: rename to **cmMSEXCEL_jcdMSExcel1**
- COM/DCOM External Application: rename to **eaCOM**

Figure 34 cmMSEXCEL Connectivity Map

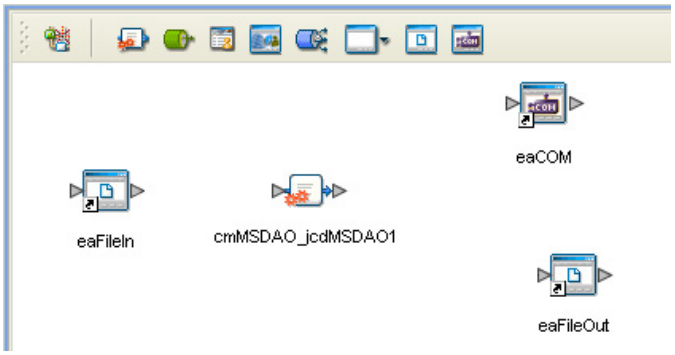


cmMSDAO

The cmMSDAO Connectivity Map contains the following components, as displayed in Figure 35:

- File External Application (2): rename to **eaFileIn** and **eaFileOut**
- Service: rename to **cmMSDAO_jcdMSDAO1**
- COM/DCOM External Application: rename to **eaCOM**

Figure 35 cmMSDAO Connectivity Map



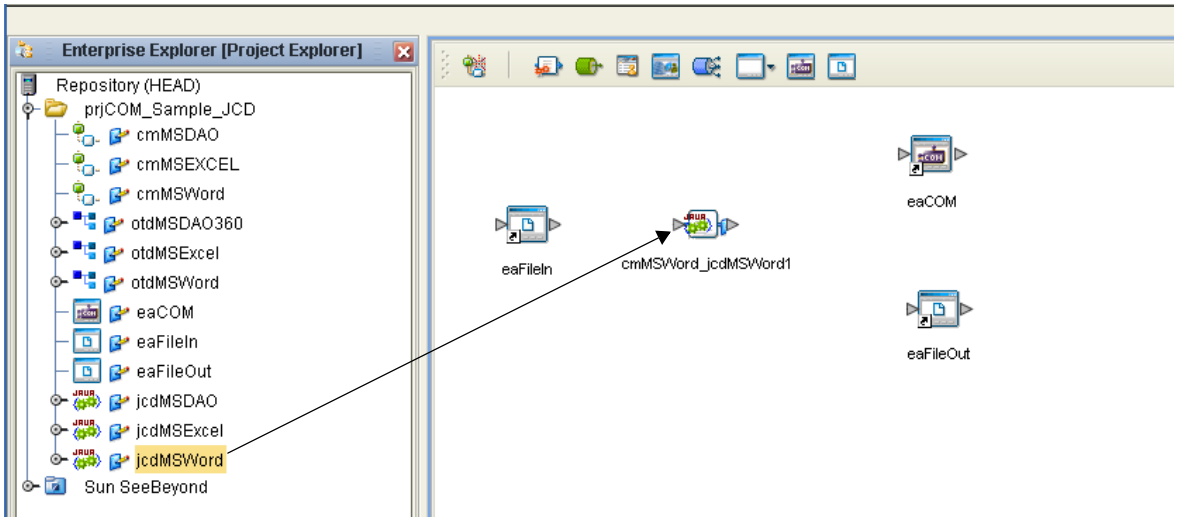
5.6.8 Creating Collaboration Bindings

Next, the components are associated and bindings are created in the Connectivity Map.

Create the cmMSWord Bindings

1. From the Project Explorer tree, double-click **cmMSWord** to display the Connectivity Map.
2. Drag and drop the **jcdMSWord** Collaboration from the Project Explorer tree to the **cmMSWord_jcdMSWord1 Service** in the Connectivity Map. If the Collaboration is successfully associated, the Service icon's "gears" change from red to green (see **Figure 36 on page 62**).

Figure 36 Connectivity Map - Binding the Collaborations



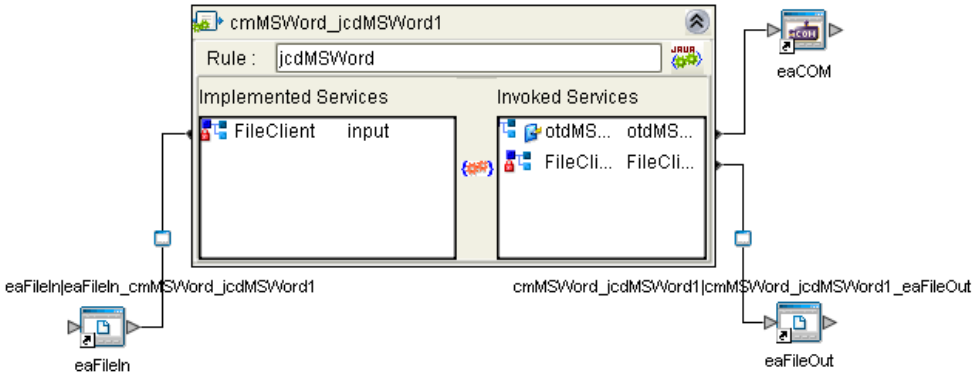
3 Double-click the **cmMSWord_jcdMSWord1** service in the Connectivity Map. The **cmMSWord_jcdMSWord1** binding dialog box appears.

4 From the **cmMSWord_jcdMSWord1** binding dialog box, map **FileClient input** (under Implemented Services) to the **eaFileIn** inbound External Application. To do this, click on **FileClient input** in the **cmMSWord_jcdMSWord1** binding dialog box and drag your cursor to the **eaFileIn** application.

5 From the **cmMSWord_jcdMSWord1** binding dialog box, map **otdMSWord_1, otdMSWord** (under Invoked Services) to the **eaCOM** External Application.

6 From the **cmMSWord_jcdMSWord1** binding dialog box, map **FileClient_1 FileClient** (under Invoked Services) to the **eaFileOut** outbound External Application (see Figure 37).

Figure 37 Connectivity Map - Connecting the Project's Components



7 Minimize the **cmMSWord_jcdMSWord1** binding dialog box.

Create the cmMSEXCEL Bindings

1 From the Project Explorer, double-click **cmMSExcel** to open the Connectivity Map.

2 Drag and drop the **jcdMSExcel** Collaboration from the Project Explorer tree to **cmMSEXCEL_jcdMSExcel1** in the Connectivity Map.

3 Double-click **cmMSEXCEL_jcdMSExcel1** in the Connectivity Map. The **cmMSEXCEL_jcdMSExcel1** binding dialog box appears.

4 From the **cmMSEXCEL_jcdMSExcel1** binding dialog box, map **FileClient input** (under Implemented Services) to the **esFileIn** inbound External Application. To do this, click on **FileClient input** in the **cmMSEXCEL_jcdMSExcel1** binding dialog box and dragging the cursor to the **esFileIn** application.

5 From the **cmMSEXCEL_jcdMSExcel1** binding dialog box, map **otdMSExcel_1, otdMSExcel** (under Invoked Services) to the **esCOM** External Application.

6 From the **cmMSEXCEL_jcdMSExcel1** binding dialog box, map **FileClient_1 FileClient** (under Invoked Services) to the **esFileOut** outbound External Application.

7 Minimize the **cmMSEXCEL_jcdMSExcel1** binding dialog box.

Create the cmMSDAO Bindings

1 From the Project Explorer, double-click **cmMSDAO** to open the Connectivity Map.

2 Drag and drop the **jcdMSDAO** Collaboration from the Project Explorer tree to **cmMSDAO_jcdMSDAO1** in the Connectivity Map.

3 Double-click **cmMSDAO_jcdMSDAO1** in the Connectivity Map. The **cmMSDAO_jcdMSDAO1** binding dialog box appears.

4 From the **cmMSDAO_jcdMSDAO1** binding dialog box, map **FileClient input** (under Implemented Services) to the **esFileIn** inbound External Application.

5 From the **cmMSDAO_jcdMSDAO1** binding dialog box, map **otdMSDAO360_1, otdMSDAO360** (under Invoked Services) to the **esCOM** External Application.

6 From the **cmMSDAO_jcdMSDAO1** binding dialog box, map **FileClient_1 FileClient** (under Invoked Services) to the **esFileOut** outbound External Application.

7 Minimize the **cmMSDAO_jcdMSDAO1** binding dialog box, and save your current changes to the Repository

5.6.9 Create the Environment

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

1 From the Enterprise Designer's Enterprise Explorer, click the **Environment Explorer** tab.

2 Right-click the Repository and select **New Environment**. A new Environment is added to the Environment Explorer tree.

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

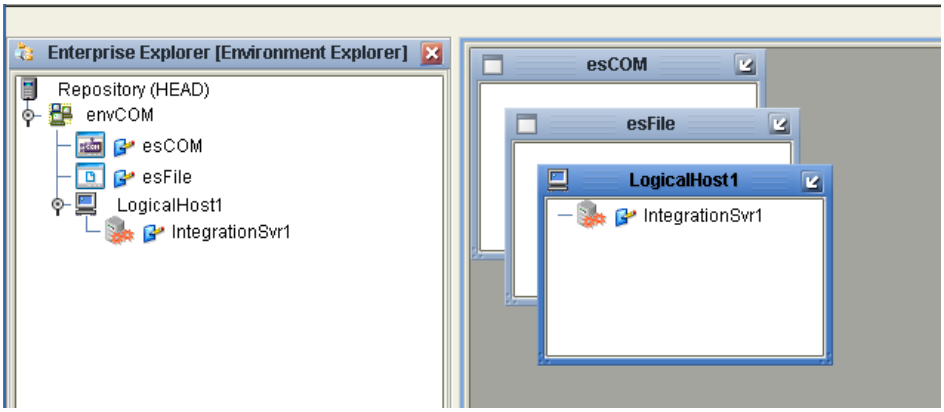
4 Right-click **envCOM** and select **New > COM External System**. Name the External System **esCOM**. **esCOM** is added to the Environment Editor.

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

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

7 From the Environment Explorere tree, right-click **LogicalHost1** and select **New > Sun SeeBeyond Integration Server** from the shortcut menu. A new Integration Server (**IntegrationSvr1**) is added to the Environment Explorer tree under LogicalHost1 (see Figure 38).

Figure 38 Environment Editor - envCOM



8 Save your current changes to the Repository.

5.6.10 Configuring the eWays

Component eWays are represented in the Connectivity Map by the nodes located between the External Applications and the Collaborations. The eWays facilitate communication and movement of data between the External Applications and the eGate system. The COM/DCOM eWay does not include any Connectivity Map Properties, and is not visible from the Connectivity Map.

Configure the File eWays Connectivity Map Properties for each of the samples

1 From the Project's Connectivity Maps, double-click the **eaFileIn** eWays. The **Properties Editor** opens to the **eaFileIn** eWay properties. Modify the properties for your system, including the settings in Table 5, and click **OK**.

Table 5 eaFileIn eWay Connectivity Map Properties

eaFileIn (inbound) eWay Connection Parameters	
Parameter Settings- Set as directed, otherwise use the default settings	
Input file name	For cmMSWord: inputWord*.fin. For cmMSEXCEL: inputExcel*.fin. For cmMSDAO: inputMSDAO*.fin.

2 In the same way, modify the eaFileOut eWay properties for your system, including the settings in Table 6, and click **OK**.

Table 6 eaFileOut eWay Connectivity Map Properties

eaFileOut eWay Connection Parameters	
Parameter Settings- Set as directed, otherwise use the default settings	
Output file name	For cmMSWord: MSWordoutput%d.dat. For cmMSExcel: MSExceloutput%d.dat. For cmMSDAO: MSDAOoutput%d.dat.

Configure the File eWay Environment Explorer Properties

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

4 Modify the File eWay Environment configuration properties for your system, including the settings in Table 7, and click **OK**.

Table 7 File eWay Environment Explorer Properties

File eWay Environment Explorer Properties	
Inbound File eWay - Set as directed, otherwise use the default settings	
Parameter Settings > Directory	*C:/temp or the input directory of your choice*
Outbound File eWay - Set as directed, otherwise use the default settings.	
Parameter Settings > Directory	*C:/temp or the output directory of your choice*

Configuring the COM/DCOM eWay

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

Modifying the COM/DCOM eWay Environment Explorer Properties

1 From the **Environment Explorer** tree, right-click the COM/DCOM External System (**esCOM** 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.

For more information on the COM/DCOM eWay configuration properties see **"Configuring the COM/DCOM eWay" on page 19**.

5.6.11 Configuring the Integration Server

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

1. From the Environment Explorer, right-click **IntegrationSvr1** under your **Logical Host**, and select **Properties** from the shortcut menu. The Integration Server Properties Editor appears.
2. Click the **Password** property field under **Sun SeeBeyond Integration Server Configuration**. An ellipsis appears in the property field.
3. Click the ellipsis. The **Password Settings** dialog box appears. Enter **STC** as the **Specific Value** and as the **Confirm Password**, and click **OK**.
4. Click **OK** to accept the new property and close the Properties Editor.

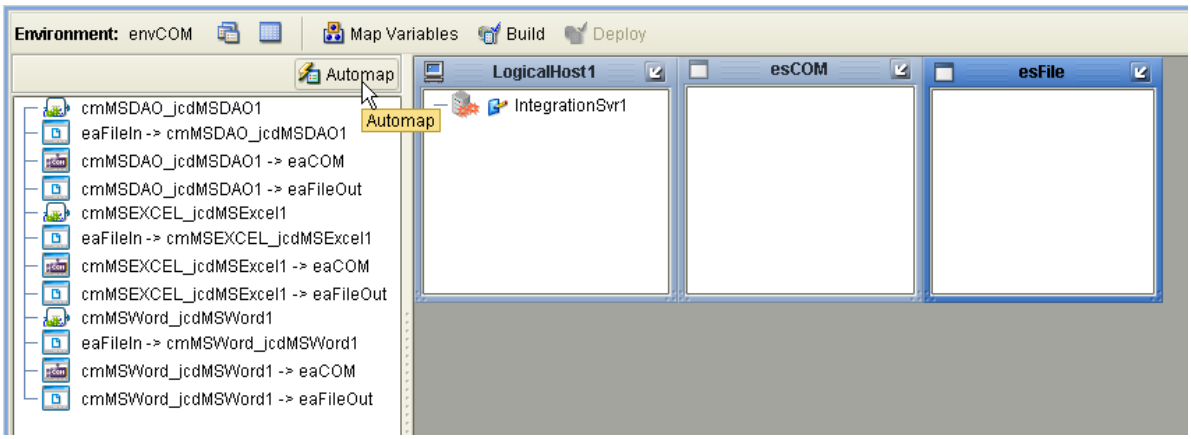
For more information on deploying a Project see the *Sun SeeBeyond Java™ Composite Application Platform Suite Deployment Guide*.

5.6.12 Creating the Deployment Profile

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

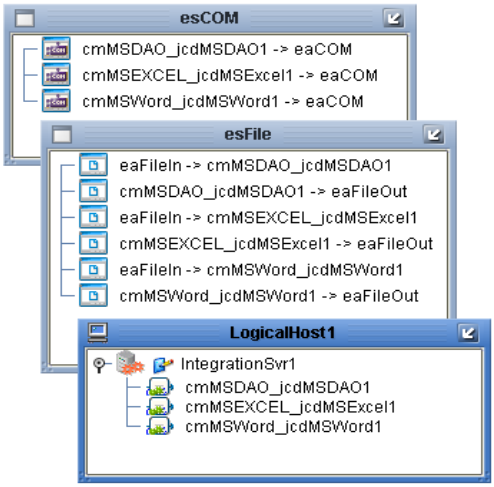
1. From the Enterprise Explorer's Project Explorer, right-click the Project (**prjCOM_Sample_JCD**) and select **New** > **Deployment Profile**.
2. Enter a name for the Deployment Profile (for this sample **dpCOM**). Make sure that the selected Environment is **envCOM**. Click **OK**.
3. Click the **Automap** icon as displayed in Figure 39.

Figure 39 Deployment Profile - Automaphj



The Project's components are automatically mapped to their system window as displayed in Figure 40.

Figure 40 Deployment Profile



4 Save your changes to the Repository.

5.6.13 Creating and Starting a Domain

Note: *The LogicalHost requires additional code (an additional permission) added to the server.policy file before you create a domain. See* **"Editing the LogicalHost server.policy File" on page 15** *for directions.*

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

Create and Start the Domain

1 Navigate to your **<*JavaCAPS51*>\logicalhost** directory (where <*JavaCAPS51*> is the location of your Sun Java Composite Application Platform Suite installation.

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

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

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

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

For more information about creating and managing domains see the *Sun SeeBeyond eGate™ Integrator User's Guide* and the *Sun SeeBeyond eGate™ Integrator System Administration Guide*.

5.6.14 Building and Deploying the Project

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

Build the Project

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

Deploy the Project

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

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

5.6.15 Running the Sample

*The runtime JNI bridge DLL file, **comewayruntime.dll**, must be added to the PATH for the Logical Host process before running a COM/DCOM Project. For directions, see* **"Adding the DLL file to the Path for the Logical Host Process" on page 15**.

To run your Project samples do the following:

MSWord Sample

1. Copy the sample Word document, **COMeWay_WordTest.doc**, and the sample input file, **inputWord.fin.~in**, to your specified input directory.
2. Rename the sample input file to inputWord.fin to trigger the eWay. When the file is picked up, the sample input file in the directory is renamed inputWord.fin.~in.
3. The new document appears. After a few seconds the document is saved and closed.
4. From your output directory, verify the output data.

MSExcel Sample

1. Copy the sample Excel Spreadsheet, **COMeWay_ExcelTest.xls**, and the sample input file, **inputExcel.fin.~in**, to your specified input directory.
2. Rename the sample input file to **inputExcel.fin** to trigger the eWay. When the file is picked up, the sample input file in the directory is renamed inputExcel.fin.~in.
3. The Excel spreadsheet appears, populates the two specified cells with the value **Hello World**, and closes.
4. From the output directory, you can verify the output file

MSDAO Sample

1. Copy an Access database and the sample input file, **inputMSDAO.fin.~in**, to your specified input directory.
2. Rename the sample input file to **inputMSDAO.fin** to trigger the eWay. When the file is picked up, the sample input file in the directory is renamed inputMSDAO.fin.~in.
3. From your output directory, verify the output data.

Chapter 6

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

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 Sun Java Composite Application Platform Suite Installer. 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

P

Q

R

S

T

V

Z