

**SeeBeyond™ eBusiness Integration Suite**

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

*Release 4.5.3*

*Java Version*



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

e\*Gate, e\*Insight, e\*Way, e\*Xchange, e\*Xpressway, eBI, iBridge, Intelligent Bridge, IQ, SeeBeyond, and the SeeBeyond logo are trademarks and service marks of SeeBeyond Technology Corporation. All other brands or product names are trademarks of their respective companies.

© 1999–2002 by SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

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

Version 20021106104254.

# Contents

---

## Chapter 1

|                                   |          |
|-----------------------------------|----------|
| <b>Introduction</b>               | <b>7</b> |
| COM/DCOM Overview                 | 7        |
| Intended Reader                   | 7        |
| Supported Operating Systems       | 8        |
| System Requirements               | 8        |
| External Application Requirements | 9        |

---

## Chapter 2

|   |           |
|---|-----------|
| <b>Installation</b>                           | <b>10</b> |
| Windows NT and Windows 2000 Installation      | 10        |
| Pre-installation                              | 10        |
| Installation Procedure                        | 10        |
| Files/Directories Created by the Installation | 11        |

---

## Chapter 3

|  |           |
|--|-----------|
| <b>Multi-Mode e*Way Configuration</b>      | <b>12</b> |
| Multi-Mode e*Way                           | 12        |
| JVM Settings                               | 12        |
| JNI DLL Absolute Pathname                  | 13        |
| CLASSPATH Prepend                          | 14        |
| CLASSPATH Override                         | 14        |
| CLASSPATH Append From Environment Variable | 14        |
| Initial Heap Size                          | 15        |
| Maximum Heap Size                          | 15        |
| Maximum Stack Size for Native Threads      | 15        |
| Maximum Stack Size for JVM Threads         | 15        |
| Disable JIT                                | 15        |
| Remote Debugging port number               | 16        |
| Suspend option for debugging               | 16        |
| Auxiliary JVM Configuration File           | 16        |
| General Settings                           | 16        |
| Rollback Wait Interval                     | 16        |
| Standard IQ FIFO                           | 17        |

---

Chapter 4

|                                       |           |
|---------------------------------------|-----------|
| <b>e*Way Connection Configuration</b> | <b>18</b> |
| <b>Configuring e*Way Connections</b>  | <b>18</b> |
| Connector                             | 18        |
| Type                                  | 19        |
| Class                                 | 19        |
| COM/DCOM Configuration                | 19        |
| ProgID                                | 19        |
| Hostname                              | 20        |

---

Chapter 5

|  |           |
|--|-----------|
| <b>Implementation</b>  | <b>21</b> |
| <b>Implementation Notes</b>                                      | <b>21</b> |
| <b>COM/DCOM Sample Implementation Overview</b>                   | <b>22</b> |
| Importing the Sample Schema                                      | 23        |
| <b>Creating the Sample Schema Manually</b>                       | <b>23</b> |
| <b>Creating Event Type Definitions</b>                           | <b>24</b> |
| Creating an Event Type Definition Using an Existing .xsc         | 24        |
| Creating an Event Type Definition Using the COM ETD Wizard       | 25        |
| <b>Creating and Configuring the Component e*Ways</b>             | <b>27</b> |
| To Create the Trigger e*Way                                      | 27        |
| To Create the COM/DCOM e*Way (Multi-Mode)                        | 28        |
| <b>Creating the e*Way Connection</b>                             | <b>29</b> |
| To Create and Configure a New e*Way Connection                   | 30        |
| <b>Creating Intelligent Queues (IQs)</b>                         | <b>31</b> |
| To Create and Modify an Intelligent Queue for the COM/DCOM e*Way | 31        |
| <b>Creating the Collaboration Rules</b>                          | <b>31</b> |
| To Create the Pass Through Collaboration Rules                   | 32        |
| To Create the Java Collaboration Rules                           | 33        |
| Using the Collaboration Rules Editor                             | 34        |
| <b>Creating Collaborations</b>                                   | <b>40</b> |
| To Create the trigger e*Way Collaboration                        | 41        |
| To Create the COM_Multi_Mode Collaboration                       | 42        |
| <b>Configuring the DCOM Server</b>                               | <b>44</b> |
| <b>Sample Schema</b>   | <b>48</b> |
| Completing the Configuration of the COMtest Schema               | 48        |
| Executing the Schema   | 48        |
| <b>COM/DCOM Run-Time Exceptions</b>                              | <b>49</b> |

## Chapter 6

|                                      |           |
|--------------------------------------|-----------|
| <b>Java Methods</b>                  | <b>50</b> |
| <b>The COMETDVariant Class</b>       | <b>50</b> |
| Methods of the COMETDVariant Class   | 50        |
| COMETDVariant                        | 51        |
| getType                              | 52        |
| getValue                             | 52        |
| setbool                              | 53        |
| getbool                              | 53        |
| setshort                             | 53        |
| getshort                             | 54        |
| setbyte                              | 54        |
| getbyte                              | 55        |
| setint                               | 55        |
| getint                               | 55        |
| seterror                             | 56        |
| geterror                             | 56        |
| setstring                            | 57        |
| getstring                            | 57        |
| setfloat                             | 57        |
| getfloat                             | 58        |
| setdouble                            | 58        |
| getdouble                            | 59        |
| getarray                             | 59        |
| setarray                             | 59        |
| <b>The COMETDWrapper Class</b>       | <b>60</b> |
| Methods of the COMETDWrapper Class   | 60        |
| COMETDWrapper                        | 61        |
| COMETDWrapper_Init                   | 61        |
| setParam                             | 62        |
| getByteArray                         | 67        |
| getByte2DArray                       | 68        |
| getBooleanArray                      | 68        |
| getBoolean2DArray                    | 69        |
| getShortArray                        | 69        |
| getShort2DArray                      | 70        |
| getIntArray                          | 70        |
| getInt2DArray                        | 71        |
| getFloatArray                        | 71        |
| getFloat2DArray                      | 72        |
| getDoubleArray                       | 72        |
| getDouble2DArray                     | 73        |
| getStringArray                       | 73        |
| getString2DArray                     | 74        |
| getVariantParam                      | 74        |
| getRetVal                            | 75        |
| setdisphandler                       | 75        |
| invoke                               | 75        |
| <b>The STCComException Class</b>     | <b>76</b> |
| Methods of the STCComException Class | 76        |
| Fields of the STCComException Class  | 77        |
| STCComException                      | 77        |
| getMessage                           | 77        |
| getErrorCode                         | 78        |
| getSystemErrorText                   | 78        |

Index

# 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™-enabled COM/DCOM e\*Way is installed, configured, and implemented.

---

## 1.1 COM/DCOM Overview

The Microsoft™ *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 NT™ or Windows 2000™ 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 2000 SP1, Windows 2000 SP2, and Windows 2000 SP3
- Windows NT 4 SP6a

**Note:** *If you intend to install e\*Gate onto a totally clean Windows NT computer, you must first download a patch from Microsoft. If you do not perform the download, the installation of the SeeBeyond Editors (MSI Installer) fails. Download the patch from: [http://www.microsoft.com/data/download\\_250rtm.htm](http://www.microsoft.com/data/download_250rtm.htm)*

To confirm that Windows NT SP6a is installed

- 1 At the command prompt, type **Winver**.
- 2 The **About Window NT (R)** window appears (this is the same window that appears when you choose **Help** on the Menu bar and click **About Windows NT** with Microsoft Explorer open). However, by using the command prompt, a line with **Revised Service Pack** displays on the window with the correct version listed.

---

## 1.4 System Requirements

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

- An e\*Gate Participating Host, version 4.5.1 or later.
- 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.

# 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 NT and Windows 2000 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.

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

---

## 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.

**Table 1** Files Created by the Installation

| e*Gate Directory | File(s)  |
|------------------|--|
| bin\             | stccombuilder.exe<br>stccomjavabuilder.exe<br>stccomutil.dll         |
| bin\java         | gnu-regexp-1.1.1.jar<br>jcscomp.jar<br>stcjintegra.jar<br>xerces.jar |
| classes\         | ewscm.jar<br>ewmscm.jar<br>stcjcs.jar                                |
| configs\MSCOM\   | com.def  |
| etd\             | com.ctl<br>comwizard.ctl<br>stcewmscm                                |

# 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 Enterprise Manager.

### 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 12
- [General Settings](#) on page 16

#### 3.1.1. JVM Settings

The JVM Settings control basic Java Virtual Machine settings.

- [JNI DLL Absolute Pathname](#) on page 13
- [CLASSPATH Prepend](#) on page 14

- **CLASSPATH Override** on page 14
- **CLASSPATH Append From Environment Variable** on page 14
- **Initial Heap Size** on page 15
- **Maximum Heap Size** on page 15
- **Maximum Stack Size for Native Threads** on page 15
- **Maximum Stack Size for JVM Threads** on page 15
- **Disable JIT** on page 15
- **Remote Debugging port number** on page 16
- **Suspend option for debugging** on page 16
- **Auxiliary JVM Configuration File** on page 16

---

## 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 NT 4.0/ Windows 2000 | 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 (NT).*

---

## 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.

*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.

### 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.

---

## 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 16
- [Standard IQ FIFO](#) on page 17

---

## 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.

# 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 Enterprise Manager.

To create and configure e\*Way Connections

- 1 In the Enterprise Manager'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 18
- **COM/DCOM Configuration** on page 19

### 4.1.1. Connector

This section contains a set of top level parameters.

- **Type** on page 19
- **Class** on page 19

---

## Type

### 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 19
- [Hostname](#) on page 20

---

## 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 precedence 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:

```
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 precedence 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.*

# 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:

**Table 2** Supported Data Types

| OLE Data Types | Description                                    |
|----------------|--|
| VT_I2          | 2-byte signed int                              |
| VT_I4          | 4-byte signed int                              |
| VT_R4          | 4-byte real                                    |
| VT_R8          | 8-byte real                                    |
| VT_BSTR        | Binary string                                  |
| VT_DISPATCH    | IDispatch (1 dimensional 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). |

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

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

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.*

**Note:** *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.*

---

## 5.2 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 Enterprise Manager 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 Enterprise Manager, 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 prior to being sent to the external system.

The following sections describe how to define and associate each of the above components, and [Sample Schema](#) on page 48, provides the details necessary to create the components of a sample schema consisting of three e\*Ways, three Event Types, one Collaboration Rule, one Intelligent Queue and three Collaborations.

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.

### 5.2.1. 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 Enterprise Manager 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 **COMtest**, or any appropriate name as desired.
- 5 Select **Create from export**, locate the **ExcelSample.zip** on the CD, and click **Open**.

The e\*Gate Enterprise Manager 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**.

---

## 5.3 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.

## 5.4 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.

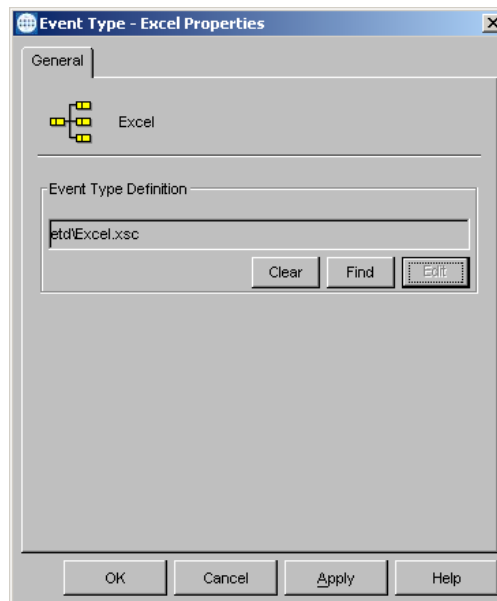
In COM, the Event Type corresponds to the object being used (it's methods). These methods are exposed in the Collaboration to allow the user to get and set the properties or call one or more methods. In addition, methods are made available to get and set the objects properties.

### 5.4.1. 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).

**Figure 1** Event Type - Properties

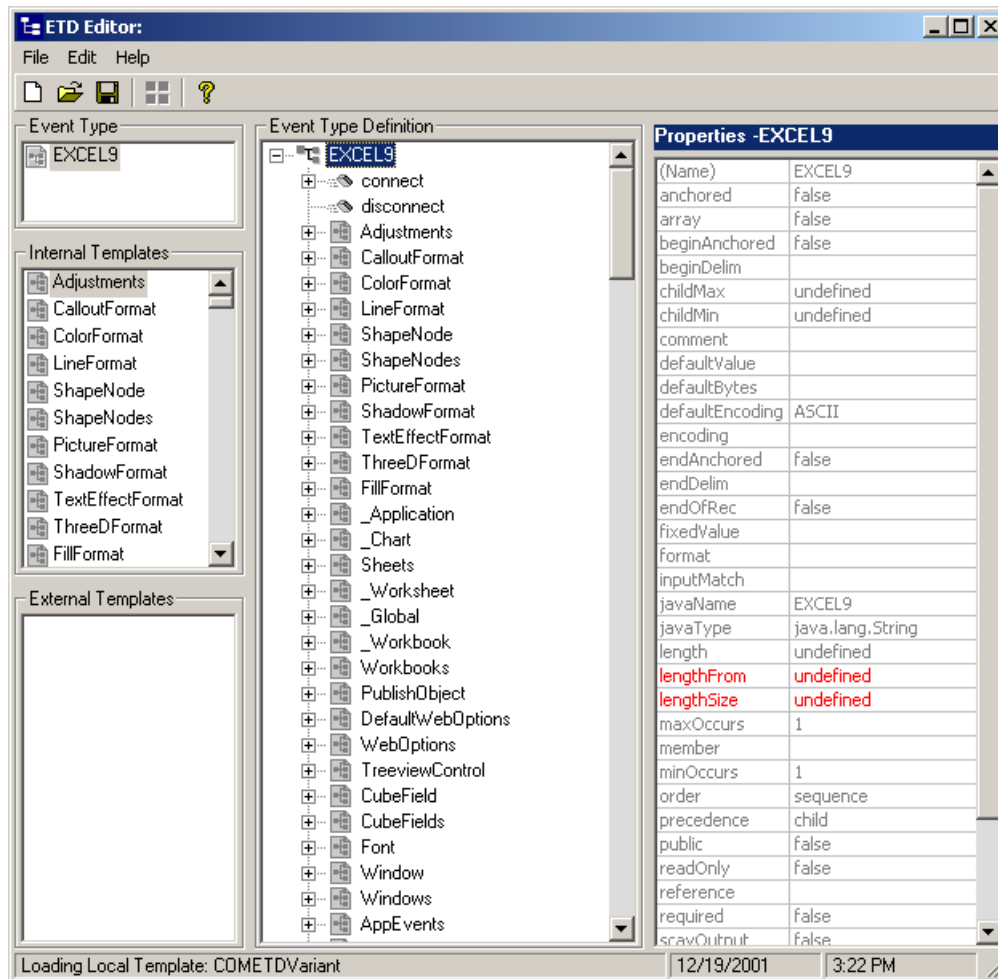


- 5 Click **Find** under the Event Type Definition field. Find and select **Excel.xsc**.



- 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.

**Figure 2** Event Type Definition Editor - Excel



- On the File menu, click **Compile and Save**. This should be done after any modifications are made to the file.
- 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.
- Close the ETD Editor. Click **OK** to close the Event Type Properties Dialog box.

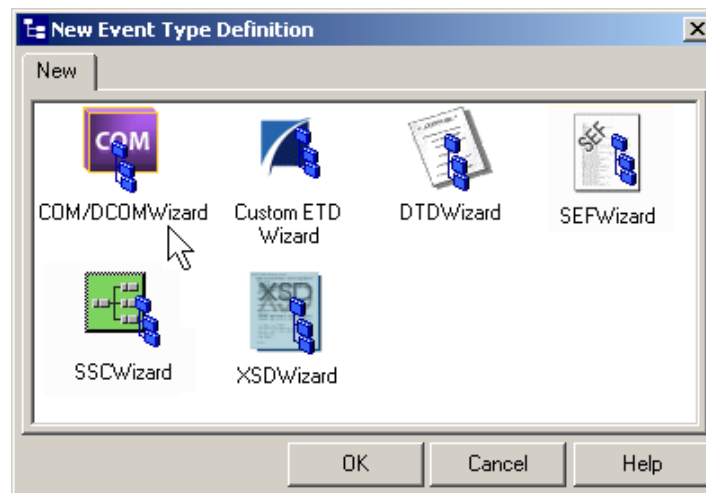
### 5.4.2. Creating an Event Type Definition Using the COM ETD Wizard

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.

- 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**.

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

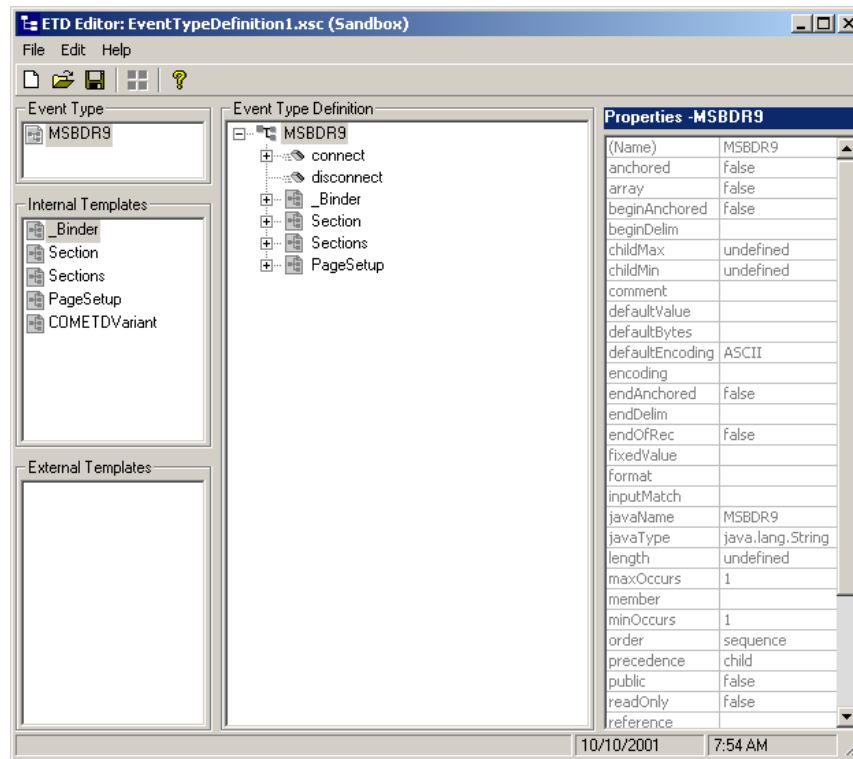


- 8 Enter a Package Name where the ETD builder can place all the generated Java classes associated with the created ETD (for this sample, **com.stc.COMtest**).
- 9 Enter the **COM Type** library file from which the **.xsc** file is created. TypeLib files are \*.tlb, \*.olb, \*.dll, \*.ocx, and \*.exe 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. The ETD is read-only, allowing for no further configuration from the ETD Editor (see Figure 4).

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.

## 5.5 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 case "**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.

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

| Parameter  | Value                         |
|--|-------------------------------|
| <b>General Settings (unless otherwise stated, leave settings as default)</b> |                               |
| AllowIncoming  | YES                           |
| AllowOutgoing  | NO                            |
| <b>Outbound Settings</b>   | Default                       |
| <b>Poller Inbound Settings</b>   |                               |
| 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                             |
| <b>Performance Testing</b>   | Default                       |

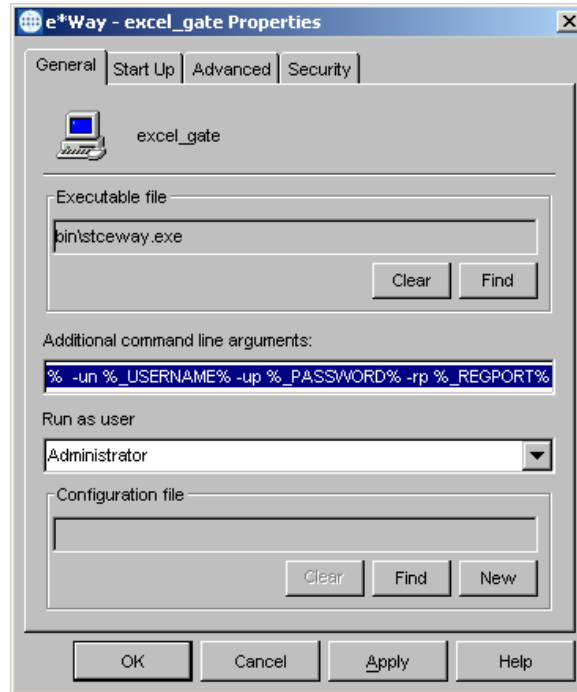
- 9 After selecting the desired parameters, save the **configuration** file (as “**trigger.cfg**”).
- 10 From the **File** menu, click **Promote to Run Time**. This closes the .cfg file.
- 11 In the e\*Way - Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default 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 case, “**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).

**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 11.
- 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.

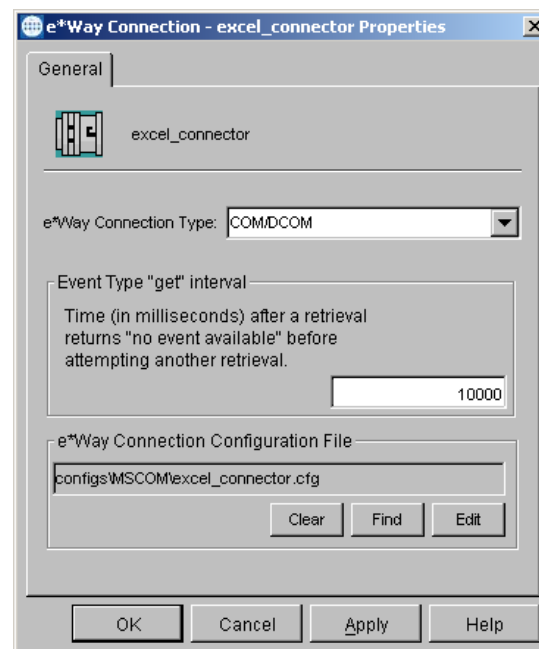
## 5.6 Creating the e\*Way Connection

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

### 5.6.1. 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.

**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 17.
- 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.

---

## 5.7 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, 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 case “**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

---

## 5.8 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 Enterprise Manager Task Bar, select **Options** and click **Default Editor**. Make sure editor is set to **Java**.

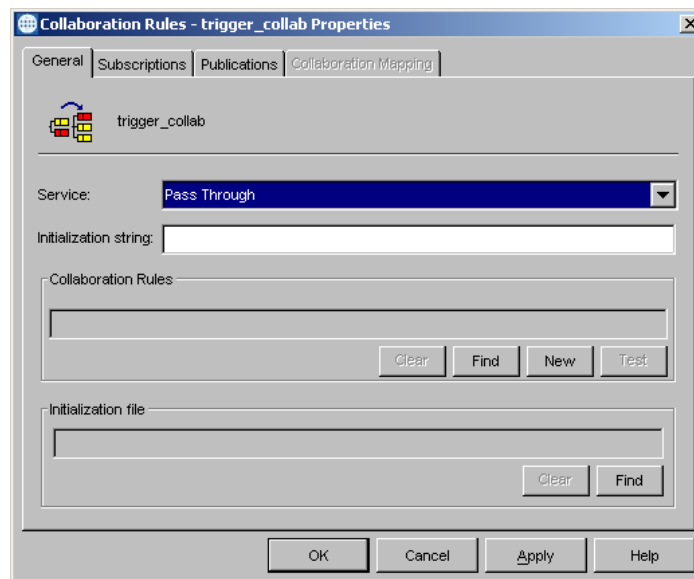
The sample schema calls for the creation of two Collaboration Rules files.

- **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 Enterprise Manager.
- 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 case "trigger\_collab"), then click **OK**.
- 5 Double-click the new Collaboration Rules Component. The **Collaboration Rules Properties** dialog box opens (see Figure 7).

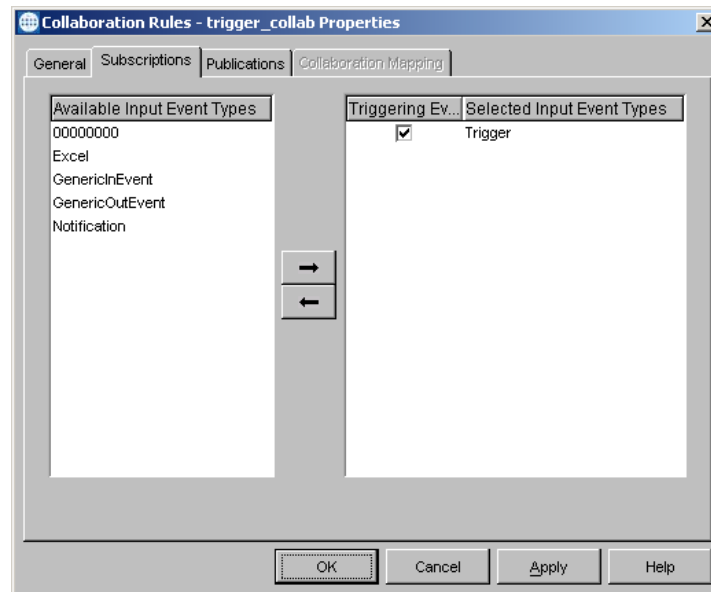
**Figure 7** Collaboration Properties - Pass Through



- 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).



**Figure 8** Collaboration Properties - Subscriptions



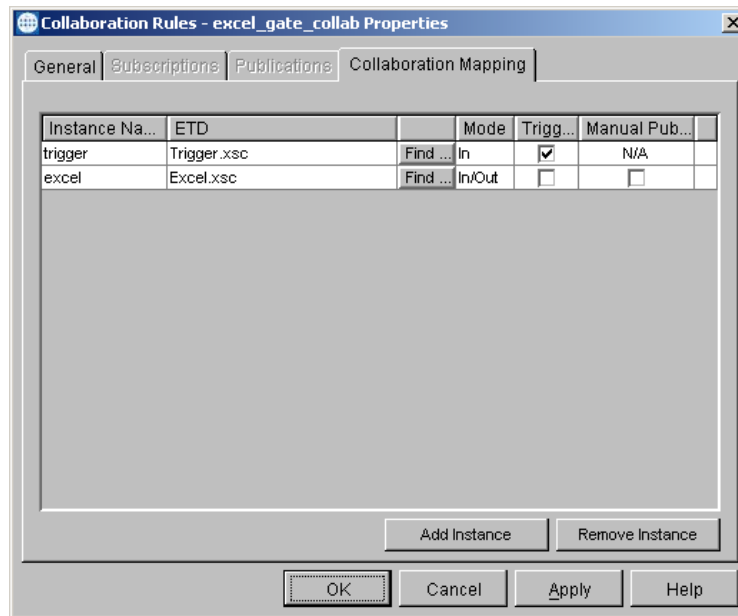
- 9 Click **OK** to close the **Collaboration Rules Properties** dialog box.

## To Create the Java Collaboration Rules

- 1 Select the Navigator's **Components** tab in the e\*Gate Enterprise Manager.
- 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 case, "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.
- 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



- 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.

### 5.8.1. 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:

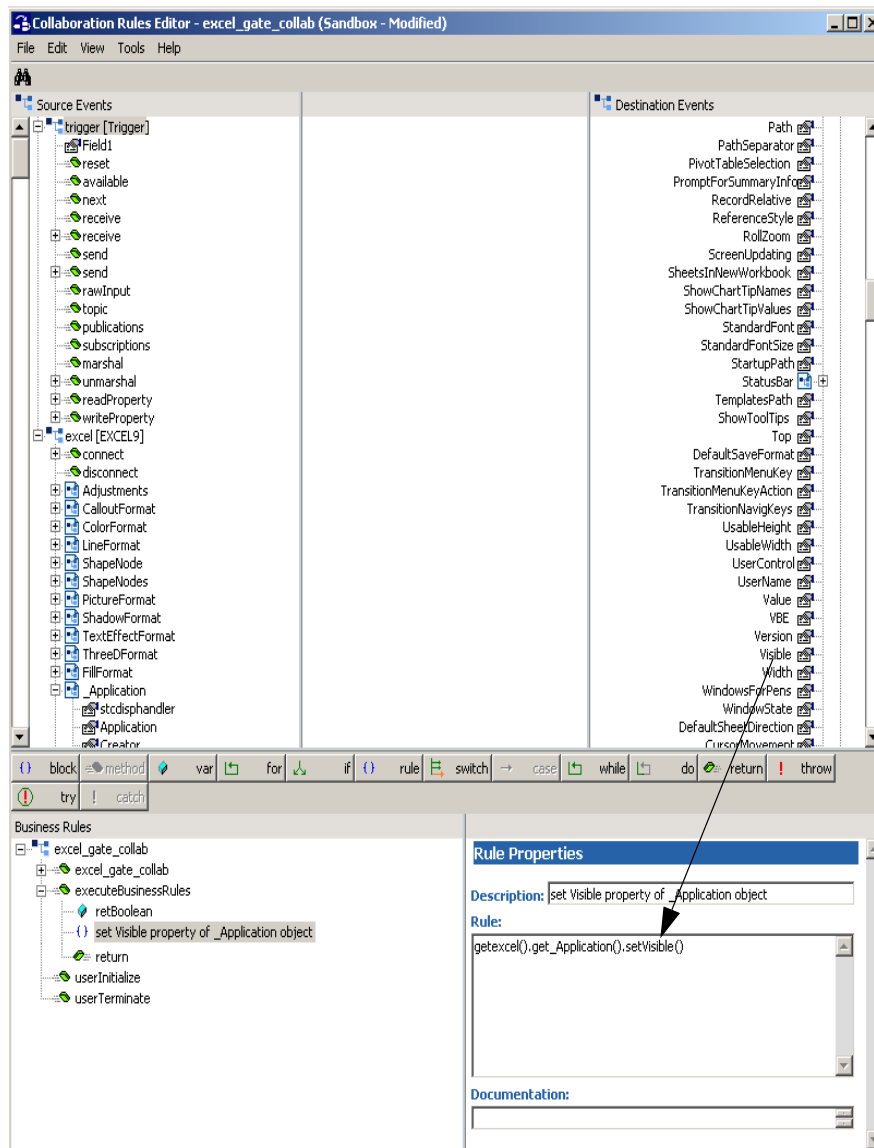
- 1 To open the Collaboration Rules Editor to the **excel\_gate\_collab** Collaboration Rules, click **New** or **Edit** under the **Collaboration Rules** field in the Properties dialog box (see above section.) The Collaboration Rules Editor opens. Expand the

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.
- 3 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 cursor in the last set of parentheses and enter **true** as the value. This creates the following code:

```
getexcel().get_Application().setVisible(true)
```

**Figure 10** Collaboration Rules Editor



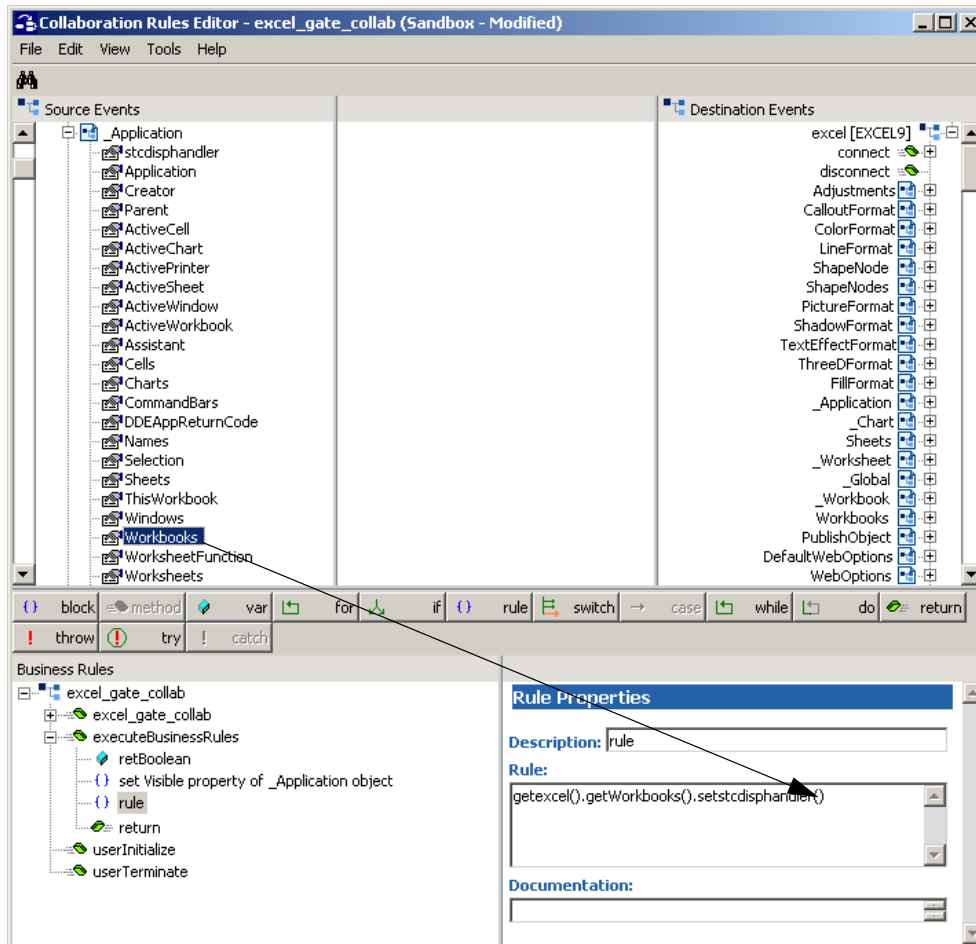
- 4 Type “**set Visible** property of **\_Application** object” in the Rule Properties, Description field. This replaces “rule” as the expression for this rule in the Business Rules pane.

- To create the **set 'Workbooks' handle** rule, drag **stddisphandler** under excel, 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().setstdisphandler(getexcel().get_Application().getWorkbooks())
```

**stdisphandler** is the node where dispatch handles are stored.

**Figure 11** Collaboration Rules Editor



- 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")
```

- 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 **stdisphandler**, 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().setstdisphandler(getexcel().getWorkbooks().getOpen().getretval())
```

- 21 The **set Worksheets handle** rule is created by dragging **stdisphandler**, 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().setstdisphandler(getexcel().get_Workbook().getWorksheets())
```

- 22 To create the **set variant parameter of property 'Item' to 1** rule, drag **short**, under excel, Sheets, propget\_Item, Index, on the Destination Events command node, into the Rule Properties, Rule field. Edit **setshort()** in the Rule Properties, Rule field, to read **setint(1)**. This creates 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** Collaboration Rules - Complete

```
Business Rules
└─ excel_gate_collab : public class excel_gate_collab extends excel_gate_collabBase implements JCollaboratorExt
  └─ excel_gate_collab : public excel_gate_collab()
    └─ () rule : super();
  └─ executeBusinessRules : public boolean executeBusinessRules() throws Exception
    └─ retBoolean : boolean retBoolean = true;
    └─ () set Visible property of _Application object : getexcel().get_Application().setVisible(true);
    └─ () set 'Workbooks' handle : getexcel().getWorkbooks().setstdisphandler(getexcel().get_Application().getWorkbooks());
    └─ () set Filename argument for 'Open' : getexcel().getWorkbooks().getOpen().setFilename("c:\\temp\\test.xls");
    └─ () set UpdateLinks for 'Open' : getexcel().getWorkbooks().getOpen().getUpdateLinks().seterror(-2147352572);
    └─ () set ReadOnly for 'Open' : getexcel().getWorkbooks().getOpen().getReadOnly().seterror(-2147352572);
    └─ () set Format for 'Open' : getexcel().getWorkbooks().getOpen().getFormat().seterror(-2147352572);
    └─ () set Password for 'Open' : getexcel().getWorkbooks().getOpen().getPassword().seterror(-2147352572);
    └─ () 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);
    └─ () set Delimiter for 'Open' : getexcel().getWorkbooks().getOpen().getDelimiter().seterror(-2147352572);
    └─ () set Editable for 'Open' : getexcel().getWorkbooks().getOpen().getEditable().seterror(-2147352572);
    └─ () set Notify for 'Open' : getexcel().getWorkbooks().getOpen().getNotify().seterror(-2147352572);
    └─ () set Converter for 'Open' : getexcel().getWorkbooks().getOpen().getConverter().seterror(-2147352572);
    └─ () set AddToMru for 'Open' : getexcel().getWorkbooks().getOpen().getAddToMru().seterror(-2147352572);
    └─ () invoke 'Open' to get _Workbook : getexcel().getWorkbooks().getOpen().invoke();
    └─ () set _Workbook handle : getexcel().get_Workbook().setstdisphandler(getexcel().getWorkbooks().getOpen().getretval());
    └─ () set Worksheets handle : getexcel().getSheets().setstdisphandler(getexcel().get_Workbook().getWorksheets());
    └─ () set variant parameter of property 'Item' to 1 : getexcel().getSheets().getpropget_Item().getIndex().setint(1);
    └─ () invoke PROPERTYGET of 'Item' : getexcel().getSheets().getpropget_Item().invoke();
    └─ () set _Worksheet handle : getexcel().get_Worksheet().setstdisphandler(getexcel().getSheets().getpropget_Item().getretval());
    └─ () set 'Cell1' of Range : getexcel().get_Worksheet().getpropget_Range().getCell1().setstring("A1");
    └─ () set 'Cell2' of Range : getexcel().get_Worksheet().getpropget_Range().getCell2().setstring("A1");
    └─ () invoke 'Range' : getexcel().get_Worksheet().getpropget_Range().invoke();
    └─ () set Range handle : getexcel().getRange().setstdisphandler(getexcel().get_Worksheet().getpropget_Range().getretval());
    └─ () 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);
    └─ return : return retBoolean;
  └─ userInitialize : public void userInitialize()
  └─ userTerminate : public void userTerminate()
```

## 5.9 Creating Collaborations

Collaborations are the components that receive and process Event Types, and then forward the output to other e\*Gate components or an external. Collaborations consist of the Subscriber, which “listens” for Events of a known type (sometimes from a given

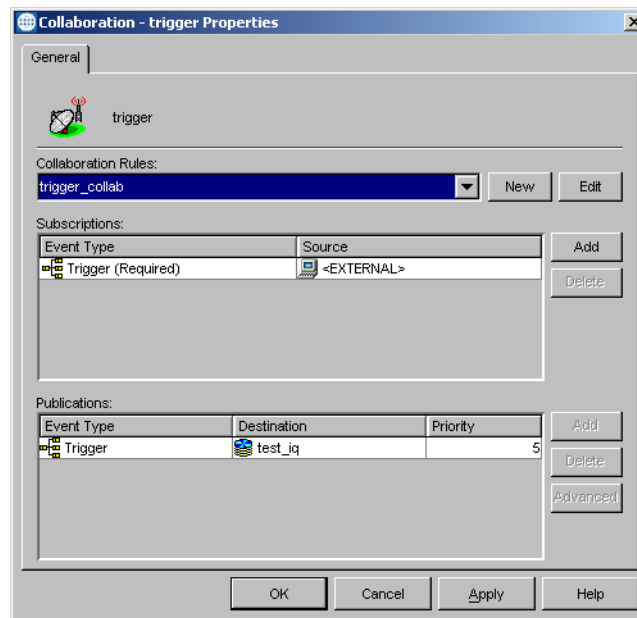


source), and the Publisher, which distributes the transformed Event to a specified recipient.

## To Create the trigger e\*Way Collaboration

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

**Figure 13** Collaboration Properties - trigger



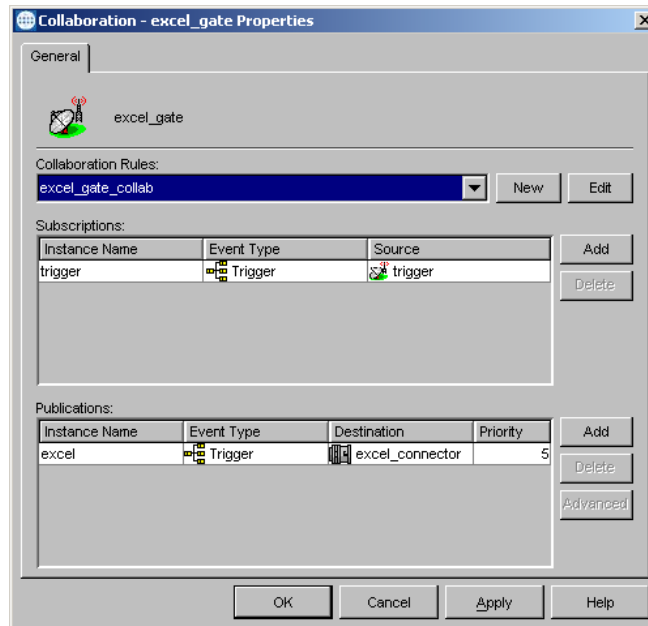
- 8 From the Collaboration Rules list, select the Collaboration Rules file that you created previously (for the sample, “**trigger\_collab**”).
- 9 In the **Subscriptions** area, click **Add** to define the input Event Types to which this Collaboration subscribes.
  - A From the **Event Type** list, select the **Event Type** that you previously defined “**Trigger**”.
  - B Select the **Source** from the **Source** list. In this case, it should be **<External>**.
- 10 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 **“Trigger.”**
  - B Select the publication destination from the **Destination** list. In this case, it should be **“test\_iq”**.
  - C The Priority field defaults to **5**.
- 11 Click **OK** to close.

## To Create the COM\_Multi\_Mode Collaboration

- 1 In the e\*Gate Enterprise Manager, select the Navigators **Components** tab.
- 2 Open the host on which you want to create the Collaboration.
- 3 Select a **Control Broker**.
- 4 Select the e\*Way to assign the Collaboration (for this case, **“excel\_gate”**).
- 5 On the palette, click the **Create a New Collaboration** button.
- 6 Enter the name of the new Collaboration, then click **OK**. (For the sample, **“excel\_gate”**)
- 7 Double-click the new Collaboration to edit its properties.
- 8 From the Collaboration Rules list, select the Collaboration Rules file that you created previously (for the sample, **“excel\_gate\_collab”**).
- 9 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 **“trigger”**.
  - B From the **Event Type** list, select the **Event Type** that you previously defined **“Trigger”**.
  - C Select the **Source** from the **Source** list (in this case, it should be **“trigger”**).
- 10 In the **Publications** area, click **Add** to define the output **Event Types** that this Collaboration publishes (see Figure 13).
  - A From the **Instance Name** list, select the **Instance Name** that you previously defined **“excel”**.
  - B From the **Event Types** list, select the **Event Type** that you previously defined **“Trigger”**.
  - C Select the publication destination from the **Destination** list (in this case, **“excel\_connector”**).
  - D The Priority field defaults to **5**.

**Figure 14** Collaboration Properties

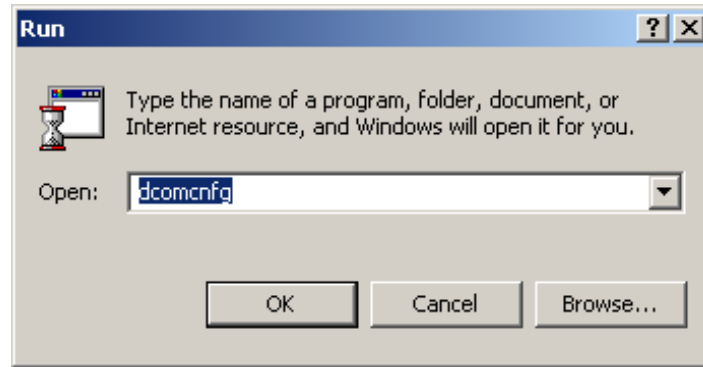


11 Click **OK** to close.

## 5.10 Configuring the DCOM Server

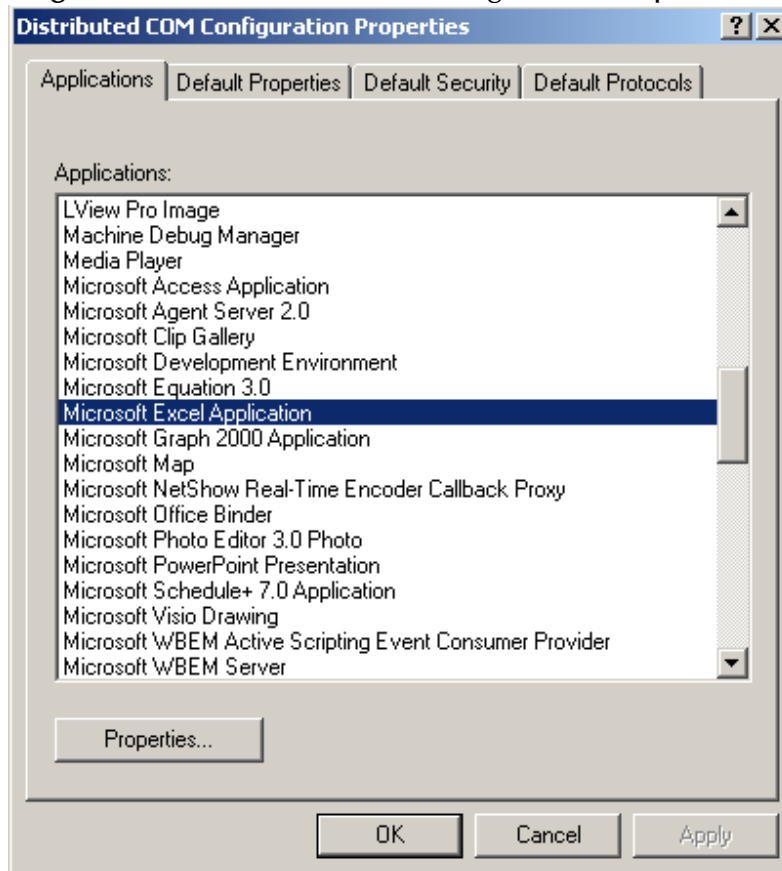
- 1 To configure the DCOM Server, select the Windows **Run** option. When the Run dialog box opens, type `dcomcnfg` in the **Open** field, and click **OK** (see Figure 15).

Figure 15 Run Dialog



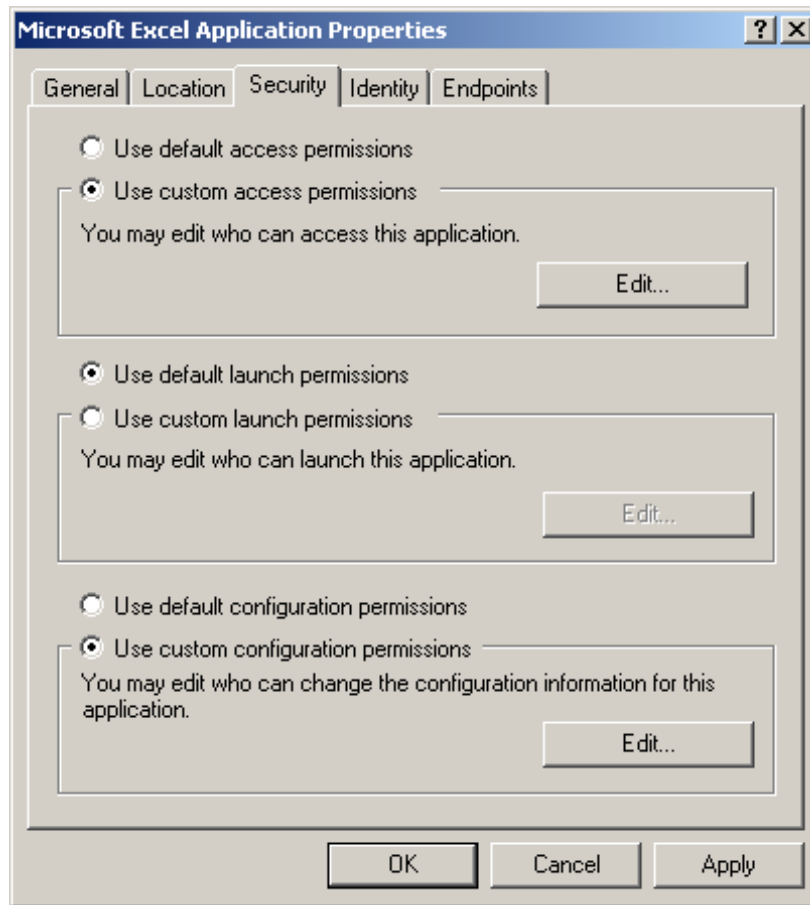
- 2 The Distributed COM Configuration Properties lists the registered applications. Select the **Microsoft Excel** application (see Figure 16). Click **Properties**.

Figure 16 Distributed COM Configuration Properties



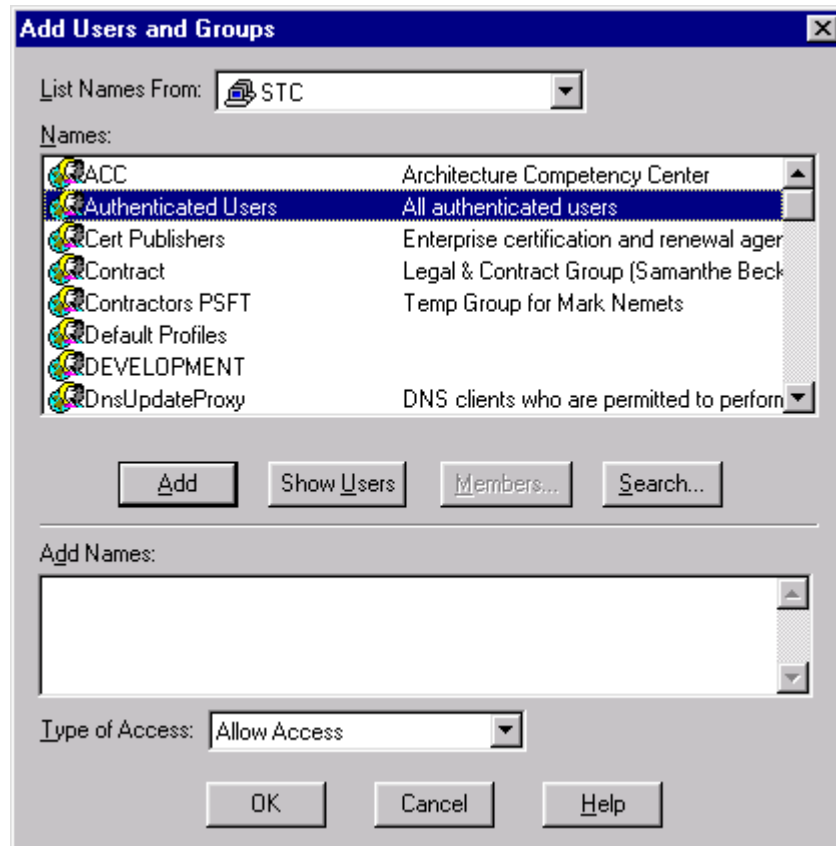
- 3 The Microsoft Excel Application Properties dialog box appears. Select the **Security** tab (see Figure 17).

**Figure 17** Application Properties - Security Tab



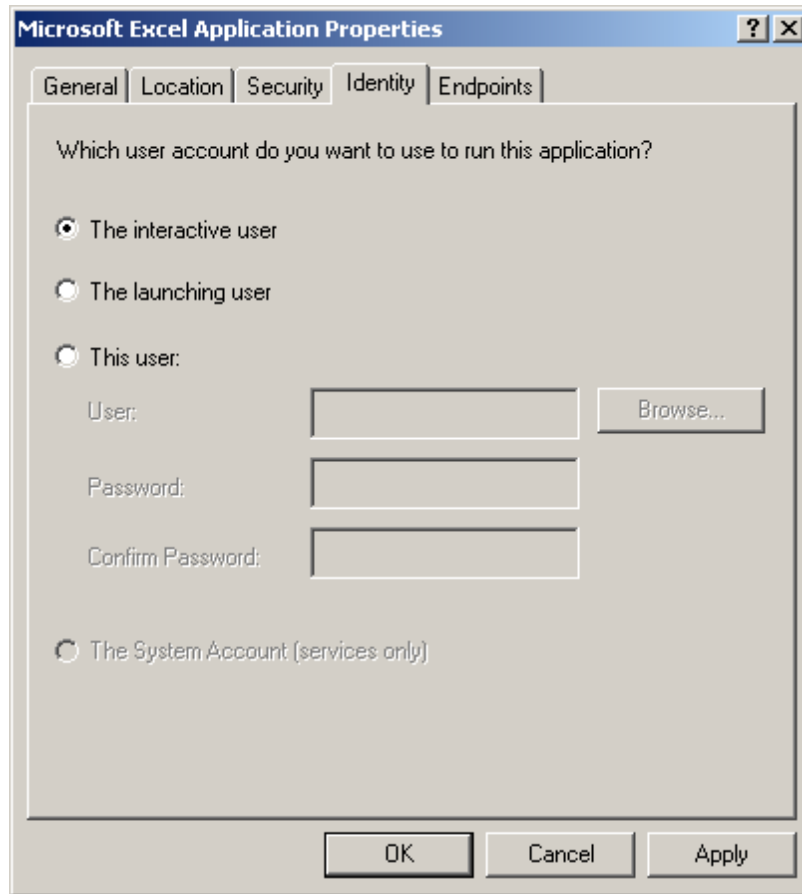
- 4 Select the **Use custom access permission** option button and click **Edit**. The Add Users and Groups dialog box appears (see Figure 18).

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).

**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.

## 5.11 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 Types, 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, and forwards data to the *ewMultiSME* Multi-Mode e\*Way.

### 5.11.1. Completing the Configuration of the COMtest 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 18). For DCOM the Host name also needs to be entered (see "Hostname" on page 19).
- 2 Create two folders on the C: drive named DATA and INDATA.

### 5.11.2. 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 e\*Gate Monitor. 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 e\*Gate Monitor, 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.

---

## 5.12 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 see [The STCComException Class](#) on page 76.

# Java Methods

A number of Java methods have been added to make it easier to set information in the e\*Way ETD Editor and to get information from it. These methods are contained in classes:

- [The COMETDVariant Class](#) on page 50
- [The COMETDWrapper Class](#) on page 60
- [The STCComException Class](#) on page 76

---

## 6.1 The COMETDVariant Class

```
interface com.stc.eways.com.COMETDVariant
public class COMETDVariant
Java.lang.Object
|
+ -- com.stc.eways.com.COMETDVariant
```

The COMETDVariant Class allows passing variant information in the Collaboration via an opaque object wrapper. It is similar to the COM Variant and includes methods for getting and setting the “contained” data. The COMETDVariant Class extends java.lang.Object.

### Methods of the COMETDVariant Class

These methods are described in detail on the following pages:

[COMETDVariant](#) on page 51

[getType](#) on page 52

[getValue](#) on page 52

[setbool](#) on page 53

[getbool](#) on page 53

[setshort](#) on page 53

[getshort](#) on page 54

[setbyte](#) on page 54

[seterror](#) on page 56

[geterror](#) on page 56

[setstring](#) on page 57

[getstring](#) on page 57

[setfloat](#) on page 57

[getfloat](#) on page 58

[setdouble](#) on page 58

[getdouble](#) on page 59

[getbyte](#) on page 55

[setint](#) on page 55

[getint](#) on page 55

[getarray](#) on page 59

[setarray](#) on page 59

---

## COMETDVariant

### Description

Constructor. Constructs the COMETDVariant.

### Syntax

```
public COMETDVariant()  
public COMETDVariant(java.lang.Object obj)  
public COMETDVariant(boolean b)  
public COMETDVariant(java.lang.String s)  
public COMETDVariant(int i)  
public COMETDVariant(byte b)  
public COMETDVariant(short s)  
public COMETDVariant(float f)  
public COMETDVariant(double d)
```

### Parameters

| Name | Type             | Description  |
|------|------------------|--|
| obj  | java.lang.Object | An object of Boolean, integer, short, byte, double, or string. |
| b    | Boolean          | Java Boolean type value.                                       |
| s    | java.lang.String | Java string type value.  |
| i    | int              | Java integer type value.                                       |
| b    | byte             | Java byte type value.  |
| s    | short            | Java short type value.   |
| f    | float            | Java float type value.   |
| d    | double           | Java double type value.  |

### Return Values

None.

### Throws

None.

---

## getType

### Description

Returns the type of the variant as an integer.

### Syntax

```
public int getType()
```

### Parameters

None.

### Return Values

#### int

Returns the type of the variant as follows:

- 0: COMETD\_EVARIANT\_UNKNOWN
- 1: ISBOOLVALUE
- 2: ISSHORTVALUE
- 3: ISINTVALUE
- 4: ISSTRINGVALUE
- 5: ISFLOATVALUE
- 6: ISDOUBLEVALUE
- 7: ISVARIANTVALUE
- 8: ISERRORVALUE
- 9: ISBYTEVALUE

### Throws

None.

---

## getValue

### Description

Returns a **java.lang.Object** as the value of the variant.

### Syntax

```
public java.lang.Object getValue()
```

### Parameters

None.

### Return Values

**java.lang.Object**

### Throws

None.

---

## setbool

### Description

Sets the variant's value to the passed Boolean value and sets the type to Boolean.

### Syntax

```
public void setbool(boolean b)
```

### Parameters

| Name | Type    | Description         |
|------|---------|---------------------|
| b    | Boolean | Java Boolean value. |

### Return Values

None.

### Throws

None.

---

## getbool

### Description

Returns the Boolean value of the variant.

### Syntax

```
public boolean getbool()
```

### Parameters

None.

### Return Values

**Boolean**

### Throws

None.

---

## setshort

### Description

Sets the variant's value to the passed short and sets the type to short.

### Syntax

```
public void setshort(short s)
```

### Parameters

| Name | Type  | Description       |
|------|-------|-------------------|
| s    | short | Java short value. |

### Return Values

None.

### Throws

None.

---

## getshort

### Description

Returns the short value of the variant.

### Syntax

```
public short getshort()
```

### Parameters

None.

### Return Values

**short**

### Throws

None.

---

## setbyte

### Description

Sets the variant's value to the passed byte and sets the type to byte.

### Syntax

```
public void setbyte(byte b)
```

### Parameters

| Name | Type | Description      |
|------|------|------------------|
| b    | byte | Java byte value. |

### Return Values

None.

### Throws

None.

---

## getbyte

### Description

Returns the byte value of the variant.

### Syntax

```
public byte getbyte()
```

### Parameters

None.

### Return Values

**byte**

### Throws

None.

---

## setint

### Description

Sets the variant's value to the passed integer and sets the type to int.

### Syntax

```
public void setint(int i)
```

### Parameters

| Name | Type | Description         |
|------|------|---------------------|
| i    | int  | Java integer value. |

### Return Values

None.

### Throws

None.

---

## getint

### Description

Returns the integer value of the variant.

### Syntax

```
public int getint()
```

### Parameters

None.

## Return Values

**int**

## Throws

None.

---

## seterror

### Description

Sets the variant's value to the passed error value and sets the type to integer.

### Syntax

```
public void seterror(int i)
```

### Parameters

| Name | Type | Description                 |
|------|------|-----------------------------|
| i    | int  | Java (error) integer value. |

## Return Values

None.

## Throws

None.

---

## geterror

### Description

Returns the error integer value of the variant.

### Syntax

```
public int geterror()
```

### Parameters

None.

## Return Values

**int**

## Throws

None.



---

## setstring

### Description

Sets the variant's value to the passed string and sets the type to java.lang.String.

### Syntax

```
public void setstring(java.lang.String s)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| s    | java.lang.String | java.lang.String value. |

### Return Values

None.

### Throws

None.

---

## getstring

### Description

Returns the string value of the variant.

### Syntax

```
public java.lang.String getstring()
```

### Parameters

None.

### Return Values

**java.lang.String**

### Throws

None.

---

## setfloat

### Description

Sets the variant's value to the passed float value and sets the type to float.

### Syntax

```
public void setfloat(float f)
```

### Parameters

| Name | Type  | Description       |
|------|-------|-------------------|
| f    | float | Java float value. |

### Return Values

None.

### Throws

None.

---

## getfloat

### Description

Returns the float value of the variant.

### Syntax

```
public float getfloat()
```

### Parameters

None.

### Return Values

**float**

### Throws

None.

---

## setdouble

### Description

Sets the variant's value to the passed double value and sets the type to double.

### Syntax

```
public void setdouble(double d)
```

### Parameters

| Name | Type   | Description        |
|------|--------|--------------------|
| d    | double | Java double value. |

### Return Values

None.

### Throws

None.

---

## getdouble

### Description

Returns the double value of the variant.

### Syntax

```
public double getdouble()
```

### Parameters

None.

### Return Values

**double**

### Throws

None.

---

## getarray

### Description

Returns the specified item on the array.

### Syntax

```
public COMETDVariant getarray(int index)
```

### Parameters

| Name  | Type | Description     |
|-------|------|-----------------|
| index | int  | Java int value. |

### Return Values

**COMETDVariant**

### Throws

None.

---

## setarray

### Description

Set the specified item on the array to passed byte, Boolean, short, int, java.lang.String, float, or double value.

### Syntax

```
public void setarray(int index, byte b)  
public void setarray(int index, boolean b)
```

```
public void setarray(int index, short s)
public void setarray(int index, int i)
public void setarray(int index, java.lang.String s)
public void setarray(int index, float f)
public void setarray(int index, double d)
```

### Parameters

| Name  | Type             | Description         |
|-------|------------------|---------------------|
| index | int              | Java integer value. |
| b     | byte             | Java byte value.    |
| b     | Boolean          | Java Boolean value. |
| s     | short            | Java short value.   |
| i     | int              | Java integer value. |
| s     | java.lang.String | Java string value.  |
| f     | float            | Java float value.   |
| d     | double           | Java double value.  |

### Return Values

None.

### Throws

None.

---

## 6.2 The COMETDWrapper Class

```
class com.stc.eways.com.COMETDWrapper
public class COMETDWrapper
```

```
Java.lang.Object
```

```
    |
    + -- com.stc.eways.com.COMETDWrapper
```

The COMETDWrapper Class extends java.lang.Object

### Methods of the COMETDWrapper Class

These methods are described in detail on the following pages:

- [COMETDWrapper](#) on page 61
- [COMETDWrapper\\_Init](#) on page 61
- [setParam](#) on page 62
- [getFloatArray](#) on page 71
- [getFloat2DArray](#) on page 72
- [getDoubleArray](#) on page 72

- [getBytesArray](#) on page 67
- [getBytes2DArray](#) on page 68
- [getBooleanArray](#) on page 68
- [getBoolean2DArray](#) on page 69
- [getShortArray](#) on page 69
- [getShort2DArray](#) on page 70
- [getIntArray](#) on page 70
- [getInt2DArray](#) on page 71
- [getDouble2DArray](#) on page 73
- [getStringArray](#) on page 73
- [getString2DArray](#) on page 74
- [getVariantParam](#) on page 74
- [getRetVal](#) on page 75
- [setdisphandler](#) on page 75
- [invoke](#) on page 75

## COMETDWrapper

### Description

Constructor. Constructs the COMETDWrapper.

### Syntax

```
public COMETDWrapper()
```

### Parameters

None.

### Return Values

None.

### Throws

None.

## COMETDWrapper\_Init

### Description

Initializes the COMETD\_Wrapper instance, sets the default STCDispatchDriver instance, sets the invoke type, sets the parameter matrix and sets the method name.

### Syntax

```
public void COMETDWrapper_Init(com.stc.eways.com.STCDispatchDriver
                                driver, int invoke_type,
                                java.lang.String func_name,
                                java.util.ArrayList param_name,
                                java.util.ArrayList param_type,
                                java.util.ArrayList param_direction,
                                java.lang.String ret_type)
```

### Parameters

| Name   | Type                                | Description                              |
|--------|-------------------------------------|--|
| driver | com.stc.eways.com.STCDispatchDriver | An instance of STCDispatchDriver object. |

| Name            | Type                | Description                                  |
|-----------------|---------------------|--|
| invoke_type     | int                 | int  |
| func_name       | java.lang.String    | String function name.                        |
| param_name      | java.util.ArrayList | An array of strings for parameter names.     |
| param_type      | java.util.ArrayList | An array of strings for parameter type.      |
| param_direction | java.util.ArrayList | An array of strings for parameter direction. |
| ret_type        | java.lang.String    | String value for the returned value type.    |

### Return Values

None.

### Throws

None.

## setParam

### Description

Sets the parameter's values.

### Syntax

```

public void setParam(java.lang.String name,
                    com.stc.eways.com.COMETDVariant value)

public void setParam(java.lang.String name, int value)

public void setParam(java.lang.String name, int index, int value)

public void setParam(java.lang.String name, int[] value)

public void setParam(java.lang.String name, int[][] value)

public void setParam(java.lang.String name, java.lang.String value)

public void setParam(java.lang.String name, int index,
                    java.lang.String value)

public void setParam(java.lang.String name, java.lang.String[] value)

public void setParam(java.lang.String name, java.lang.String[][]
                    value)

public void setParam(java.lang.String name, boolean value)

public void setParam(java.lang.String name, int index, boolean value)

public void setParam(java.lang.String name, boolean[] value)

public void setParam(java.lang.String name, boolean[][] value)

```

```

public void setParam(java.lang.String name, byte value)
public void setParam(java.lang.String name, int index, byte value)
public void setParam(java.lang.String name, byte[] value)
public void setParam(java.lang.String name, byte[][] value)
public void setParam(java.lang.String name, short value)
public void setParam(java.lang.String name, int index, short value)
public void setParam(java.lang.String name, short[] value)
public void setParam(java.lang.String name, short[][] value)
public void setParam(java.lang.String name, float value)
public void setParam(java.lang.String name, int index, float value)
public void setParam(java.lang.String name, float[] value)
public void setParam(java.lang.String name, float[][] value)
public void setParam(java.lang.String name, double value)
public void setParam(java.lang.String name, int index, double value)
public void setParam(java.lang.String name, double[] value)
public void setParam(java.lang.String name, double[][] value)
public void setParam(java.lang.String name, java.lang.Object value)
public void setParam(java.lang.String name, int index,
                    java.lang.Object value)

```

### Parameters

| Name  | Type                            | Description                              |
|-------|---------------------------------|--|
| name  | java.lang.String                | String name.                             |
| value | com.stc.eways.com.COMETDVariant | An instance of the COMETDVariant object. |

| Name  | Type             | Description    |
|-------|------------------|----------------|
| name  | java.lang.String | String name.   |
| value | int              | Java int value |

| Name  | Type             | Description    |
|-------|------------------|----------------|
| name  | java.lang.String | String name.   |
| index | int              | Int            |
| value | int              | Java int value |

| Name  | Type             | Description                           |
|-------|------------------|---------------------------------------|
| name  | java.lang.String | String name.                          |
| value | int[]            | One-dimensional array of the integer. |

| Name  | Type             | Description                           |
|-------|------------------|---------------------------------------|
| name  | java.lang.String | String name.                          |
| value | int[][]          | Two-dimensional array of the integer. |

| Name  | Type             | Description                  |
|-------|------------------|------------------------------|
| name  | java.lang.String | String name.                 |
| value | java.lang.String | Java java.lang.String value. |

| Name  | Type             | Description                  |
|-------|------------------|------------------------------|
| name  | java.lang.String | String name.                 |
| index | int              | Int                          |
| value | java.lang.String | Java java.lang.String value. |

| Name  | Type               | Description                                    |
|-------|--------------------|--|
| name  | java.lang.String   | String name.                                   |
| value | java.lang.String[] | One-dimensional array of the java.lang.String. |

| Name  | Type                 | Description                                    |
|-------|----------------------|--|
| name  | java.lang.String     | String name.                                   |
| value | java.lang.String[][] | Two-dimensional array of the java.lang.String. |

| Name  | Type             | Description         |
|-------|------------------|---------------------|
| name  | java.lang.String | String name.        |
| value | Boolean          | Java Boolean value. |

| Name  | Type             | Description  |
|-------|------------------|--------------|
| name  | java.lang.String | String name. |
| index | int              | Int          |



| Name  | Type    | Description         |
|-------|---------|---------------------|
| value | Boolean | Java Boolean value. |

| Name  | Type             | Description                           |
|-------|------------------|---------------------------------------|
| name  | java.lang.String | String name.                          |
| value | Boolean[]        | One-dimensional array of the Boolean. |

| Name  | Type             | Description                           |
|-------|------------------|---------------------------------------|
| name  | java.lang.String | String name.                          |
| value | Boolean[][]      | Two-dimensional array of the Boolean. |

| Name  | Type             | Description      |
|-------|------------------|------------------|
| name  | java.lang.String | String name.     |
| value | byte             | Java byte value. |

| Name  | Type             | Description      |
|-------|------------------|------------------|
| name  | java.lang.String | String name.     |
| value | int              | Java int value   |
| value | byte             | Java byte value. |

| Name  | Type             | Description                        |
|-------|------------------|------------------------------------|
| name  | java.lang.String | String name.                       |
| value | byte[]           | One-dimensional array of the byte. |

| Name  | Type             | Description                        |
|-------|------------------|------------------------------------|
| name  | java.lang.String | String name.                       |
| value | byte[][]         | Two-dimensional array of the byte. |

| Name  | Type             | Description       |
|-------|------------------|-------------------|
| name  | java.lang.String | String name.      |
| value | short            | Java short value. |

| Name  | Type             | Description       |
|-------|------------------|-------------------|
| name  | java.lang.String | String name.      |
| value | int              | Java int value    |
| value | short            | Java short value. |

| Name  | Type             | Description                         |
|-------|------------------|-------------------------------------|
| name  | java.lang.String | String name.                        |
| value | short[]          | One-dimensional array of the short. |

| Name  | Type             | Description                         |
|-------|------------------|-------------------------------------|
| name  | java.lang.String | String name.                        |
| value | short[][]        | Two-dimensional array of the short. |

| Name  | Type             | Description       |
|-------|------------------|-------------------|
| name  | java.lang.String | String name.      |
| value | float            | Java float value. |

| Name  | Type             | Description       |
|-------|------------------|-------------------|
| name  | java.lang.String | String name.      |
| value | int              | Java int value    |
| value | float            | Java float value. |

| Name  | Type             | Description                         |
|-------|------------------|-------------------------------------|
| name  | java.lang.String | String name.                        |
| value | float[]          | One-dimensional array of the float. |

| Name  | Type             | Description                         |
|-------|------------------|-------------------------------------|
| name  | java.lang.String | String name.                        |
| value | float[][]        | Two-dimensional array of the float. |

| Name  | Type             | Description        |
|-------|------------------|--------------------|
| name  | java.lang.String | String name.       |
| value | double           | Java double value. |

| Name  | Type             | Description        |
|-------|------------------|--------------------|
| name  | java.lang.String | String name.       |
| value | int              | Java int value     |
| value | double           | Java double value. |

| Name  | Type             | Description                          |
|-------|------------------|--------------------------------------|
| name  | java.lang.String | String name.                         |
| value | double[]         | One-dimensional array of the double. |

| Name  | Type             | Description                          |
|-------|------------------|--------------------------------------|
| name  | java.lang.String | String name.                         |
| value | double[][]       | Two-dimensional array of the double. |

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| name  | java.lang.String | String name.            |
| value | java.lang.Object | java.lang.object value. |

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| name  | java.lang.String | String name             |
| index | int              | Int                     |
| value | java.lang.Object | java.lang.object value. |

#### Return Values

None.

#### Throws

None.

---

## getBytesArray

### Description

If the passed object is of the ArrayList type, the method converts the array to a byte array. It returns a byte if an index is specified.

### Syntax

```
public byte[] getBytesArray(java.lang.Object obj)
```

```
public byte getByteArray(int index, java.lang.Object obj)
```

#### Parameters

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| obj   | java.lang.Object | java.lang.Object value. |
| index | int              | Java integer value.     |

#### Return Values

byte[]  
byte

#### Throws

None.

## getBytes2DArray

#### Description

If the object contains a two-dimensional ArrayList the method converts the array to a byte[ ][ ].

#### Syntax

```
public byte[][] getBytes2DArray(java.lang.Object obj)
```

#### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

#### Return Values

byte[ ][ ]

#### Throws

None.

## getBooleanArray

#### Description

If the passed object is of the ArrayList type, the method converts the array to a Boolean array. It returns a Boolean if an index is specified.

#### Syntax

```
public boolean[] getBooleanArray(java.lang.Object obj)
```

```
public boolean getBooleanArray(int index, java.lang.Object obj)
```

### Parameters

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| index | int              | Java integer value.     |
| obj   | java.lang.Object | java.lang.Object value. |

### Return Values

**Boolean**

### Throws

None.

## getBoolean2DArray

### Description

If the object contains a two-dimensional ArrayList the method converts the array to a byte[ ][ ].

### Syntax

```
public boolean[ ][ ] getBoolean2DArray(java.lang.Object obj)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.util.Object | java.lang.Object value. |

### Return Values

**Boolean[ ][ ]**

### Throws

None.

## getShortArray

### Description

If the passed object is of the ArrayList type, the method converts the array to a short array. It returns short if an index is specified.

### Syntax

```
public short[] getShortArray(java.lang.Object obj)
public short getShortArray(int index, java.lang.Object obj)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

| Name  | Type | Description         |
|-------|------|---------------------|
| index | int  | Java integer value. |

#### Return Values

**short**

#### Throws

None.

## getShort2DArray

### Description

If the object contains a two-dimensional ArrayList the method converts the array to a short[ ][ ].

### Syntax

```
public short[][] getShort2DArray(java.lang.Object obj)
```

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

#### Return Values

**short[ ][ ]**

#### Throws

None.

## getIntArray

### Description

If the passed object is of the ArrayList type, the method converts the array to an int array. It returns an int if an index is specified.

### Syntax

```
public int[] getIntArray(java.lang.Object obj)
```

```
public int getIntArray(int index, java.lang.Object obj)
```

### Parameters

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| obj   | java.lang.Object | java.lang.Object value. |
| index | int              | Java integer value.     |

### Return Values

**int[]**  
**int**

Returns an integer for the array if an index is specified.

### Throws

None.

## getInt2DArray

### Description

If the object contains a two-dimensional ArrayList the method converts the array to an int[ ][ ].

### Syntax

```
public int[][] getInt2DArray(java.lang.Object obj)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

### Return Values

**int[][]**

### Throws

None.

## getFloatArray

### Description

If the passed object is of the ArrayList type, the method converts the array to a float array. It returns a float if an index is specified.

### Syntax

```
public float[] getFloatArray(java.lang.Object obj)
public float getFloatArray(int index, java.lang.Object obj)
```

### Parameters

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| obj   | java.lang.Object | java.lang.Object value. |
| index | int              | Java integer value.     |

### Return Values

**float[]**  
**float**

### Throws

None.

---

## getFloat2DArray

### Description

If the object contains a two-dimensional ArrayList the method converts the array to a float[ ][ ].

### Syntax

```
public float[ ][ ] getFloat2DArray(java.lang.Object obj)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

### Return Values

**float[ ][ ]**

### Throws

None.

---

## getDoubleArray

### Description

If the passed object is of the ArrayList type, the method converts the array to a double array. It returns a double if an index is specified.

### Syntax

```
public double[] getDoubleArray(java.lang.Object obj)
```

```
public double getDoubleArray(int index, java.lang.Object obj)
```

### Parameters

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| obj   | java.lang.Object | java.lang.Object value. |
| index | int              | Java integer value.     |



### Return Values

**double[]**  
**double**

### Throws

None.

## getDouble2DArray

### Description

If the object contains a two-dimensional ArrayList the method converts the array to a double[ ][ ].

### Syntax

```
public double[][] getDouble2DArray(java.lang.Object obj)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

### Return Values

**double[][]**

### Throws

None.

## getStringArray

### Description

If the passed object is of the ArrayList type, the method converts the array to a string array. It returns a string if an index is specified.

### Syntax

```
public java.lang.String[] getStringArray(java.lang.Object obj)
public java.lang.String getStringArray(int index,
                                         java.lang.Object obj)
```

### Parameters

| Name  | Type             | Description             |
|-------|------------------|-------------------------|
| obj   | java.lang.Object | java.lang.Object value. |
| index | int              | Java integer value.     |

### Return Values

`java.lang.String[]`

### Throws

None.

---

## getString2DArray

### Description

If the object contains a two-dimensional ArrayList the method converts the array to a `java.lang.String[ ][ ]`.

### Syntax

```
public java.lang.String[ ][ ] getString2DArray(java.lang.Object obj)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| obj  | java.lang.Object | java.lang.Object value. |

### Return Values

`java.lang.String[ ][ ]`

### Throws

None.

---

## getVariantParam

### Description

When given the field name the method returns a COMETDVariant object.

### Syntax

```
public java.lang.Object getParam(java.lang.String name)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| name | java.lang.String | java.lang.String value. |

### Return Values

`java.lang.Object`  
Represents COMETDVariant.

### Throws

None.

---

## getRetVal

### Description

After invoking the method, this method returns the result of the COM call in the form of a `java.lang.Object`.

### Syntax

```
public java.lang.Object getRetVal()
```

### Parameters

None.

### Return Values

**java.lang.Object**  
Represents `COMETDVariant`.

### Throws

None.

---

## setdisphandler

### Description

Sets the `STCDispatchDriver` instance.

### Syntax

```
public void setdisphandler(java.lang.Object disp)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| disp | java.lang.Object | java.lang.Object value. |

### Return Values

None.

### Throws

None.

---

## invoke

### Description

Once the parameters are prepared these two methods invoke the COM call.

### Syntax

```
public void invoke()  
public void invoke(java.lang.Object disp)
```

### Parameters

| Name | Type             | Description             |
|------|------------------|-------------------------|
| disp | java.lang.Object | java.lang.Object value. |

### Return Values

None.

### Throws

None.

---

## 6.3 The STCComException Class

```

class com.stc.eways.com.STCComException
public class STCComException
java.lang.Object
|
+ -- java.lang.Throwable
|
+ -- java.lang.Exception
|
+ -- java.lang.RuntimeException
|
+ -- com.stc.eways.STCComException

```

The STCComException Class is a part of the low-level JNI layer of the COM/DCOM e\*Way. If one of the methods (invoke, getProp, or setProp) fails on the STCDispatchDriver class, an exception of this type is thrown. Exceptions is not caught by the builder-generated code. It is up to the user to catch an exception if desired.

This operation provides a mechanism for the Collaboration code to detect when and why a failure occurs at the lower level. Access to the COM HRESULT is provided, as well as a description of the HRESULT from the operating system if available. The STCComException Class extends **java.lang.RuntimeException**.

### Methods of the STCComException Class

These methods are described in detail on the following pages:

- [STCComException](#) on page 77
- [getErrorMessage](#) on page 77
- [getErrorCode](#) on page 78
- [getSystemErrorText](#) on page 78

## Fields of the STCComException Class

| Name  | Type                             | Description                             |
|-------|----------------------------------|---|
| m_nHR | protected int m_nHR              | The HRESULT passed to the constructor.  |
| m_sOp | protected java.lang.String m_sOp | The context string passed to the c'tor. |

---

## STCComException

### Description

Constructor. Pass in the HRESULT and a context string. The context string can be null, but must be a string that indicates what is causing the exception to be thrown, for example, invoking foo or getting property bar.

### Syntax

```
public STCComException (int hr, java.lang.String sOp)
```

### Parameters

| Name | Type             | Description         |
|------|------------------|---------------------|
| hr   | int              | The COM error code. |
| sOp  | java.lang.String | The content string. |

### Return Values

None.

### Throws

None.

---

## getMessage

### Description

Overrides the base method. This method must be called for the e\*Way to attempt to get available error text from the OS.

### Syntax

```
public java.lang.String getMessage()
```

### Parameters

None.

### Return Values

**java.lang.String.**

Returns a string formatted like the following: **COM Exception: 0x8004002E - Creating component of progid Some.Class: class not registered.**

**Throws**

None.

**Overrides**

`getMessage` in class `java.lang.Throwable`

---

## **getErrorCode**

**Description**

Provides access to the HRESULT passed to the constructor.

**Syntax**

```
public int getErrorCode()
```

**Parameters**

None.

**Return Values**

**int**

Returns the error code.

**Throws**

None.

---

## **getSystemErrorText**

**Description**

A convenience method exposed to allow the user to get the error text for an HRESULT.

**Syntax**

```
public java.lang.String getSystemErrorText(int hr)
```

**Parameters**

| <b>Name</b> | <b>Type</b> | <b>Description</b>  |
|-------------|-------------|---------------------|
| hr          | int         | HRESULT error code. |

**Return Values**

**java.lang.String**

Returns the error text.

**Throws**

None.

# Index

## C

- Classpath Override 14
- Classpath Prepend 14
- collaboration rules 31
- collaborations 40
  - for the Multi-Mode e\*Way 42
- COM/DCOM
  - runtime exceptions 49
- COMETDVariant Class 50
  - methods
    - COMETDVariant 51
    - getarray 59
    - getbool 53
    - getbyte 55
    - getdouble 59
    - geterror 56
    - getfloat 58
    - getint 55
    - getshort 54
    - getstring 57
    - getType 52
    - getValue 52
    - setarray 59
    - setbool 53
    - setbyte 54
    - setdouble 58
    - seterror 56
    - setfloat 57
    - setint 55
    - setstring 57
- COMETDWrapper Class 60
  - methods
    - COMETDWrapper 61
    - COMETDWrapper\_Init 61
    - getBoolean2DArray 69
    - getBooleanArray 68
    - getByte2DArray 68
    - getByteArray 67
    - getDouble2DArray 73
    - getDoubleArray 72
    - getFloat2DArray 72
    - getFloatArray 71
    - getInt2DArray 71
    - getIntArray 70

- getRetVal 75
- getShort2DArray 70
- getShortArray 69
- getString2DArray 74
- getStringArray 73
- getVariantParam 74
- invoke 75
- setdisphandler 75
- setParam 62

## D

- Disable JIT 15

## E

- e\*Way Connection
  - creating 29
  - parameters
    - SME 18
- e\*Way connection
  - configuration 18
- e\*Ways
  - creating and configuring 27
- error messages 49
- event type
  - creating from an existing .xsc 24
  - creating from the COMWizard 25
- event types 24
- exceptions 49
- external application requirements 9

## F

- files/directories created by installation 11

## I

- implementation 21
- Initial Heap Size 15
- installation 10
  - directories created 11
  - files/directories 11
  - pre-installation 10
  - Windows NT 10
  - Windows NT and Windows 2000 10
- intelligent queues 31
- intended reader 7

## J

- JNI DLL Absolute Pathname 13
- JVM settings 12

## M

- Maximum Heap Size 15
- Multi-Mode e\*Way 12
  - configuration 12
  - configuration parameters 12
    - Auxiliary JVM Configuration File 16
    - CLASSPATH Append From Environment Variable 14
    - CLASSPATH Override 14
    - CLASSPATH Prepend 14
    - Disable JIT 15
    - JNI DLL Absolute Pathname 13
    - Maximum Heap Size 15
    - Maximum Stack Size for JVM Threads 15
    - Maximum Stack Size for Native Threads 15
    - Remote Debugging port number 16
    - Suspend option for debugging 16
  - parameters 12

## O

- operating systems 8
- overview
  - COM/DCOM 7
  - sample implementation 22

## P

- parameters
  - Class 19
  - connector 18
  - host 20
  - Multi-Mode e\*Way
    - CLASSPATH prepend 14
    - Initial Heap Size 15
    - JNI DLL absolute pathname 13
    - JVM settings 12
    - Maximum Heap Size 15
  - progid 19
  - Type 19

## R

- runtime exceptions 49

## S

- sample schema 48
- STCComException Class
  - methods
    - getErrorCode 78
    - getMessage 77
    - getSystemErrorText 78

- system requirements 8
  - external 9
  - Windows 2000 8
  - Windows NT 8

## T

- test.xls
  - creating 23

## U

- ulti 12

## W

- Windows installation 10
- Windows NT 10