SeeBeyond ICAN Suite

# e\*Way Intelligent Adapter for COM/DCOM User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Java Version



SeeBeyond Proprietary and Confidential

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

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

© 2005 SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence. Version 20050406084157.

## Contents

Chapter 1	
Introduction	6
COM/DCOM Overview	6
Intended Reader	6
Supported Operating Systems	7
System Requirements External Application Requirements	7 7
Chapter 2	
Installation	8
Windows Installation Pre-installation Installation Procedure	8 8 8
Files/Directories Created by the Installation	9
Chapter 3	
Multi-Mode e*Way Configuration	10

Multi-Mode e*Way10JVM Settings10JNI DLL Absolute Pathname11CLASSPATH Prepend12CLASSPATH Override12CLASSPATH Append From Environment Variable12Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14Kollback Wait Interval14Standard IQ FIFO15	finder finde e findy configuration	
JNI DLL Absolute Pathname11CLASSPATH Prepend12CLASSPATH Override12CLASSPATH Append From Environment Variable12Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14Rellback Wait Interval14	Multi-Mode e*Way	10
CLASSPATH Prepend12CLASSPATH Override12CLASSPATH Append From Environment Variable12Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14Rental Settings14Rollback Wait Interval14	JVM Settings	10
CLASSPATH Override12CLASSPATH Append From Environment Variable12Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	JNI DLL Absolute Pathname	11
CLASSPATH Override12CLASSPATH Append From Environment Variable12Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	CLASSPATH Prepend	12
Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	CLASSPATH Override	12
Initial Heap Size13Maximum Heap Size13Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	CLASSPATH Append From Environment Variable	12
Maximum Stack Size for Native Threads13Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Initial Heap Size	13
Maximum Stack Size for JVM Threads13Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Maximum Heap Size	13
Disable JIT13Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Maximum Stack Size for Native Threads	13
Remote Debugging port number14Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Maximum Stack Size for JVM Threads	13
Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Disable JIT	13
Suspend option for debugging14Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Remote Debugging port number	14
Auxiliary JVM Configuration File14General Settings14Rollback Wait Interval14	Suspend option for debugging	14
General Settings14Rollback Wait Interval14	Auxiliary JVM Configuration File	14
Rollback Wait Interval 14	General Settings	14
	0	14

### Chapter 4

e*Way Connection Configuration	16
Configuring e*Way Connections Connector Type Class COM/DCOM Configuration ProgID Hostname	
Chapter 5	
Implementation	19
Implementation Notes	19
Considerations	20
COM/DCOM Sample Implementation Overview Importing the Sample Schema	20 21
Creating the Sample Schema Manually	21
Creating Event Type Definitions Creating an Event Type Definition Using an Existing .xsc Creating an Event Type Definition Using the COM ETD Wizard	22 22 23
<b>Creating and Configuring the Component e*Ways</b> To Create the Trigger e*Way To Create the COM/DCOM e*Way (Multi-Mode)	<b>25</b> 25 26
Creating the e*Way Connection To Create and Configure a New e*Way Connection	27 28
<b>Creating Intelligent Queues (IQs)</b> To Create and Modify an Intelligent Queue for the COM/DCOM e*Way	<b>29</b> 29

To create and modify an intelligent Quede for the COM/Decome Way	25
Creating the Collaboration Rules	29
To Create the Pass Through Collaboration Rules	30
To Create the Java Collaboration Rules	31
Using the Collaboration Rules Editor	32
Creating Collaborations	39
To Create the trigger e*Way Collaboration	39
To Create the COM_Multi_Mode Collaboration	40
Configuring the DCOM Server	42
Sample Schema	46
Completing the Configuration of the ExcelSample Schema	46
Executing the Schema	46
COM/DCOM Run-Time Exceptions	47
Java Classes and the Lower JNI Layer	48

4

Creating Business Rules Using the e*Way's Low-level JNI Layer	
Chapter 6	
e*Way Classes and Methods	52
COM/DCOM e*Way Javadoc	52
Index	53

### Chapter 1

## Introduction

The e\*Way Intelligent Adapter for COM/DCOM enables the e\*Gate system to exchange data with server side COM/DCOM-enabled applications and components. This document describes how the Java<sup>TM</sup>-enabled COM/DCOM e\*Way is installed, configured, and implemented.

### 1.1 COM/DCOM Overview

The Microsoft<sup>™</sup> Component Object Model (COM) is a component software architecture that allows applications and systems to be built using separate components. 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. The COM objects create *handles* that provide access to the COM-enabled applications.

The COM/DCOM e\*Way enables configurable transparent data exchange with COM/DCOM enabled applications and components.

### **1.2** Intended Reader

The reader of this guide is presumed:

• to be a developer or system administrator with responsibility for maintaining the e\*Gate system.

- to have a working knowledge of Windows operations and administration.
- to be thoroughly familiar with COM/DCOM, Java, and Windows-style GUI operations.

### **1.3 Supported Operating Systems**

The Java-enabled COM/DCOM e\*Way is available on the following operating systems:

• Windows 2000, Windows XP, and Windows Server 2003

### **1.4 System Requirements**

To use the COM/DCOM e\*Way, you need the following:

- An e\*Gate Participating Host.
- A TCP/IP network connection
- Additional disk space for e\*Way executable, configuration, library, and script files. The disk space is required on both the Participating and the Registry Host. Additional disk space is required to process and queue the data that this e\*Way processes; the amount necessary varies based on the type and size of the data being processed, and any external applications performing the processing.

The client components of the external systems with which the e\*Way interfaces, have their own requirements; see those systems' documentation for more details.

*Note:* Open and review the *Readme.txt* for the COM/DCOM e\*Way regarding any additional requirements prior to installation. The Readme.txt is located on the Installation CD\_ROM at setup\addons\ewims.

### 1.4.1. External Application Requirements

The COM/DCOM e\*Way requires an existing COM/DCOM-compliant application or component.

### **Chapter 2**

## Installation

This chapter contains pre-installation requirements and the procedures for installing the Java-enabled COM/DCOM e\*Way. In addition, a list of installed files is provided, along with the directories where these files are located.

### 2.1 Windows Installation

### 2.1.1. Pre-installation

- Quit all Windows programs before running the setup program, including any anti-virus applications.
- You must have Administrator privileges to install this e\*Way.

### 2.1.2. Installation Procedure

#### To install the COM/DCOM e\*Way on Windows

- 1 Log in as an Administrator to the workstation on which you are installing the e\*Way.
- 2 Insert the e\*Way installation CD-ROM into the CD-ROM drive.
- 3 If the CD-ROM drive's Autorun feature is enabled, the setup application launches automatically; skip ahead to step 4. Otherwise, use Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.
- 4 The InstallShield setup application launches. Follow the installation instructions until you come to the **Please choose the product to install** dialog box.
- 5 Select **e\*Gate Integrator**, then click **Next**.
- 6 Follow the on-screen instructions until you come to the second **Please choose the product to install** dialog box.
- 7 Clear the check boxes for all selections except **Add-ons**, and then click **Next**.
- 8 Follow the on-screen instructions until you come to the **Select Components** dialog box.

- 9 Highlight (but do not check) e\*Ways, and then click the Change button. The SelectSub-components dialog box appears.
- 10 Select the **COM/DCOM e\*Way**. Click the continue button to return to the **Select Components** dialog box, then click **Next**.
- 11 Follow the rest of the on-screen instructions to install the Java-enabled COM/DCOM e\*Way. Be sure to install the e\*Way files in the suggested client installation directory. The installation utility detects and suggests the appropriate installation directory. Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested installation directory setting.

Once you have installed and configured this e\*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e\*Way can perform its intended functions. For more information about any of these procedures, please see the online Help.

### 2.2 Files/Directories Created by the Installation

The Java-enabled COM/DCOM e\*Way installation process installs the following files within the e\*Gate "client" directory tree. Files are installed within the "egate/client" tree on the Participating Host and committed to the "default" schema on the Registry Host.

e*Gate Directory	File(s)
bin\	stccombuilder.exe stccomjavabuilder.exe stccomutil.dll
bin\java	gnu-regexp-1.1.1.jar jcscomp.jar stcjintegra.jar xerces.jar
classes\	ewscom.jar ewmscom.jar stcjcs.jar
configs\MSCOM\	com.def
etd\	com.ctl comwizard.ctl stcewmscom

Table 1	Files Created by the Installation
---------	-----------------------------------

*Note:* For more information about configuring e\*Ways or how to use the e\*Way Editor, see the e\*Gate Integrator User's Guide.

### Chapter 3

## **Multi-Mode e\*Way Configuration**

This chapter describes how to configure the Multi-Mode e\*Way.

### 3.1 Multi-Mode e\*Way

Multi-Mode e\*Way properties are set using the Schema Designer.

#### To create and configure a New Multi-Mode e\*Way

- 1 Select the Navigator's Components tab.
- 2 Open the host on which you want to create the e\*Way.
- 3 On the Palette, click on the **Create a New e\*Way** button.
- 4 The New e\*Way Component window opens. Enter the name of the new e\*Way, then click **OK**.
- 5 Right-click the new e\*Way and select **Properties** to open the Properties dialog box.
- 6 The **Executable File** field defaults to **stceway.exe**. (stceway.exe is located in the "bin\" directory).
- 7 Under the **Configuration File** field, click on the **New** button. When the Settings page opens, set the configuration parameters for this configuration file.
- 8 After selecting the desired parameters, save the configuration file. Close the **.cfg** file and select **OK** to close the e\*Way Properties Window.

#### Multi-Mode e\*Way Configuration Parameters

The Multi-Mode e\*Way configuration parameters are arranged in the following sections:

- JVM Settings on page 10
- General Settings on page 14

### 3.1.1. JVM Settings

The JVM Settings control basic Java Virtual Machine settings.

- JNI DLL Absolute Pathname on page 11
- CLASSPATH Prepend on page 12

- CLASSPATH Override on page 12
- CLASSPATH Append From Environment Variable on page 12
- Initial Heap Size on page 13
- Maximum Heap Size on page 13
- Maximum Stack Size for Native Threads on page 13
- Maximum Stack Size for JVM Threads on page 13
- Disable JIT on page 13
- Remote Debugging port number on page 14
- Suspend option for debugging on page 14
- Auxiliary JVM Configuration File on page 14

### **JNI DLL Absolute Pathname**

#### Description

Specifies the absolute pathname to where the JNI DLL installed by the *Java 2 SDK* 1.3.1\_02 is located on the Participating Host.

#### **Required Values**

A valid pathname.

#### **Additional Information**

The JNI dll name varies on different O/S platforms:

OS	Java 2 JNI DLL Name
Windows	jvm.dll

The value assigned may contain a reference to an environment variable. To do this, enclose the variable name within a pair of % symbols. For example:

%MY JNIDLL%

Such variables are used when multiple Participating Hosts are used on different platforms.

*Note:* To ensure that the JNI DLL loads successfully, the Dynamic Load Library search path environment variable must be set appropriately to include all the directories under the Java 2 SDK (or JDK) installation directory that contain shared libraries (UNIX) or DLLs.

### **CLASSPATH Prepend**

#### Description

Specifies the paths to be prepended to the CLASSPATH environment variable for the JVM.

#### **Required Values**

An absolute path or an environmental variable. This parameter is optional.

#### **Additional Information**

If left unset, no paths are prepended to the CLASSPATH environment variable.

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of % signs. For example:

%MY\_PRECLASSPATH%

### CLASSPATH Override

#### Description

Specifies the complete CLASSPATH variable to be used by the JVM. This parameter is optional. If left unset, an appropriate CLASSPATH environment variable (consisting of required e\*Gate components concatenated with the system version of CLASSPATH) is set.

#### **Required Values**

An absolute path or an environmental variable. This parameter is optional.

#### **Additional Information**

Existing environment variables may be referenced in this parameter by enclosing the variable name in a pair of % signs. For example:

%MY\_CLASSPATH%

### **CLASSPATH Append From Environment Variable**

#### Description

Specifies whether the path is appended for the CLASSPATH environmental variable to jar and zip files needed by the JVM.

#### **Required Values**

YES or NO. The configured default is YES.

*Note:* All necessary JAR and ZIP files needed by both e\*Gate and the JVM must be included. It is advised that the CLASSPATH Prepend parameter be used.

### Initial Heap Size

#### Description

Specifies the value for the initial heap size in bytes. If set to 0 (zero), the preferred value for the initial heap size of the JVM is used.

#### **Required Values**

An integer between 0 and 2147483647. This parameter is optional.

### Maximum Heap Size

#### Description

Specifies the value of the maximum heap size in bytes. If set to 0 (zero), the preferred value for the maximum heap size of the JVM is used.

#### **Required Values**

An integer between 0 and 2147483647. This parameter is optional.

### Maximum Stack Size for Native Threads

#### Description

Specifies the value of the maximum stack size in bytes for native threads. If set to 0 (zero), the default value is used.

#### **Required Values**

An integer between 0 and 2147483647. This parameter is optional.

### Maximum Stack Size for JVM Threads

#### Description

Specifies the value of the maximum stack size in bytes for JVM threads. If set to 0 (zero), the preferred value for the maximum heap size of the JVM is used.

#### **Required Values**

An integer between 0 and 2147483647. This parameter is optional.

### **Disable JIT**

#### Description

Specifies whether the Just-In-Time (JIT) compiler is disabled.

#### **Required Values**

YES or NO.

*Note:* This parameter is not supported for Java Release 1.

### **Remote Debugging port number**

#### Description

Specifies the port number by which the e\*Gate Java Debugger can connect with the JVM to allow remote debugging.

#### **Required Values**

An unused port number in the range 2000 through 65535. If not specified, the e\*Gate Java Debugger is not able to connect to this e\*Way.

### Suspend option for debugging

#### Description

Allows you to specify that the e\*Way should do no processing until an e\*Gate Java Debugger has successfully connected to it.

#### **Required Values**

**YES** or **No**. YES suspends e\*Way processing until a Debugger connects to it. NO enables e\*Way processing immediately upon startup.

### Auxiliary JVM Configuration File

#### Description

Specifies an auxiliary JVM configuration file for additional parameters.

#### **Required Values**

The location of the auxiliary JVM configuration file.

### 3.1.2. General Settings

For more information on the General Settings configuration parameters see the *e*\**Gate Integrator User's Guide*. The General Settings section contains the following parameters:

- Rollback Wait Interval on page 14
- Standard IQ FIFO on page 15

### **Rollback Wait Interval**

#### Description

Specifies the time interval to wait before rolling back the transaction.

#### **Required Values**

A number within the range of 0 to 99999999, representing the time interval in milliseconds.

### Standard IQ FIFO

#### Description

Specifies whether the highest priority messages from all STC\_Standard IQs will be delivered in the first-in-first-out (FIFO) order.

#### **Required Values**

Select **YES** or **NO**. YES indicates that the e\*Way will retrieve messages from all STC\_Standard IQs in the first-in-first-out (FIFO) order. NO indicates that this feature is disabled. NO is the configured default.

### Chapter 4

## e\*Way Connection Configuration

This chapter describes how to configure the COM/DCOM e\*Way Connection.

### 4.1 Configuring e\*Way Connections

e\*Way Connections are set using the Schema Designer.

#### To create and configure e\*Way Connections

- 1 In the Schema Designer's Component editor, select the **e\*Way Connections** folder.
- 2 On the palette, click on the icon to create a new **e\*Way Connection**.
- 3 The New e\*Way Connection Component dialog box opens, enter a name for the e\*Way Connection. Click OK.
- 4 Double-click on the new **e\*Way Connection**. For this example, the connection has been defined as **com\_connector**.
- 5 The e\*Way Connection Properties dialog box opens.
- 6 From the **e\*Way Connection Type** drop-down box, select **COM/DCOM**.
- 7 Enter the **Event Type "get"** interval in the dialog box provided. The configured default is 100 milliseconds. The "get interval is the intervening period at which the e\*Way connection is polled.
- 8 From the **e\*Way Connection Configuration File**, click **New** to create a new Configuration File for this e\*Way Connection. (To use an existing file, click **Find**.)

The COM/DCOM e\*Way Connection configuration parameters are organized into the following sections:

- Connector on page 16
- COM/DCOM Configuration on page 17

### 4.1.1. Connector

This section contains a set of top level parameters.

- Type on page 17
- Class on page 17

### Туре

### Description

String-set. Specifies the connector type. Retain the default (COM) value for COM connections.

### **Required Values**

**COM**. The value defaults to **COM**.

### Class

### Description

String-set. Specifies the class name of the COM/DCOM connector object.

### **Required Values**

Retain the default value. The default is **com.stc.eways.com.ComEWayConnector**.

### 4.1.2. COM/DCOM Configuration

This section contains a set of top level parameters.

- ProgID on page 17
- Hostname on page 18

### ProgID

### Description

String-set. Specifies the programmatic ID that identifies the component to be used (for example: Excel.Application). Changes to this parameter in the ETD Editor Properties take presidence over the e\*Way Connection parameter.

The ProgID is a Windows registry entry that uniquely identifies a program or a COM object. Unlike a Globally Unique Identifier (GUID), the ProgID is a humanly-readable alphanumeric string. The ProgID consists of three parts in this format:

```
vendor.component.version
```

The *vendor* parameter is the control's library, the *component*, it's class, and the *version* number, which is optional. If no version is given, the latest version of the application object is assumed. Each parameter is separated by periods and no spaces; for example the Visual Basic Command Button object's ProgID is:

VB.CommandButton

The ProgID for most objects can be determined by checking, for example, the Visual Basic Object Browser. The Windows registry entry reads as follows:

 $\texttt{HKEY\_LOCAL\_MACHINE\SOFTWARE\Classes} ProgID = \\$ 

#### **Required Values**

A valid programmatic ID to be used by the component.

### Hostname

#### Description

String-set. Specifies the name the remote server on which the DCOM object resides. Changes to this parameter in the ETD Editor Properties take presidence over the e\*Way Connection parameter.

#### **Required Values**

A valid host name.

*Note:* Hostname is only relevant to DCOM components (that is .EXEs). Do not use the Hostname value for .DLLs.

### Chapter 5

## Implementation

This chapter includes information pertinent to implementing the Java-enabled COM/ DCOM e\*Way in a production environment. Information on implementing the sample schema is included.

The following assumptions are applicable to this implementation:

- The COM/DCOM e\*Way has been successfully installed.
- The executable and the configuration files have been appropriately assigned.
- All necessary .jar files are accessible.
- The user has a working understanding of COM/DCOM concepts.

### 5.1 Implementation Notes

The COM/DCOM e\*Way supports the following data types:

OLE Data Types	Description
VT_l2	2-byte signed int
VT_14	4-byte signed int
VT_R4	4-byte real
VT_R8	8-byte real
VT_BSTR	Binary string
VT_DISPATCH	IDispatch (1 dimentional array only)
VT_ERROR	4-byte error code
VT_BOOL	Boolean
VT_UI1	Unsigned char
VT_DATE	The standard COM DATE data type (8- byte real).

19

#### Table 2Supported Data Types

OLE Data Types	Description
VT_SAFEARRAY VT_I2 VT_I4 VT_R4 VT_R8 VT_BSTR VT_DISPATCH VT_ERROR VT_BOOL VT_UI1 VT_DATE VT_VARIANT	1 dimensional array (VT_VARIANT can support a 2 dimensional array)

Table 2	Supported Data	Types (Continued)
---------	----------------	-------------------

All of the above types are supported for In and Out parameters.

The COM/DCOM e\*Way does not supports the following data types:

 Table 3
 Non-Supported Data Types

OLE Data Types	Description
VT_CY	currency

*Important: The code processes one dimensional SAFEARRAYs only. The number of elements in all rows are expected to be equal.* VT\_SAFEARRAY *has specific, limited functionality as used in the COM/DCOM e\*Way.* 

### 5.2 **Considerations**

- Currently the Java e\*Way Intelligent Adapter for COM/DCOM does not support accessing a range of cells from Excel. Each cells must be accessed one by one.
- The method names used for created custom Com objects must not match a method name on the Java Object class. Duplicate names will result in compile-time errors indicating that a name collision has occurred. It is also recommended that you avoid methods names that are similar to those of any other Java object that may be used in your Collaboration.

### 5.3 COM/DCOM Sample Implementation Overview

During installation, the host and Control Broker are automatically created and configured. The default name of each is the name of the host on which you are installing the e\*Gate Schema Designer GUI. To complete the implementation of the Java-enabled COM/DCOM e\*Way, do the following:

• Make sure that the **Control Broker** is activated.

- In the e\*Gate Schema Designer, define and configure the following as necessary:
  - **Trigger e\*Way** using **stcewfile.exe**.
  - The Multi-Mode e\*Way component as described in Chapter 3.
  - Event Type Definitions used to package the data to be exchanged with the external system.
  - Collaboration Rules to process Events.
  - The e\*Way Connection as described in Chapter 4.
  - **Collaborations**, to be associated with each e\*Way component, to apply the required Collaboration Rules.
  - The destination to which data is published.

The following sections describe how to define and associate each of the above components. The sample schema provided on the Installation CD-ROM is, for the most part, complete, once it is imported. The e\*Way component implementation is provided for the purpose of explaining how each of these components are created manually.

### 5.3.1. Importing the Sample Schema

The first task in deploying the sample implementation is to create a new schema name. While it is possible to use the default schema for the sample implementation, it is recommended that you create a separate schema for testing purposes. After you install the COM/DCOM e\*Way, do the following:

- 1 Start the e\*Gate Schema Designer GUI.
- 2 When prompted to log in, select the host specified during installation, and enter the password.
- 3 When prompted to select a schema, click New.
- 4 Enter a name for the new schema. In this case, enter **ExcelSample**, or any appropriate name as desired.
- 5 Select Create from export, locate the ExcelSample.zip on the CD, and click Open.

The e\*Gate Schema Designer opens to the new schema. It is also necessary to create the file **test.xls** in C:\temp. To create this file do the following:

- 1 Open Microsoft Excel.
- 2 Create a new worksheet called **test.xls**.
- 3 Save this file to the **c:\temp** directory and close **Excel**.

### 5.4 Creating the Sample Schema Manually

The following sections explain how to manually configure the e\*Way components to create the same sample schema that is created when the sample is imported.

### 5.5 **Creating Event Type Definitions**

An ETD is a structural representation of an Event (that is, the blueprint of an Event). The COM/DCOM e\*Way installation includes the COM Event Type Definition Wizard to assist in the quick creation for COM specific ETDs. In addition, if you have imported the sample schema, you have existing .xsc files that can be opened in the ETD Editor.

In COM, the Event Type corresponds to the object being used (it's methods). These methods are exposed in the Collaboration to allow the user to get and set the properties or call one or more methods. In addition, methods are made available to get and set the objects properties.

### 5.5.1. Creating an Event Type Definition Using an Existing .xsc

For the purpose of this example, the following procedure shows how to edit an **Event Type** from an existing .xsc file using **com.xsc** as the input file.

- 1 Select the **Event Types** folder on the **Components** tab of the e\*Gate Navigator.
- 2 On the palette, click the **Create a New Event Type** button.
- 3 Enter the name of the **Event Type** in the **New Event Type Component** window, then click **OK** (for this sample the Event Type is defined as **Excel**).
- 4 Double-click the new **Event Type** to edit its properties. The Event Type Properties window opens (see Figure 1).

Event Type - Excel Properties	×
General	
I	
Excel	
Event Type Definition	
	1
etd\Excel.xsc	1
Clear Find Edit	
	1
	1
	1
	1
OK Cancel Apply Help	

Figure 1 Event Type - Properties

5 Click **Find** under the Event Type Definition field. Find and select **Excel.xsc**.

6 Click the **Edit** button to open the xsc file in the ETD Editor (see Figure 2). This is a large file and may take more than a few minutes to open.

<b>= ETD Editor:</b> ile Edit Help				
) 🚅 🖬 🔛 🤋				
Event Type	Event T	ype Definition	Properties	EXCEL 9
EXCEL9		EXCEL9		
-		S connect	(Name)	EXCEL9
		S disconnect	anchored	false
		Adjustments	array	false
Internal Templates		CalloutFormat	beginAnchor	red false
Adjustments	- 1	ColorFormat	beginDelim	
CalloutFormat		LineFormat	childMax	undefined
ColorFormat		ShapeNode	childMin	undefined
LineFormat		ShapeNodes	comment	
		PictureFormat	defaultBytes	
ShapeNode		ShadowFormat	defaultEnco	
🗄 ShapeNodes		TextEffectFormat	encoding	
🖶 PictureFormat		ThreeDFormat	endAnchore	d false
🗄 ShadowFormat			endDelim	
TextEffectFormat		FillFormat	endOfRec	false
ThreeDFormat		Application	fixedValue	
🖥 FillFormat		💼 _Chart	format	
		💼 Sheets	inputMatch	
External Templates		💼 _Worksheet	iavaName	EXCEL9
		💼 _Global	javaType	java.lang.String
		💼 _Workbook	length	undefined
		📲 Workbooks	lengthFrom	undefined
		🖶 PublishObject	lengthSize	undefined
		📲 DefaultWebOptions	maxOccurs	1
		📲 WebOptions	member	
	[	📲 TreeviewControl	minOccurs	1
		💼 CubeField	order	sequence
		🗄 CubeFields	precedence	child
		🗄 Font	public	false
	÷	🗄 Window	readOnly	false
		Windows	reference	
		AppEvents	required	false
oading Local Template:			IscavOutnut	Ifalse  01 3:22 PM

Figure 2 Event Type Definition Editor - Excel

- 7 On the File menu, click **Compile and Save**. This should be done after any modifications are made to the file.
- 8 On the **File** menu, click **Promote to Run Time** to promote the file from the **sandbox** to run time. A dialog box opens listing the files that have been successfully promoted.
- 9 Close the ETD Editor. Click **OK** to close the Event Type Properties Dialog box.

### 5.5.2. Creating an Event Type Definition Using the COM ETD Wizard

The COM ETD Wizard is used to create a ETD structure specific to the COM, with corresponding nodes and methods. For the purpose of this example, the following procedure shows how to create an **Event Type Definition** (**ETD**) using the **COM ETD Wizard**. This ETD is not used in the sample schema included with the CD.

1 Select the **Event Types** folder on the **Components** tab of the e\*Gate Navigator.

- 2 On the palette, click the **Create a New Event Type** button.
- 3 Enter the name of the **Event Type** in the **New Event Type Component** window, then click **OK** (for this sample the Event Type is defined as **Excel**).
- 4 Double-click the new **Event Type** to edit its properties.
- 5 When the **Properties** window opens, click the **New** button. The ETD Editor opens.
- 6 Select File, New to open the New Event Type Definition window (see Figure 3).
- 7 Select the **COM/DCOMWizard**.

New			
COM/DCOMWizard	Custom ETD	DTDWizard	SEFWizard
	Wizard		SET WIZdid
SSCWizard	XSDWizard		

#### **Figure 3** New Event Type Definition - COM/DCOMWizard

- 8 Enter a Package Name where the ETD builder can place all the generated Java classes associated with the created ETD (for this sample, **com.stc.ExcelSample**).
- 9 Enter the COM Type library file from which the .xsc file is created. TypeLib files are \*.tlb, \*.olb, \*.dll, \*.ocx, and \*.exe files. For the purpose of this sample the selected file is Excel9.olb at C:\Program Files\Microsoft Office\Office. Click OK and Finish.
- *Note:* COM type library files describe the methods exposed from an automation compatible component. Com type library typically have the file extension .tlb or .olb. However, most components embed the type library file in the DLL, OCX or EXE that houses the component.
  - 10 The **EDT Editor** generates an Event Type structured for the specific COM/DCOM application file (seeFigure 4). The ETD is read-only, allowing for no further configuration from the ETD Editor.

🔚 ETD Editor: EventTypeDe	finition1.xsc (Sandbox)			
File Edit Help				
🗅 🚅 🖬 🔡 💡				
Event Type	Event Type Definition	_	Properties -MS	RDR9
MSBDR9 Internal Templates Section PageSetup COMETDVariant External Templates	WSBDR9     Sconnect     MSBDR9     Section     Sections     Sections     P-     PageSetup		Properties -MS (Name) anchored array beginAnchored beginDelim childMax childMin comment defaultValue defaultEncoding encoding endAnchored endDelim endOfRec fixedValue format inputMatch javaType length maxOccurs member minOccurs order precedence public readOnly reference	MSBDR9 false false false undefined undefined
		10/	10/2001 7	7:54 AM

Figure 4 Event Type Definition Editor

- 11 From the File menu click **Compile and Save**.
- 12 Once the file has compiled, from the **File** menu, click **Promote to Run Time**. The file is promoted to the run time environment.
- **13** Close the ETD Editor.

## 5.6 Creating and Configuring the Component e\*Ways

e\*Ways connect with external systems to poll or send data. They also transform and route data. Multi-Mode e\*Ways are used to run Java Collaborations that utilize e\*Way Connections to send and receive Events to and from multiple external systems.

### To Create the Trigger e\*Way

- 1 Select the Navigator's **Components** tab.
- 2 Open the host on which you want to create the e\*Ways.
- 3 Select the **Control Broker** that manages the new e\*Ways.
- 4 On the palette, click the **Create a New e\*Way** button.
- 5 Enter the name of the new e\*Way (in this sample, **trigger**), then click **OK**.
- 6 Right-click the new e\*Way and select **Properties** to edit its properties.

- 7 The e\*Way Properties window opens. Click the **Find** button beneath the **Executable File** field, and select **stcewfile.exe** as the executable file.
- 8 Under the **Configuration File** field, click the **New** button. The Edit Settings window opens. Select the settings as displayed in Table 4 for this configuration file.

Parameter	Value			
General Settings (unless otherwise stated, leave settings as default)				
AllowIncoming	YES			
AllowOutgoing	NO			
Outbound Settings	Default			
Poller Inbound Settings				
PollDirectory	C:\Indata (input file folder)			
InputFileExtension	*.fin (input file extension)			
PollMilliseconds	1000			
Remove EOL	YES			
MultipleRecordsPerFile	YES			
MaxBytesPerLine	4096			
BytesPerLineIsFixed	NO			
File Records Per eGate Event	1			
Performance Testing	Default			

Table 4	Configuration Parameters for the trigger e*Way	
Table 4	Configuration Parameters for the trigger e	e*Way

- **9** After selecting the desired parameters, save the **configuration** file (as "**trigger.cfg**").
- 10 From the File menu, click Promote to Run Time. This closes the .cfg file.
- 11 In the e\*Way Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each e\*Way you configure.
  - A Use the **Startup** tab to specify whether the e\*Way starts automatically, or restarts after abnormal termination or due to scheduling, and so forth.
  - **B** Use the **Advanced** tab to specify or view the activity and error logging levels, as well as the Event threshold information.
  - C Use **Security** to view or set privilege assignments.
- 12 Select **OK** to close the e\*Way Properties window.

### To Create the COM/DCOM e\*Way (Multi-Mode)

- 1 Select the Navigator's **Components** tab.
- 2 Open the host on which you want to create the e\*Way.
- 3 Select the **Control Broker** that manages the new e\*Way.
- 4 On the palette, click the **Create a New e\*Way** button.
- 5 Enter the name of the new e\*Way (in this sample, excel\_gate), then click OK.

- 6 Right-click the new e\*Way and select **Properties** to edit its properties.
- 7 When the e\*Way Properties window opens, click the **Find** button beneath the **Executable File** field, and select **stceway.exe** as the executable file (see Figure 5).

🜐 e*Way - excel_gate Properties		×
General Start Up Advanced Security		
excel_gate		
Executable file		
bin\stceway.exe		
	Clear	Find
Additional command line arguments: <u> % -un %_USERNAME% -up %_PASSWOF</u> Run as user	RD% -rp %_	REGPORT%
Administrator		•
Configuration file		
Clear	Find	New
OK Cancel g	Apply	Help

Figure 5 Multi-Mode e\*Way Properties

- 8 To edit the JVM Settings, select **New** under Configuration file. For information on setting the parameters for the Multi-Mode e\*Way see **Multi-Mode e\*Way Configuration** on page 10.
- 9 Save the **.cfg** file, and promote to run time to close the Edit Settings window.
- 10 In the e\*Way Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each.
  - A Use the **Startup** tab to specify whether the e\*Way starts automatically, restarts after abnormal termination or due to scheduling, etc.
  - **B** Use the **Advanced** tab to specify or view the activity and error logging levels, as well as the Event threshold information.
  - **C** Use **Security** to view or set privilege assignments.
- 11 Click **OK** to close e\*Way Properties window.

### 5.7 Creating the e\*Way Connection

The e\*Way Connection configuration file contains connection information along with the information needed to communicate using COM/DCOM.

### 5.7.1. To Create and Configure a New e\*Way Connection

- 1 Select the **e\*Way Connection** folder on the **Components** tab of the e\*Gate Navigator.
- 2 On the palette, click the **Create a New e\*Way Connection** button.
- 3 Enter the name of the e\*Way Connection (for this sample, excel\_connector), then click OK.
- 4 Double-click the new e\*Way Connection to edit its properties.
- 5 The e\*Way Connection Properties window opens. Select **COM/DCOM** from the **e\*Way Connection Type** drop-down menu (see Figure 6).
- 6 Enter the **Event Type "get" interval** in the dialog box provided. 10000 milliseconds is the configured default. The "get interval is the intervening period at which, when subscribed to, the e\*Way connection is polled.

e*Way Connection - excel_connector Properties					
General					
excel_connector					
e*Way Connection Type: COM/DCOM					
Event Type "get" interval					
Time (in milliseconds) after a retrieval returns "no event available" before attempting another retrieval.					
e*Way Connection Configuration File					
configs\MSCOM\excel_connector.cfg					
Clear Find Edit					
OK Cancel <u>A</u> pply Help					

#### Figure 6 e\*Way Connection Properties

- 7 Under e\*Way Connection Configuration File, click the New button.
- 8 The e\*Way Connection editor opens. Select the necessary parameters. For information on the COM/DCOM e\*Way Connection parameters, see Configuring e\*Way Connections on page 16.
- 9 Save the .cfg file. From the **File** menu, click **Promote to Run Time**. The Settings Editor closes.
- **10** Close the e\*Way Connection Properties dialog box.

### 5.8 Creating Intelligent Queues (IQs)

The next step is to create and associate Intelligent Queues (IQs). IQs manage the exchange of information between components within the e\*Gate system, providing non-volatile storage for data as it passes from one component to another. IQs use IQ Services to transport data. IQ Services provide the mechanism for moving Events between IQs, and handling the low-level implementation of data exchange (such as system calls to initialize or reorganize a database).

## To Create and Modify an Intelligent Queue for the COM/DCOM e\*Way

- 1 Select the Navigator's **Components** tab.
- 2 Open the host on which you want to create the IQ.
- 3 Open a **Control Broker**.
- 4 Select an **IQ Manager**.
- 5 On the palette, click the **Create a New IQ** button.
- 6 Enter the name of the new IQ (in this sample, test\_iq), then click OK.
- 7 Double-click the new **IQ** to edit its properties.
- 8 On the General tab, specify the Service and the Event Type Get Interval.

The **STC\_Standard** IQ Service provides sufficient functionality for most applications. If specialized services are required, custom IQ Service DLLs may be created.

The default **Event Type Get Interval** of 100 Milliseconds is satisfactory for the purposes of this initial implementation.

- 9 On the **Advanced** tab, make sure that **Simple publish/subscribe** is checked under the **IQ behavior** section.
- 10 Click OK to close the IQ Properties window

### 5.9 Creating the Collaboration Rules

The next step is to create the Collaboration Rules that extract and process selected information from the source Event Type defined above, according to its associated Collaboration Service. The Default Editor can be set to either **Monk** or **Java**.

From the Schema Designer Task Bar, select **Options** and click **Default Editor**. Make sure editor is set to **Java**.

The sample schema calls for the creation of two Collaboration Rules files.

- trigger\_collab (Pass Through)
- excel\_gate\_collab (Java)

### To Create the Pass Through Collaboration Rules

- 1 Select the Navigator's **Components** tab in the e\*Gate Schema Designer.
- 2 In the Navigator, select the **Collaboration Rules** folder.
- 3 On the palette, click the Create New Collaboration Rules button.
- 4 Enter the name of the new Collaboration Rule Component (for this sample trigger\_collab), then click OK.
- 5 Double-click the new Collaboration Rules Component. The **Collaboration Rules Properties** dialog box opens (seeFigure 7).

🖲 Collabo	oration R	ules - trigger	_collab Proper	ties		×
General	Subscript	ions Publicati	ions Collaboratio	m Mapping		
Î	trigger	_collab				
Service:		Pass Through	ı			<b>•</b>
Initializati	ion string:					
Collabo	pration Rul	BS				
				Clear	ind New	Test
Initializa	ation file —					
					Clear	Find
			ок	Cancel	Apply	Help

Figure 7 Collaboration Properties - Pass Through

- 6 The **Service** field defaults to **Pass Through**.
- 7 Go to the Publications tab. Select Trigger under Available Output Event Types, and click the right arrow to move it to Selected Output Event Types. Make sure the Default option is be enabled.
- 8 Go to the **Subscriptions** tab. Select **Trigger** under **Available Input Event Types**, and click the right arrow to move it to **Selected Input Event Types**. Make sure the checkbox under **Triggering Event** is selected (see Figure 8).

Collaboration Rules - trigger_c	collab Properties	×
Collaboration Rules - trigger_c General Subscriptions Publication Available Input Event Types 00000000 Excel GenericInEvent GenericOutEvent Notification		X
	OK Cancel Apply Help	

### Figure 8 Collaboration Properties - Subscriptions

9 Click OK to close the Collaboration Rules Properties dialog box.

### To Create the Java Collaboration Rules

- 1 Select the Navigator's **Components** tab in the e\*Gate Schema Designer.
- 2 In the Navigator, select the Collaboration Rules folder.
- 3 On the palette, click the Create New Collaboration Rules button.
- 4 Enter the name of the new Collaboration Rule, then click **OK** (for this sample, **excel\_gate\_collab**).
- 5 Double-click the new Collaboration Rules Component to edit its properties. The **Collaboration Rules Properties** dialog box opens.
- 6 From the **Service** field drop-down list box, select **Java**. The **Collaboration Mapping** tab is now enabled, and the **Subscriptions** and **Publications** tabs are disabled.
- 7 In the **Initialization string** field, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
- 8 Select the **Collaboration Mapping** tab (see Figure 9).
- 9 Using the **Add Instance** button, create instances to coincide with the Event Types. For this sample, do the following:
  - A In the **Instance Name** column, enter **trigger** for the instance name.
  - **B** Click **Find**, navigate to **Trigger.xsc**, and double-click to select. **Input.xsc** is added to the **ETD** column for this instance.
  - C In the **Mode** column, select **In** from the drop–down menu available.
  - D In the **Trigger** column, select the box to enable trigger mechanism.
- **10** Repeat steps 9–13 using the following values:

- Instance Name excel
- ETD Excel.xsc
- ◆ Mode In/Out
- Trigger clear
- Manual Publish clear

#### Figure 9 Collaboration Rules - Collaboration Mapping Properties

			g		
ETD		Mode	Trigg	Manual Pub	
Trigger.xsc	Find	In	•	N/A	
Excel.xsc	Find	In/Out			
	Trigger.xsc	Trigger.xsc Find	Trigger.xsc Find In	Trigger.xsc Find In	Trigger.xsc Find In V/A

11 Select the General tab, under the Collaboration Rule box, select New to open the Collaboration Rules Editor. Click OK to close the Collaboration Rules dialog box or click New or Edit under the Collaboration Rules field to open the Collaboration Rules Editor.

### 5.9.1. Using the Collaboration Rules Editor

This section contains an example of how the business logic was defined for the sample schema using the Java Collaboration Rules Editor. The required logic is defined by selecting the desired command and dragging and dropping the nodes (Source or Event) into the corresponding Properties Box, or onto the corresponding node. The resulting code is displayed in the Business Rules pane of the Collaboration Rules Editor.

Each new rule is created by clicking the **rule** button on the Business Rules toolbar. For additional information on using the Java Collaboration Rules Editor, see the *e*\**Gate Integrator User's Guide*. The **excel\_gate\_collab** Business Rules are created as follows:

1 To open the Collaboration Rules Editor to the **excel\_gate\_collab** Collaboration Rules, click **New** or **Edit** under the **Collaboration Rules** field in the Properties dialog box (see above section.) The Collaboration Rules Editor opens. Expand the Editor to full size for optimum viewing and expand the **Source** and **Destination** Events as well.

- 2 Select **retBoolean** in the **Business Rules** pane. All of the user-defined Business Rules are added as part of this method.
- <sup>3</sup> The **set Visible property of \_Application object** rule is created by dragging Visible, under excel, Application, on the Destination Events command node, into the Rule Properties, Rule field (see Figure 10). Place the curser in the last set of parentheses and enter **true** as the value. This creates the following code:

getexcel().get\_Application().setVisible(true)

Scollaboration Rules Editor - excel_gate_collab (Sandbo File Edit View Tools Help	x - Modified)
A	
Source Events	Contraction Events
Image: Second	Path gan Path Separator gan PromptForSummaryInfoga RecordRelative gan ReferenceStyle gan ScreenUpdding gan SheetSinNetWorkhook (gan ShowChartTipNames gan ShowChartTipNames gan ShowChartTipNames gan ShowChartTipNames gan ShowChartTipNames gan ShowChartTipNames gan StatusBar gan StatusBar gan StatusBar gan StatusBar gan TemplateStath gan DefaultSaveFormat gan TransitionMenuKey gan UsableHeight gan UsableWidth gan UsableWidth gan UsableWidth gan UsableWidth gan UsableWidth gan UsableWidth gan UsableWidth gan WindowState gan WindowState gan WindowState gan UsableXitheoChection gan UsableWidth gan Usabl
① try ! catch	
Business Rules	
texcel_gate_collab	Rule         Description:         set Visible property of Application object         Rule:         getexcel().get_Application().setVisible()
	Documentation:

Figure 10 Collaboration Rules Editor

4 Type "set Visible property of \_Application object" in the Rule Properties, Description field. This replaces "rule" as the expression for this rule in the Business Rules pane.

To create the set 'Workbooks' handle rule, drag stcdisphandler under excel, 5 Workbooks, on the Destination Events command node into the Rule Properties, Rule field. From the Source Events command node, select **Workbooks** under excel, Application. Drag-and-drop it into the last set of parentheses in the Rule Properties, Rule field (see Figure 11), to create the following code:

getexcel().getWorkbooks().setstcdisphandler(getexcel().get\_Applica tion().getWorkbooks())

stcdisphandler is the node where dispatch handles are stored.

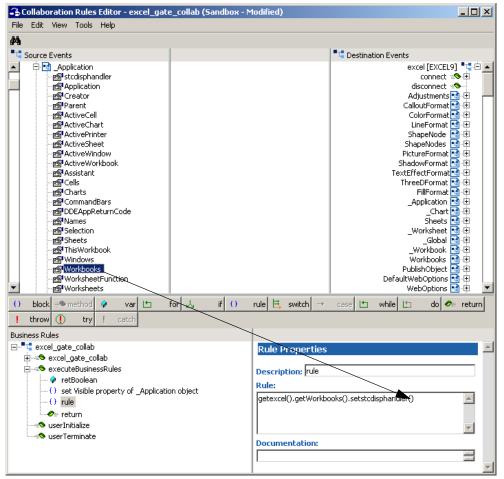


Figure 11 Collaboration Rules Editor

6 To create the set Filename argument for 'Open' rule, drag Filename, under excel, Workbooks, Open, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and type "c\\temp\\test.xls" as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().setFilename("c\\temp\\test.xls
```

7 To create the **set UpdateLinks for 'Open'** rule, drag **error**, under excel, Workbooks, Open, UpdateLinks, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getUpdateLinks().seterror(-
2147352572)
```

- *Note:* A VT-error of -2147352572 indicates that the parameters of the function are unset.
  - 8 To create the **set ReadOnly for 'Open'** rule, drag **error**, under excel, Workbooks, Open, ReadOnly, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -**2147352572** as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getReadOnly().seterror(-
2147352572)
```

9 To create the set Format for 'Open' rule, drag error, under excel, Workbooks, Open, Format, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getFormat().seterror(-
2147352572)
```

10 The set Password for 'Open' rule is created by dragging error, under excel, Workbooks, Open, Password, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getPassword().seterror(-
2147352572)
```

11 The **set WriteResPassword for 'Open'** rule is created by dragging **error**, under excel, Workbooks, Open, WriteResPassword, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter **-2147352572** as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getWriteResPassword().seterror
(-2147352572)
```

12 To create the **set IgnoreReadOnlyRecommended for 'Open'** rule, drag **error**, under excel, Workbooks, Open, IgnoreReadOnlyRecommended, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter **-2147352572** as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getIgnoreReadOnlyRecommended()
.seterror(-2147352572)
```

13 To create the **set Origin for 'Open'** rule, drag **error**, under excel, Workbooks, Open, Origin, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter **-2147352572** as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getOrigin().seterror(-
2147352572)
```

14 The set Delimiter for 'Open' rule is created by dragging error, under excel, Workbooks, Open, Delimiter, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getDelimiter().seterror(-
2147352572)
```

15 The set Editable for 'Open' rule is created by dragging error, under excel, Workbooks, Open, Editable, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getEditable().seterror(-
2147352572)
```

16 To create the set Notify for 'Open' rule, drag error, under excel, Workbooks, Open, Notify, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getNotify().seterror(-
2147352572)
```

17 To create the set Converter for 'Open' rule, drag error, under excel, Workbooks, Open, Converter, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getConverter().seterror(-
2147352572)
```

18 The set AddToMru for 'Open' rule is created by dragging error, under excel, Workbooks, Open, AddToMru, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter -2147352572 as the value. This creates the following code:

```
getexcel().getWorkbooks().getOpen().getAddToMru().seterror(-
2147352572)
```

19 The invoke 'Open' to get \_Workbook rule is created by dragging invoke, under excel, Workbooks, Open, on the Source Events command node, into the Rule Properties, Rule field to create the following code:

```
getexcel().getWorkbooks().getOpen().invoke()
```

20 The **set \_Workbook handle** rule is created by dragging **stcdisphandler**, under excel, Workbook, on the Source Events command node, into the Rule Properties, Rule field. Drag **retval**, under excel, Workbooks, Open, on the Source Events command node, into the last set of parentheses to create the following code:

```
getexcel().get_Workbook().setstcdisphandler(getexcel().getWorkbook
s().getOpen().getretval())
```

21 The **set Worksheets handle** rule is created by dragging **stcdisphandler**, under excel, Sheets, on the Source Events command node, into the Rule Properties, Rule field. Drag **Worksheets**, under excel, Workbook, on the Source Events command node, into the last set of parentheses to create the following code:

```
getexcel().getSheets().setstcdisphandler(getexcel().get_Workbook()
.getWorksheets())
```

22 To create the **set variant parameter of property 'Item' to 1** rule, drag **int**, under excel, Sheets, propget\_Item, Index, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and type 1 to create the following code:

getexcel().getSheets().getpropget\_Item().getIndex().setint(1)

23 The **invoke PROPERTYGET of 'Item'** rule is created by dragging the **invoke** method, under excel, Sheets, propget\_Item, on the Destination Events command node, into the Rule Properties, Rule field. This creates the following code:

```
getexcel().getSheets().getpropget_Item().invoke()
```

24 The **set \_Worksheet handle** rule is created by dragging the **stcdisphandler**, under excel, Worksheet, on the Destination Events command node, into the Rule Properties, Rule field. Drag **retval**, under excel, Sheets, propget\_Item, on the Source Events command node, into the last set of parentheses to create the following code:

```
getexcel().get_Worksheet().setstcdisphandler(getexcel().getSheets().getpropget_Item().getretval())
```

25 The **set 'Cell1' of Range** rule is created by dragging **String**, under excel, Worksheet, propget\_Range, Cell1, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and enter **A1** as the value to create the following code:

getexcel().get\_Worksheet().getpropget\_Range().getCell1().setstring
("A1")

26 The **set 'Cell2' of Range** rule is created by dragging **String**, under excel, Worksheet, propget\_Range, Cell2, on the Destination Events command node, into the Rule Properties, Rule field. Place the cursor in the last set of parentheses and type **A1** as the value to create the following code:

```
getexcel().get_Worksheet().getpropget_Range().getCell2().setstring
("A1")
```

27 To create the **invoke 'Range'** rule, drag the **invoke** method, under excel, Worksheet, propget\_Range, on the Destination Events command node, into the Rule Properties, Rule field. This creates the following code:

getexcel().get\_Worksheet().getpropget\_Range().invoke()

28 To create the **set Range handle** rule, drag the **stcdisphandler** method, under excel, Range, on the Destination Events command node, into the Rule Properties, Rule field. Drag **retval**, under excel, Worksheet, propget\_Range, on the Source Events command node, into the last set of parentheses to create the following code:

```
getexcel().getRange().setstcdisphandler(getexcel().get_Worksheet()
.getpropget_Range().getretval())
```

**29** The **output value from cell A1** rule is created by typing the following code, into the Rule Properties, Rule field:

```
System.err.println("Cell(A1): " +
 ((COMETDVariant)getexcel().getRange().getValue()).getValue())
```

- 30 The Variant variable is created by clicking the variant (var) button on the Business Rules toolbar. A variable appears in the Business Rules under the last-selected rule. In the Variable Properties window, enter Variant as the description, variant as the name, select com.stc.eways.com.COMETDVariant as the type, and enter new COMETDVariant("test value") as the initial value.
- 31 To create the **change Cell 'A1'** rule, drag **Value**, under excel, Range, on the Destination Events command node, into the Rule Properties, Rule field. Place the

cursor in the last set of parentheses and type **variant** as the value to create the following code:

getexcel().getRange().setValue(variant)

- 32 When the business logic has been completed, as displayed in Figure 12, compile and save the Collaboration Rules. From the **File** menu, click **Compile**. The **Save** menu opens, provide a name for the .xpr file (for this sample, excel\_gate\_collab.xpr).
- *Note:* The .xpr file, excel\_gate\_collab.xpr is included with the sample schema on the Installation CD-ROM, and contains the completed version of this sample.

Figure 12	<b>Collaboration Rule</b>	s - Complete
-----------	---------------------------	--------------

Business Rules
🔄 "texcel_gate_collab : public class excel_gate_collab extends excel_gate_collabBase implements JCollaboratorExt
i⊟ieseeteeteeteeteeteeteeteeteeteeteeteetee
. O rule : super();
🛱 🛥 🛇 executeBusinessRules : public boolean executeBusinessRules() throws Exception
🔗 retBoolean : boolean retBoolean = true;
<ul> <li>O set Visible property of _Application object : getexcel().get_Application().setVisible(true);</li> </ul>
- () set 'Workbooks' handle : getexcel().getWorkbooks().setstcdisphandler(getexcel().get_Application().getWorkbooks());
<ul> <li>O set Filename argument for 'Open' : getexcel().getWorkbooks().getOpen().setFilename("c:\\temp\\test.xls");</li> </ul>
<ul> <li>O set UpdateLinks for 'Open' : getexcel().getWorkbooks().getOpen().getUpdateLinks().seterror(-2147352572);</li> </ul>
- () set ReadOnly for 'Open' : getexcel().getWorkbooks().getOpen().getReadOnly().seterror(-2147352572);
- () set Format for 'Open' : getexcel().getWorkbooks().getOpen().getFormat().seterror(-2147352572);
<ul> <li>O set Password for 'Open' : getexcel().getWorkbooks().getOpen().getPassword().seterror(-2147352572);</li> </ul>
- () set WriteResPassword for 'Open' : getexcel().getWorkbooks().getOpen().getWriteResPassword().seterror(-2147352572);
- () set IgnoreReadOnlyRecommended for 'Open': getexcel().getWorkbooks().getOpen().getIgnoreReadOnlyRecommended().seterror(-2147352572);
- () set Origin for 'Open' : getexcel().getWorkbooks().getOpen().getOrigin().seterror(-2147352572);
<ul> <li>O set Delimiter for 'Open': getexcel().getWorkbooks().getOpen().getDelimiter().seterror(-2147352572);</li> </ul>
- () set Editable for 'Open' : getexcel().getWorkbooks().getOpen().getEditable().seterror(-2147352572);
- () set Notify for 'Open' : getexcel().getWorkbooks().getOpen().getNotify().seterror(-2147352572);
<ul> <li>- () set Converter for 'Open' : getexcel().getWorkbooks().getOpen().getConverter().seterror(-2147352572);</li> </ul>
<ul> <li>- () set AddToMru for 'Open' : getexcel().getWorkbooks().getOpen().getAddToMru().seterror(-2147352572);</li> </ul>
<ul> <li>- () invoke 'Open' to get _Workbook : getexcel().getWorkbooks().getOpen().invoke();</li> </ul>
<ul> <li>- () setWorkbook handle : getexcel().get_Workbook().setstcdisphandler(getexcel().getWorkbooks().getOpen().getretval());</li> </ul>
<ul> <li>- () set Worksheets handle : getexcel().getSheets().setstcdisphandler(getexcel().get_Workbook().getWorksheets());</li> </ul>
<ul> <li>O set variant parameter of property 'Item' to 1 : getexcel().getSheets().getpropget_Item().getIndex().setint(1);</li> </ul>
- () invoke PROPERTYGET of 'Item' : getexcel().getSheets().getpropget_Item().invoke();
<ul> <li>O set _Worksheet handle : getexcel().get_Worksheet().setstcdisphandler(getexcel().get5heets().getpropget_Item().getretval());</li> </ul>
() set 'Cell1' of Range : getexcel().get_Worksheet().getpropget_Range().getCell1().setstring("A1");
<ul> <li>- () set 'Cell2' of Range : getexcel().get_Worksheet().getpropget_Range().getCell2().setstring("A1");</li> </ul>
<ul> <li>- () invoke 'Range' : getexcel().get_Worksheet().getpropget_Range().invoke();</li> </ul>
<ul> <li>O set Range handle : getexcel().getRange().setstcdisphandler(getexcel().get_Worksheet().getpropget_Range().getretval());</li> </ul>
- () output value from cell A1 : System.err.println("Cell(A1): " + ((COMETDVariant)getexcel().getRange().getValue()).getValue());
- 🔗 Variant : com.stc.eways.com.COMETDVariant variant = new COMETDVariant("test value");
- () change Cell 'A1' : getexcel().getRange().setValue(variant);
userInitialize : public void userInitialize()
userTerminate : public void userTerminate()

**Note:** Business logic can also be created using the e\*Ways low-level JNI layer. For more information see Java Classes and the Lower JNI Layer on page 48 and Creating Business Rules Using the e\*Way's Low-level JNI Layer on page 48.

38

## 5.10 Creating Collaborations

Collaborations are the components that receive and process Event Types, and then forward the output to other e\*Gate components or to an external directory. Collaborations consist of the Subscriber, which "listens" for Events of a known type (sometimes from a given source), and the Publisher, which distributes the transformed Event to a specified recipient.

### To Create the trigger e\*Way Collaboration

- 1 In the e\*Gate Schema Designer, select the Navigator's **Components** tab.
- 2 Open the host on which you want to create the Collaboration, and select the **Control Broker.**
- 3 Select the **trigger** e\*Way, and from the palette, click the **Create a New Collaboration** button.
- 4 Enter the name of the new Collaboration (for this sample, trigger), then click **OK**.
- 5 Double-click the new Collaboration to edit its properties. The Collaboration trigger Properties dialog box appears (see Figure 13).

🌐 Collaboration - trigger Pi	operties				×
General					
trigger					
Collaboration Rules:					
trigger_collab			-	New	Edit
Subscriptions:					
Event Type		Source			Add
📲 Trigger (Required)		🔜 <external></external>			Delete
					Delete
Publications:	D r r		( n i a i		0.454
Event Type	Destinatio		Priority	5	Add
ete Irigger	😭 test_iq			5	Delete
					Advanced
	ок	Cancel	Ар	alv	Help
				···)	Tiolp

Figure 13 Collaboration Properties - trigger

- 6 From the Collaboration Rules list, select the Collaboration Rules file that you created previously (for this sample, **trigger\_collab**).
- 7 In the **Subscriptions** area, click **Add** to define the input Event Types to which this Collaboration subscribes.
  - A From the **Event Type** list, select the **Event Type** that you previously defined as **Trigger**.

- **B** Select the **Source** from the **Source** list (for this sample, **<External>**).
- 8 In the **Publications** area, click **Add** to define the output **Event Types** that this Collaboration publishes.
  - A From the **Event Type** list, select the **Event Type** that you previously defined as **Trigger**.
  - B Select the publication destination from the **Destination** list (for this sample, **test\_iq**).
  - **C** The Priority field defaults to **5**.
- 9 Click **OK** to close.

### To Create the COM\_Multi\_Mode Collaboration

- 1 In the e\*Gate Schema Designer, select the **Control Broker**.
- 2 Select the e\*Way being assigned the Collaboration (for this sample, excel\_gate), and from the palette, click the Create a New Collaboration button.
- 3 Enter the name of the new Collaboration, then click **OK** (for this sample, **excel\_gate**).
- 4 Double-click the new Collaboration to edit its properties.
- 5 From the Collaboration Rules list, select the Collaboration Rules file that you created previously (for this sample, **excel\_gate\_collab**).
- 6 In the **Subscriptions** area, click **Add** to define the input Event Types to which this Collaboration subscribes (see Figure 13).
  - A From the **Instance Name** list, select the Instance Name that you previously defined (for this sample, **trigger**).
  - **B** From the **Event Type** list, select the **Event Type** that you previously defined (for this sample, **Trigger**).
  - C Select the **Source** from the **Source** list (for this sample, **trigger**).
- 7 In the **Publications** area, click **Add** to define the output **Event Types** that this Collaboration publishes (see Figure 14).
  - A From the **Instance Name** list, select the **Instance Name** that you previously defined as **excel**.
  - **B** From the **Event Types** list, select the **Event Type** that you previously defined (for this sample, **Trigger**).
  - C Select the publication destination from the **Destination** list (for this sample, **excel\_connector**).
  - **D** The Priority field defaults to **5**.

Figure 14	Collaboration	Properties
-----------	---------------	------------

Jeneral				
excel_ga	ite			
Collaboration Rules:				
excel_gate_collab			<ul> <li>New</li> </ul>	Edit
Subscriptions:				
Instance Name	Event Type	Source		Add
trigger	Trigger	💥 trigger		Delete
Publications:	Event Type	Destination	Priority	Add
	Event Type	Destination	Priority 5	
Instance Name				Add

8 Click **OK** to close.

# 5.11 Configuring the DCOM Server

1 To configure the DCOM Server, select the Windows **Run** option. When the Run dialog box opens, type **dcomcnfg** in the **Open** field, and click **OK** (see Figure 15).

### Figure 15 Run Dialog

Run	<u>? ×</u>
<u> </u>	Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:	dcomenfg
	OK Cancel Browse

2 The Distributed COM Configuration Properties lists the registered applications. Select the Microsoft Excel application (see Figure 16). Click Properties.

Distributed COM Configuration Properties	? ×
Applications Default Properties Default Security Default Protocols	
Applications         Default Properties         Default Security         Default Protocols           Applications:         LView Pro Image         Machine Debug Manager         Media Player           Media Player         Microsoft Access Application         Microsoft Agent Server 2.0           Microsoft Development Environment         Microsoft Development Environment           Microsoft Excel Application         Microsoft Regulation 3.0           Microsoft Revel Application         Microsoft Revel Application           Microsoft NetShow Real-Time Encoder Callback Proxy         Microsoft Office Binder           Microsoft PowerPoint Presentation         Microsoft Schedule+ 7.0 Application           Microsoft Wisio Drawing         Microsoft WBEM Active Scripting Event Consumer Provider	
Properties	
OK Cancel App	ly

Figure 16 Distributed COM Configuration Properties

3 The Microsoft Excel Application Properties dialog box appears. Select the **Security** tab (see Figure 17).

Microsoft Excel Application Properties	<u>? ×</u>
General Location Security Identity Endpoints	
<ul> <li>Use default access permissions</li> <li>Use custom access permissions</li> <li>You may edit who can access this application.</li> </ul>	
<ul> <li>Use default launch permissions</li> <li>Use custom launch permissions</li> <li>You may edit who can launch this application.</li> </ul>	
<ul> <li>Use default configuration permissions</li> <li>Use custom configuration permissions</li> <li>You may edit who can change the configuration information for this application.</li> </ul>	
OK Cancel App	oly

### Figure 17 Application Properties - Security Tab

4 Select the Use custom access permission option button and click Edit. The Add Users and Groups dialog box appears (see Figure 18).

Add Users and Groups
List Names From: 🔊 STC
Names:
Architecture Competency Center
All authenticated Users All authenticated users
Cert Publishers Enterprise certification and renewal ager
Contract Legal & Contract Group (Samanthe Beck
Contractors PSFT Temp Group for Mark Nemets
🥵 Default Profiles
C C C C C C C C C C C C C C C C C C C
Comparison of the second
Add Show Users Members Search
Add Names:
Type of Access: Allow Access
OK Cancel <u>H</u> elp

### Figure 18 Add Users and Groups - Access

- 5 Select the appropriate users to whom access is granted (for example, Authenticated Users), and click Add.
- 6 In the **Type of Access** field, select **Allow Access**, and click **OK**.
- 7 In the Application Properties window (see Figure 17), select the Use custom launch permission option button and click Edit.
- 8 Select the appropriate users to whom launch permission is granted (for example, Authenticated Users), and click Add.
- 9 In the **Type of Access** field, select **Allow Launch**, and click **OK**.
- 10 In the Application Properties window (see Figure 17), select the **Identity** tab (see Figure 19).

Microsoft Excel Application Properties	<u>?</u> ×
General Location Security Identity Endpoints	
Which user account do you want to use to run this application?	
The interactive user	
C The launching user	
C This user:	
User: Browse.	
Password:	
Confirm Password:	
C The System Account (services only)	
OK Cancel Ap	oply

### Figure 19 Application Properties - Identity Tab

- 11 Select **The interactive user** as the user account to use to run the application.
- 12 Click **OK**. The DCOM server is now configured for the sample schema.

# 5.12 Sample Schema

A sample of the COM/DCOM e\*Way is included on the Installation CD. The sample demonstrates e\*Gate's interaction with a COM/DCOM enabled application. The previous sections provided the basics for implementing the COM/DCOM e\*Way. This section describes how to implement the COM/DCOM e\*Way within a sample schema.

This implementation consists of one file-based e\*Way, one Multi-Mode e\*Way, two Event Type Definitions, two Collaboration Rules, one Intelligent Queues and two Collaborations, as follows:

- **trigger** This e\*Way receives input from an external source, applies pass through Collaboration Rules, and publish the information to an Intelligent Queue.
- **excel\_gate** This Multi-Mode e\*Way applies extended Java Collaboration Rules to an inbound Event to perform the desired business logic.
- **Excel** This Event Type contains the methods and properties to be used to perform the necessary transformation.
- **Trigger** This Event Type describes an Event that is input to the extended Java Collaboration Service.
- **trigger\_collab** This Collaboration Rule is associated with the **trigger** e\*Way, and is used for receiving the input Event.
- **excel\_gate\_collab** The Collaboration Rule is associated with the **excel\_gate** Multi-Mode e\*Way, and is used to perform the transformation process.
- test\_iq This Intelligent Queue is a STC\_Standard IQ,.

### 5.12.1. Completing the Configuration of the ExcelSample Schema

The sample schema is nearly ready to use as it is presently configured. Check the e\*Way Connection settings to make sure that they are appropriate for your specific system. Also do the following:

- 1 In the **Multi-Mode e\*Way configuration**, **mscom config**, enter the ProgID for the host (see **ProgID** on page 17). For DCOM the Host name also needs to be entered (see **Hostname** on page 18).
- 2 Create two folders on the C: drive named DATA and INDATA.

### 5.12.2. Executing the Schema

### To execute the COM/DCOM sample schema

1 Go to the command line prompt, and enter the following:

stccb -rh hostname -rs schemaname -un username -up user password -ln hostname\_cb

Substitute *hostname, schemaname, username* and *user password* as appropriate.

2 Start the Schema Manager. Specify the server that contains the Control Broker you started in Step 1 above.

- 3 Select the COM/DCOM sample schema.
- 4 Verify that the Control Broker is connected. To do this, select and right-click the Control Broker in the Schema Manager, and select **Status**. (The message in the Control tab of the console will indicate command *succeeded* and status as *up*.)
- 5 Select the IQ Manager, *hostname\_*igmgr, then right-click and select **Start**. (This will already be started if **Start automatically** is selected in the IQ Manager properties.)
- 6 Select each of the e\*Ways, right-click select **Start**. (These will already be started if **Start automatically** is selected in the e\*Way's properties.)
- 7 To view the output, copy the output file (specified in the Outbound e\*Way configuration file). Save to a convenient location, open.

*Note:* Opening the destination file while the schema is running will cause errors.

# **5.13** COM/DCOM Run-Time Exceptions

The e\*Way Intelligent Adapter for COM/DCOM can be conceptually divided into two layers. The low-level JNI code that wraps the COM IDispatch interface and the higher-level ETD 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. In general, a value of zero indicates success, greater than zero indicates a warning, and less than zero indicates an error.

The low level JNI code has always returned this HRESULT to the builder-generated code. However, the builder-generated ETD code does nothing with the HRESULT. It was possible, then, that the Collaboration could fail to create an instance of the component or a method on the component could fail, and the error may never be caught.

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 ETD code (because the ETD 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.

For more information on the methods of the STCComException Class refer to the COM/DCOM e\*Way Javadoc (see COM/DCOM e\*Way Javadoc on page 52).

# 5.14 Java Classes and the Lower JNI Layer

The COM/DCOM Java classes can be thought of as a series of layers (see Figure 20). That is, an ETD that is generated using the ETD builder utilizes the **COMETDVariant**, **COMETDWrapper**, and **ComEWayConnector** classes. These classes, in turn, "wrap" the lower level JNI layer classes.

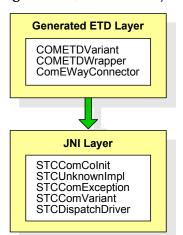


Figure 20Java Class Layers

When business logic is created using the "drag and drop" feature in the Collaboration Editor, it uses the classes shown in the top layer. It is also possible, however, to drop down to the JNI layer in order to perform other specialized tasks.

# 5.15 Creating Business Rules Using the e\*Way's Low-level JNI Layer

Typically, Business Rules are created using the Collaboration Editor via features such as "drag and drop." There may be times, however, when you need to "get a little closer to the metal." This section discusses the **low-level JNI layer**, and provides an overview of how it is used to define business logic. It is assumed that the reader of this section is well-versed in COM and COM automation, and therefore, does not attempt to address the finer points of COM or any of the semantics of the IDispatch interface.

The two classes typically used are the **STCDispatchDriver** class and the **STCComVariant** class.

The **STCDispatchDriver** class is a shim layer (wrapper) for the **COM IDispatch interface** and supports calling **Invoke** on the underlying automation-compatible object. It also allows for getting and setting properties on the IDispatch object.

The **STCComVariant** class wraps the concept of the COM VARIANT. As in the COM VARIANT structure, the members of the **STCComVariant** class are public, to mimic the union in its COM counterpart. There is also the familiar "**vt**" member variable that

specifies the VARTYPE of the data currently being held, but with a slight deviation in respect to the VT BYREF and VT ARRAY specifiers. In the COM counterpart these are bitflags that are or'd in to the vt member. In the STCComVariant there are separate member variables to hold these.

### Creating an instance of a COM object

To create an instance of a COM object, call create on the STCDispatchDriver, passing the ProgID for the object and the server name, if applicable.

The server name parameter can be **null**. It is only needed when the object is a DCOM object residing on a different server. For example:

```
STCDispatchDriver driver = new STCDispatchDriver();
int r = driver.create("Tester2.ComEWayTest", null);
System.out.println("create = " + Integer.toHexString(hr));
```

Here, we are creating an instance of an object identified by the ProgID **Tester2.ComEWayTest**. The server is **null** since it is not applicable here. Though this **Tester2.ComEWayTest** does happen to be a DCOM server (that is, an executable), it resides on the same box, and the DCOM creation semantics use the local host as the default, unless it is registered to run on another server (via the dcomcnfg.exe utility).

*Note:* The create method (and other methods on the STCDispatchDriver) can possibly throw an exception of type **STCComException**. So, technically the driver.create method should be wrapped within a try/catch block, but that is omitted here for *clarity*.

### Calling a method on the created object

In COM automation, all methods can be called by name. For example, assume that the Tester2.ComEWayTest object has a method declared in IDL as follows:

[id(1), helpstring("Add two longs and return the result")] HRESULT AddTwo([in] long a1, [in] long a2, [out,retval] long\* result);

The procedure for calling this method would be as follows:

```
// Build up the call stack. Both arguments are of type long // which is a vartype of VT_I4. Note that, as in IDispatch
\ensuremath{{\prime}}\xspace // Invoke, the arguments are packed in the VARIANT args in
// reverse order.
STCComVariant arg1 = new STCComVariant(1, STCComVariant.VT I4);
STCComVariant arg2 = new STCComVariant(2, STCComVariant.VT_I4);
STCComVariant[] args = new STCComVariant[2];
args[0] = arg2;
args[1] = arg1;
// This will be used to hold the return value
STCComVariant out = new STCComVariant();
// Invoke the AddTwo function
int r = driver.invoke("AddTwo", args, out);
System.out.println("invoke = " + Integer.toHexString(hr));
System.out.println("out.intValue = " + out.intValue);
// Release the reference count
driver.Release();
```

Single dimension arrays of most of the supported types are also supported. For example, the following assumes the object supported a method declared in IDL:

```
[id(2), helpstring("Add all the shorts in the short array and
return the result")]
HRESULT AddShortSA([in] SAFEARRAY(short)* psa,
        [out,retval] long* result);
```

The procedure for calling this method would be as follows:

```
// Create an array to add
short[] arr = new short[10];
for (int n = 0; n < 10; n++)
    arr[n] = 10;
// Set up the array argument
STCComVariant arg1 = new STCComVariant();
arg1.pshortValue = arr;
arg1.vt = STCComVariant.VT_I2;
arg1.isArray(true);
arg1.setArrayDim(1);
// pack 'em up - reverse order
STCComVariant[] args = new STCComVariant[1];
args[0] = arg1;
STCComVariant out = new STCComVariant();
int r = driver.invoke("AddShortSA", args, out);
System.out.println("invoke = " + Integer.toHexString(hr));
System.out.println("out.intValue = " + out.intValue);
driver.Release();
```

#### Calling a method that returns another IDispatch object

These classes can also be used to call a method that returns another IDispatch object. This example may be somewhat contrived, but consider the following case: A COM object is able to return an instance of a component specified by a ProgID, as represented by the following IDL:

```
[id(20), helpstring("Get IDispatch specified by progid")]
HRESULT GetApp([in] BSTR sProgID,
       [out,retval] IDispatch** ppOut);
```

The procedure for calling this method would be as follows:

```
// This is the ProgID of the application we want it to return
STCComVariant arg1 = new STCComVariant("Vim.Application");
// Pack up the arguments - reverse order
STCComVariant[] args = new STCComVariant[1];
args[0] = arg1;
STCComVariant out = new STCComVariant();
int r = driver.invoke("GetApp", args, out);
System.out.println("invoke = " + Integer.toHexString(hr));
// We're done with this driver now. The returned IDispatch,
// if invoke succeeded, is in out.dispValue
driver.Release();
// In this case we know that the application - gvim in this case -
// supports a method named SendKeys which takes a string
```

```
STCComVariant xarg1 = new STCComVariant("iHello gvim from
java!<ESC>");
STCComVariant[] xargs = new STCComVariant[1];
xargs[0] = xarg1;
STCComVariant xout = new STCComVariant();
r = out.dispValue.invoke("SendKeys", xargs, xout);
out.dispValue.Release();
```

#### Setting and getting properties

The procedure for setting a property or getting a property is very simple. The general procedure is as follows:

```
// set the property
String msg = new String("Some string property");
STCComVariant vMsg = new STCComVariant(msg);
r = driver.setProp("TheStringProperty", vMsg);
// now read it back
STCComVariant vMsgOut = new STCComVariant();
r = driver.getProp("TheStringProperty", vMsgOut);
```

### Chapter 6

# e\*Way Classes and Methods

The COM/DCOM e\*Way contains a number of Java methods that have been exposed to make it easier for the user to set information in the e\*Way ETD Editor and to get information from it. These methods are contained in the following classes:

- The COMETDVariant Class
- The COMETDWrapper Class
- The ComEWayConnector Class
- The STCComCoInit Class
- The STCComException Class
- The STCComVariant Class
- The STCDispatchDriver Class
- The STCUnknownImpl Class

For more information on the Generated ETD Class layer and the low-level JNI layer see **Java Classes and the Lower JNI Layer** on page 48.

### 6.0.1. COM/DCOM e\*Way Javadoc

For a complete list of the Java methods within the classes listed above, refer to the **Javadoc**.

# Index

### B

Business Rules using the Collaboration Rules Editor 32 using the low-level JNI layer 48

### C

Classpath Override 12 Classpath Prepend 12 Collaboration Rules creating 29 creating Business Rules 32 editor 32 Collaborations 39 collaborations 39 for the Multi-Mode e\*Way 40 COM 8 COM/DCOM runtime exceptions 47 COMETDVariant class 48, 52 COMETDWrapper class 48, 52 ComEWayConnector class 48, 52

### D

Disable JIT 13

### E

e\*Way Connection creating 27 parameters SME 16 e\*Way connection configuration 16 e\*Ways creating and configuring 25 error messages 47 Event Type Definitions 22 creating from an existing .xsc 22 creating from the COMWizard 23 Event Types 22 exceptions 47 external application requirements 7

### F

files/directories created by installation 9

### I

implementation 19 Initial Heap Size 13 installation 8 directories created 9 files/directories 9 pre-installation 8 Windows 2000 8 Windows NT 8 intelligent queues 29 intended reader 6

### J

Java class layers JNI DLL Absolute Pathname JNI layer calling a method on a created object calling a method that returns a IDispatch object **50** creating an instance of a COM object setting and getting properties IVM settings

### Μ

Maximum Heap Size 13 Multi-Mode e\*Way 10 configuration 10 configuration parameters 10 Auxiliary JVM Configuration File 14 **CLASSPATH Append From Environment** Variable 12 CLASSPATH Override 12 CLASSPATH Prepend 12 Disable JIT 13 JNI DLL Absolute Pathname 11 Maximum Heap Size 13 Maximum Stack Size for IVM Threads 13 Maximum Stack Size for Native Threads 13 Remote Debugging port number 14 Suspend option for debugging 14 parameters 10

### 0

53

operating systems 7 overview COM/DCOM 6 sample implementation 20

### Ρ

parameters Class 17 connector 16 host 18 Multi-Mode e\*Way CLASSPATH prepend 12 Initial Heap Size 13 JNI DLL absolute pathname 11 JVM settings 10 Maximum Heap Size 13 progid 17 Type 17

### R

runtime exceptions 47

### S

sample schema 46 executing the schema 46 STCComCoInit class 52 STCComException class 47, 52 STCComVariant class 52 STCDispatchDriver class 52 STCUnknownImpl class 52 system requirements 7 external 7

### Τ

test.xls creating 21

### U

ulti 10

### W

Windows 2000 installation 8 Windows NT installation 8

54