*SeeBeyond™ eBusiness Integration Suite*

# e*Way Intelligent Adapter for CICS User's Guide

*Release 4.5.3*

*Java Version*

Version 20020513120521.

# Contents

Chapter 1

# Introduction

This chapter includes a brief description of IBM's Customer Information Control System (CICS), an overview of SeeBeyond™ Technology Corporation's (SeeBeyond™) e*Way Intelligent Adapter for CICS, as well as system requirements for using the e*Way.

## 1.1 Overview

The CICS e*Way is an interface that makes bidirectional calls to CICS transactional programs remotely. CICS is a transaction processor that supports a real-time distributed processing environment and also supports online transaction processing (OLTP).

IBM provides a CICS Client Gateway that has an API (the External Call Interface or ECI) to call CICS transactions on the mainframe. The ECI allows a non-CICS application program to call a CICS program in a CICS server. SeeBeyond's CICS e*Way uses this ECI method to connect to CICS.

The CICS e*Way includes a build tool, the Cobol Copybook Converter. This feature takes as input a Cobol Copybook file and creates e*Gate Event Type Definitions (ETDs) for use within the Monk environment. These Copybook file structures are passed into the CICS environment as the data buffer (COMMAREA). These ETD files (.ssc) can be converted into **.xsc** files that are compatible with the Java Collaboration Editor.

The CICS e*Way has the following modes of operation:

- Inbound
- Outbound
- Request/reply

For more information, see **"Implementation" on page 21**. This user's guide explains how to install and configure the CICS e*Way.

### 1.1.1 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 expert-level knowledge of Windows operations and administration; to be thoroughly familiar with CICS and with Windows-style GUI operations.

### 1.1.2 Components

The following components comprise the Java-enabled CICS e*Way:

- **stcecics.jar**: Contains the logic required by the e*Way to gain access to CICS.
- **cicsclient.xsc**: Allows the user to create hierarchical Event Type Definitions manually to be used in conjunction with the parsing engine contained within the extended Java Collaboration Service.
- **e*Way Connection**: The CICS e*Way Connections provide the access to the information necessary for connection to a specified external connection.

A complete list of installed files appears in **Table 1 on page 13**.

## 1.2 Supported Operating System

The CICS e*Way is available on the following operating systems:

- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Windows NT 4.0 SP6a
- Solaris 7, and 8
- AIX 4.3.3
- HP-UX 11.0 and HP-UX 11i (Java only)
- OS/390 V2R10 (Java only)
- Japanese Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Japanese Windows NT 4.0 SP6a
- Japanese Solaris 7, and 8
- Japanese HP-UX 11.0 (Java only)
- Korean HP-UX 11.0 (Java only)

***Note:*** *For OS/390 V2R10 see* **OS/390 Configuration Requirements for the CICS Server** *on page 9*

***Note:*** *When using Solaris 7, the LC_ALL environment variable must be set to either en_GB (Great Britain) or en_US (United States) for the CP500 code page (Java 1.3.1). Set the LC_ALL environmental variable as follows: export LC_ALL=en_GB or export LC_ALL=en_US.*

***Note:*** *When using HP-UX with CICS Transaction Gateway 4.0 append the following path to the SHLIB PATH: SHLIB_PATH=$(SHLIB_PATH}:<e*Gate>/client/ThirdParty/IBMctg/lib/Hpux32. ("e*Gate" denotes where e*Gate has been installed.)*

# 1.3 System Requirements

To use the CICS e*Way, you need the following:

- An e*Gate Participating Host, version 4.5.1 or later.
- A TCP/IP network connection.
- CICS Transaction Gateway version 4.0 or greater, with APAR PQ57730 applied.
- 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.

## 1.3.1 External System Requirements

### CICS Server Requirements

To enable the e*Way to communicate correctly with CICS, the following are required

- CICS for MVS/ESA V4.1 or CICS Transaction Server v1.2 or greater
- The CICS e*Way runs and has been tested using TCP62 connectivity provided by the CICS Transaction Gateway. The Transaction Gateway supports SNA communications on Windows and AIX platforms, but the CICS e*Way has not been tested using SNA.

### OS/390 Configuration Requirements for the CICS Server

Full details on configuring OS/390 for connection via TCP62 are available in the CICS Transaction Gateway, Client Administration manual for any specified platforms. These details are found in the chapter on “Setting Up Client/Server Communications.”

The summarized requirements are as follows:

- Install any of the VTAM AnyNet® releases.
- Install a TCP major node, which defines the AnyNet interface between TCP/IP and VTAM. For further information about how to do this, see the Guide to SNA over TCP/IP book, SC31-6527.
- Install a CDRSC major node, which defines the remote Client device and instructs VTAM to route any session requests through the TCP/IP Physical Unit (ALSLIST).
- Check that the Physical Unit (PU) for the AnyNet interface is active.
- On CICS, you must define an APPC connection to the client workstation. (The connection can be statically defined, or autoinstalled.)
- Add an entry to the VTAM logon mode (LOGMODE) table for the modename specified on the SESSIONS definition. This entry specifies the class of service required for the group of sessions.

Chapter 2

# Installation

This chapter explains procedures for installing the CICS e*Way.



## 2.1 Windows NT 4.0 and Windows 2000

### 2.1.1 Pre-installation

- Exit 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 CICS e*Way on a Windows system**

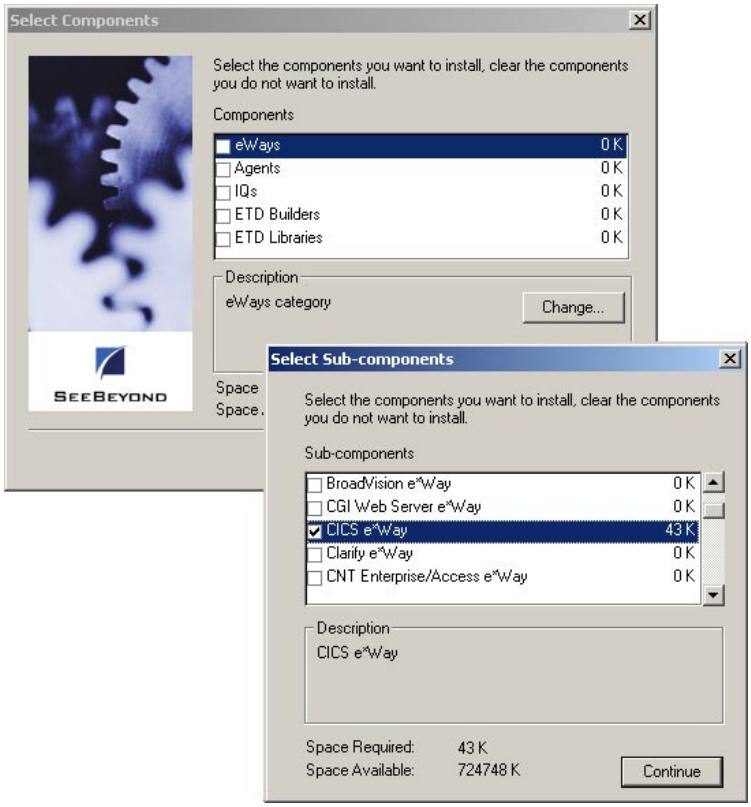
1. Log in as an Administrator to the workstation on which you are installing the e*Way.
2. Insert the e*Way installation CD-ROM into the CD-ROM drive.
3. If the CD-ROM drive's Autorun feature is enabled, the setup application launches automatically; skip ahead to step 4. Otherwise, use the Windows Explorer 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 10 Select the **CICS e*Way** as shown in figure 1. Click the continue button to return to the **Select Components** dialog box, then click **Next**.

**Figure 1** Select Components




11 Follow the rest of the on-screen instructions to install the Java-enabled CICS 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.***

***Note:*** *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.*

*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 UNIX

## 2.2.1 Pre-installation

You do not require root privileges to install this e*Way. Log in under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privileges to create files in the e*Gate directory tree.

## 2.2.2 Installation Procedure

**To install the CICS e*Way on a UNIX system**

1. Log in on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.
2. If necessary, mount the CD-ROM drive.
3. At the shell prompt, type

   **cd /cdrom**
4. Start the installation script by typing

   **setup.sh**
5. A menu of options will appear. Select the **Install e*Way** option. Then, follow the additional on-screen directions.

***Note:*** *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.***

6. After installation is complete, exit the installation utility and launch the Enterprise Manager.

***Important:*** *For HP-UX systems with CICS Transaction Gateway 4.0 append the following path to the SHLIB PATH:* SHLIB_PATH=$(SHLIB_PATH}:<e*Gate>/client/ThirdParty/IBMctg/lib/Hpux32. (*"e*Gate" is where e*Gate has been installed)

***Note:*** *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 system.*

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

## 2.3 OS/390 V2R10

e*Way installation notes for OS/390 V2R10 can be found in the **e*Gate Integrator Installation Guide**.

## 2.4 Files/Directories Created by the Installation

The Java-enabled CICS e*Way installation process will install the following files, see the table **"Files Created by the Installation" on page 13**, within the e*Gate directory tree. Files will be 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) |
|---|---|
| \classes\ | stccics.jar<br>stcutil.jar |
| \configs\cicsclient\ | cicsclient.def |
| etd | cics.ctl |
| \etd\cicsclient | cicsclient.xsc |
| \ThirdParty\ibmctg\classes | ctgclient.jar<br>ctgserver.jar |
| \ThirdParty\gnu-getopt\classes | gnu-getopt.jar |

## 2.5 CICS Transaction Gateway 4.0 Configuration

IBM CICS Transaction Gateway 4.0 is required for Java-enabled CICS e*Ways. The following describes how to configure CICS Transaction Gateway 4.0. Transaction Gateway properties are set using the CTG Configuration Tool. The Configuration Tool is located under the CICS Transaction Gateway program menu.

For system specific settings consult the CICS Transaction Gateway Documentation or visit the IBM CICS Library Website at **http://www-4.ibm.com/software/ts/cics/library/manuals/ctg40dl.html#configs**.

***Note:*** *The CICS e*Way runs and has been tested using TCP62 connectivity provided by the CICS Transaction Gateway. The Transaction Gateway supports SNA communications on Windows and AIX platforms, but the CICS e*Way has not been tested using SNA.*

Chapter 3

# Multi-Mode e*Way Configuration

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

## 3.1 Multi-Mode e*Way

Multi-Mode e*Way properties are set using the Enterprise Manager.

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

1. Select the Navigator's Components tab.
2. Open the host and control broker 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** edit its properties.
6. When the e*Way Properties window opens, click on the **Find** button beneath the **Executable File** field, and select an executable file. For the purposes of the sample select **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.

The Multi-Mode e*Way configuration parameters are organized into the following section:

- JVM Settings

### 3.1.1 JVM Settings

The JVM Settings control basic Java Virtual Machine settings.

## JNI DLL Absolute Pathname

### Description

Specifies the absolute pathname to where the JNI DLL installed by the *Java 2 SDK 1.3* 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 |
|---|---|
| NT 4.0/ Windows 2000 | jvm.dll |
| Solaris 2.6, 2.7, 2.8 | libjvm.so |
| OS/390 | libjvm.so |
| HP-UX | libjvm.sl |
| AIX 4.3 | libjvm.a |

The value assigned can contain a reference to an environment variable, by enclosing the variable name within a pair of % symbols. For example:

```
%MY_JNIDLL%
```

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

*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 Java VM.

### Required Values

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

### Additional Information

If left unset, no paths will be 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 Java VM. 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) will be set.

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

### 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 Java VM will be 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 Java VM will be 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 will be 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 Java VM will be used.

### Required Values

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

## Class Garbage Collection

### Description

Specifies whether the Class Garbage Collection will be done automatically by the Java VM. The selection affects performance issues. Reserved for future use. Do not change from default value.

### Required Values

**YES** or **NO**.

## Garbage Collection Activity Reporting

### Description

Specifies whether garbage collection activity will be reported for debugging purposes. Reserved for future use. Do not change from default value.

### Required Values

**YES** or **NO**.

## Asynchronous Garbage Collection

### Description

Specifies whether asynchronous garbage collection activity will be reported for debugging purposes.

### Required Values

**YES** or **NO**.

## Report JVM Info and all Class Loads

### Description

Specifies whether the JVM information and all class loads will be reported for debugging purposes. Reserved for future use. Do not change from default value.

### Required Values

**YES** or **NO**.

## Disable JIT

### Description

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

### Required Values

**YES** or **NO**.

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

## Allow Remote Debugging of JVM

### Description

Specifies whether to allow remote debugging of the JVM.

### Required Values

**YES** or **NO**.

Chapter 4

# e*Way Connection Configuration

This chapter describes how to configure the Java-enabled CICS e*Way Connection Configuration.

## 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 the **Create a New e*Way Connection** button.
3. The **New e*Way Connection Component** dialog box opens, enter a name for the new **e*Way Connection**. Click **OK**.
4. Double-click on the new **e*Way Connection**. For this example, the connection has been defined as **ec_CICS**.
5. The **e*Way Connection Properties** dialog box opens.
6. From the **e*Way Connection Type** drop-down box, select **CICS**.
7. Enter the **Event Type "get"** interval in the dialog box provided. The configured default is 100 milliseconds.
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**.)
9. The **e*Way Connection Edit Settings** window opens. Make any necessary changes to the CICS e*Way Connection parameters.
10. Go to **File**, **Save** to save settings.
11. Go to **File**, **Promote to Run Time**.

***Note:*** *If changes are made to an existing e*Way Connection file, any e*Ways using the revised e*Way Connection must be restarted.*

The CICS e*Way Connection configuration parameters are organized into the following sections:

- **Connector** on page 20

## 4.1.1 Connector

This section contains a set of top level parameters:

- type
- class
- Property.Tag

### Type

#### Description

Specifies the connector type.

#### Required Values

**CICS**. The value always defaults to CICIS for CICS connections.

### Class

#### Description

Specifies the class name of the CICS Client connector object.

#### Required Values

A valid package name. The default is com.stc.eways.cics.CicsClientConnector.

### Property.Tag

#### Description

Specifies the data source identity. This parameter is required by the current EBobConnectorFactory.

#### Required Values

A valid data source package name.

## 4.1.2 CICS Gateway

This section contains the following parameters for CICS Java Gateway setup:

- URL
- Port
- SSL KeyRing Class
- SSL KeyRing Password

- CICS User
- CICS user password

### Url

#### Description

Specifies the remote or local Gateway to which it is to connect.

#### Required Values

A valid remote or local Gateway (node name or IP address).

### Port

#### Description

Specifies the TCP/IP port to connect to.

#### Required Values

An integer ranging from 1 to 864000.

### SSL KeyRing Class

#### Description

Specifies the full classname of the SSL KeyRing class.

#### Required Values

A valid full classname.

### SSL KeyRing Password

#### Description

Specifies the PASSWORD for the encrypted KeyRing class.

#### Required Values

A valid password for the SSL KeyRing class.

## 4.1.3 CICS Client

This section contains the following parameters for CICS Client setup:

- CICS UserId
- CICS Password
- ECI call type
- CICS Program
- CICS TransId
- COMMAREA length

- ECI extend mode
- ECI LUW token
- Message qualifier
- Encoding

## Cics UserId

### Description

Specifies the ID of the CICS user. Maximum length is eight characters.

### Required Values

A valid CICS user ID, eight characters or less.

## Cics Password

### Description

Specifies the password for the CICS user. Maximum length is eight characters.

### Required Values

A valid password for the user ID, eight characters or less.

## ECI call type

### Description

Specifies whether the ECI call type is Asynchronous or Synchronous.

- Synchronous Calls will wait for the transaction to complete, then return the contents of the commarea.
- Asynchronous calls will *not* wait for the transaction to complete, so no data is returned.

For further detail, see the IBM publication *"CICS Family: Client/Server Programming"* (document number SC33-1435-03).

### Required Values

Asynchronous or Synchronous. Synchronous is the configured default.

## CICS Program

### Description

Specifies the CICS program to be run on the server. Maximum length is eight characters.

### Required Values

A valid CICS program name, eight characters or less.

## CICS TransId

### Description

Specifies the CICS TransId to be run on the server. Maximum length is four characters.

### Required Values

A valid CICS TransId, four characters or less.

## COMMAREA length

### Description

Specifies the length (in bytes) of the communication area (COMMAREA) passed to the ECI.

### Required Values

An integer in the range of 1 to 32659. The configured default is 1000.

## ECI extend mode

### Description

Specifies whether a logical unit of work is terminated at the end of a call.

- **No** (ECI_NO_EXTEND). IF the input eci_luw_token field is zero, this will be the only call for a logical unit of work. If the input eci_luw_token field is not zero, this will be the last call for the specified logical unit of work. In either case, changes to recoverable resources are committed by a CICS end-of-task syncpoint, and the logical unit of work ends.
- **Yes** (ECI_EXTENDED). If the input eci_luw_token field is zero, this will be the first call for a logical unit of work that is to be continued. If the input eci_luw_token field is not zero, this call will continue the specified logical unit of work. In either case the logical unit of work continues after the called program completes, and changes to recoverable resources remain uncommitted

### Required Values

Yes or No. The configured default is No.

## ECI LUW token

### Description

Specifies the logical unit of work to which a call belongs. This must be set to zero at the start of a logical work unit. The ECI will update the value on the first or only call of the logical work unit. If the unit of work is to be extended, this value should be used as input to all subsequent calls associated with the same logical work unit.

If the return code is not ECI_NO_ERROR and a call is ending or continuing an existing logical work unit, then this field will be used to report the state of the logical work unit. If it is **zero**, the logical work unit has ended and updates have been backed out. If it is **not zero**, the value is the same as the input value. The logical work unit is continuing, and updates are still pending.

### Required Values

A valid integer in the range of 0 to 1000. The configured default is 0. This is a required input and output parameter.

## Message qualifier

### Description

The ECI Message Qualifier identifies each asynchronous call if more than one call is made.

### Required Values

A valid integer in the range of 0 to 1000. This is an optional input parameter,

## Encoding

### Description

Specifies default encoding.

### Required Values

The Canonical Name for any encoding set supported by Sun's Java Runtime Environment 1.1.8 (contained in rt.jar and i18n.jar). Examples are ASCII and Cp500 (EBCDIC). When running the CICS e*Way on the OS/390 platform, encoding should be set to "ISO-8859-1”.

# 4.1.4 Tracing

This section contains a set of top level parameters:

- Level
- Filename
- Truncation Size
- Dump Offset
- Timing

## Level

### Description

Specifies the level of trace information available. Options are:

- 0 - None: no CICS Java client application tracing.
- 1 - Standard: By default, it displays only the first 128 bytes of any data blocks (for example the commarea, or network flows). This trace level is equivalent to the Gateway trace set by the ctgstart -trace option. (Can also set using System property "gateway.T.trace=on".)
- 2 - Full Debug: By default, it traces out the whole of any data blocks. The trace contains more information about CICS Transaction Gateway than the standard trace

level. This trace level is equivalent to the Gateway debug trace set by the ctgstart -x option. (Can also set using System property "gateway.T=on".)

- 3 - Exception Stacks: It traces most Java exceptions, including exception which are expected during normal operation of the CICS Transaction Gateway. No other tracing is written. This trace level is equivalent to the Gateway stack trace set by the ctgstart -stack option. (Can also set using System property "gateway.T.stack=on".)

### Required Values

An integer in the range of 0 to 3.

## Filename

### Description

Integer-set. Specifies a file location for writing the trace output. This is as an alternative to the default output on stderr. Long filenames must be surrounded by quotation marks, for example: "trace output file.log". (Can also be set using System property "gateway.T.setTFile=xxx" where xxx is a filename.)

### Required Values

A valid output file name.

## Truncation Size

### Description

Specifies the maximum size of any data blocks that will be written in the trace. Specifying 0 will cause no data blocks written in the trace. Leave it blank if you do not want to specify truncation size. (Can also be set using System property "gateway.T.setTruncationSize=xxx" where xxx is a number.)

### Required Values

An integer in the range of 0 to 864000. The configured default is 100.

## Dump Offset

### Description

Specifies the offset from which displays of any data blocks will start. If the offset is greater than the total length of data to be displayed, an offset of 0 will be used. (Can also be set using System property "gateway.T.setDumpOffset=xxx" where xxx is a number.)

### Required Values

An integer in the range of 0 to 864000.

## Timing

### Description

Specifies whether or not to display time-stamps in the trace. (Can also be set using System property "gateway.T.timing=on".)

### Required Values

Off or On. The configured default is On.

Chapter 5

# Implementation

This chapter includes information pertinent to implementing the Java-enabled CICS e*Way in a production environment. Also included is a sample schema.

The following assumptions are applicable to this implementation: 1) The CICS e*Way has been successfully installed. 2) The executable and the configuration files have been appropriately assigned. 3) All necessary .jar files are accessible.

## 5.1 Using the Cobol Copybook Converter

The Cobol Copybook Converter is a build tool that takes a Cobol copybook as input and creates an ETD .ssc file. Use the SSC Wizard feature of the ETD Editor to create Java ETDs. These ETDs can be used to map the contents of the CICS commarea, allowing parsing and data conversion as needed.

For complete instructions on using the Copybook Convertor, see the ***Cobol Copybook Converter User's Guide.***

## 5.2 CICS Sample Implementation

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 CICS e*Way, you will do the following:

- Make sure that the Control Broker is activated.
- In the e*Gate Enterprise Manager, define and configure the following as necessary:
  - Inbound e*Way using **stcewfile.exe**
  - Outbound 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 will be published prior to being sent to the external system.

The following sections describe how to define and associate each of the above components. However, the section **"Sample Schema" on page 47** provides the details necessary to create the components of a specific schema consisting of three e*Ways, three Event Types, one Collaboration Rule, two Intelligent Queues and three Collaborations.

## 5.2.1 Creating the New Schema

A sample schema for the CICS e*Way is available in ..\samples\ewcics\.. on the CD-ROM. In addition, the following pages explain how the components for the CICS e*Way sample schema were created.

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 CICS e*Way, do the following:

1. Start the e*Gate Enterprise Manager GUI.
2. When the Enterprise Manager prompts you to log in, select the host that you specified during installation, and enter your password.
3. You will then be prompted to select a schema. Click **New**.
4. Enter a name for the new Schema. In this case, enter **CICSSample**, or any name as desired.
5. To import the sample schema select **Create from Export**, and use **Find** to locate the and select the sample .zip file on the CD-ROM. For Windows or UNIX operating systems select CICSJava_Sample.zip. When running the e*Way on OS/390, use the CICSSample_OS390 schema. Click **Open**.

The e*Gate Enterprise Manager opens under your new schema. You are now ready to begin creating the necessary components for this sample schema.

## 5.2.2 Event Types

The CICS e*Way installation includes the file "**cicsclient.xsc**" which represents a standard CICS Event Type template.

### Creating an Event Type Using the Standard ETD Wizard

For the purpose of this example, the following procedure shows how to create an ETD using the Standard ETD Wizard.

1. Highlight 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 to create a new **Event Type**.

3 Enter the name of the event type, then click **OK.** (For the purpose of this sample, the first Event Type is defined as "**etd_Blob**.")

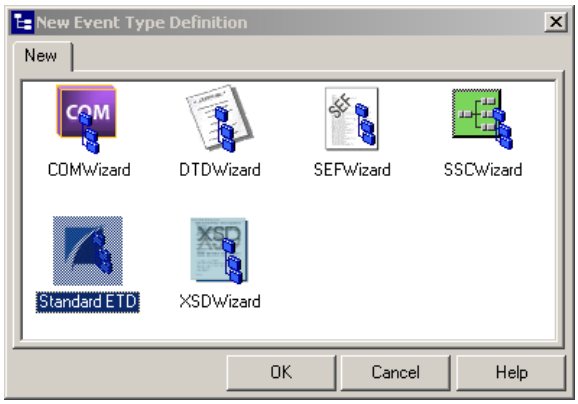
4 Double-click the new event type to edit its properties.

5 When the **Properties** window opens, click the **New** button. The ETD Editor opens.

6 Select **New** from the **File** menu on **Task Manager**.

7 The **Event Type Definition Wizard** opens.

**Figure 2** The New Event Type Definition Wizard




8 Select the appropriate wizard. (For this Event Type, select **StandardETD**.)

9 Enter a root node name. (For this example type **GenericIn**.)

10 Enter a package name where the ETD Editor can place all the generated Java classes associated with the created ETD. (For this sample, use **com.stc.GenericBlob** as the package name.)

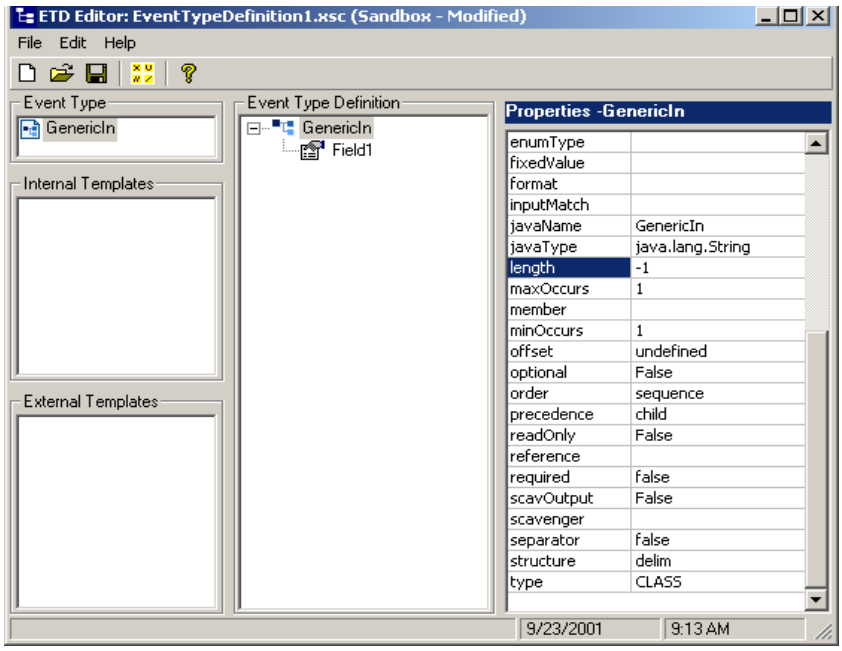
11 Click **OK** and Finish to accept the names and open the **ETD Editor**.

12 Select **GenericIn** in the Event Type Definition pane.

13 Right click, and select **Add Field, as Child Node**.

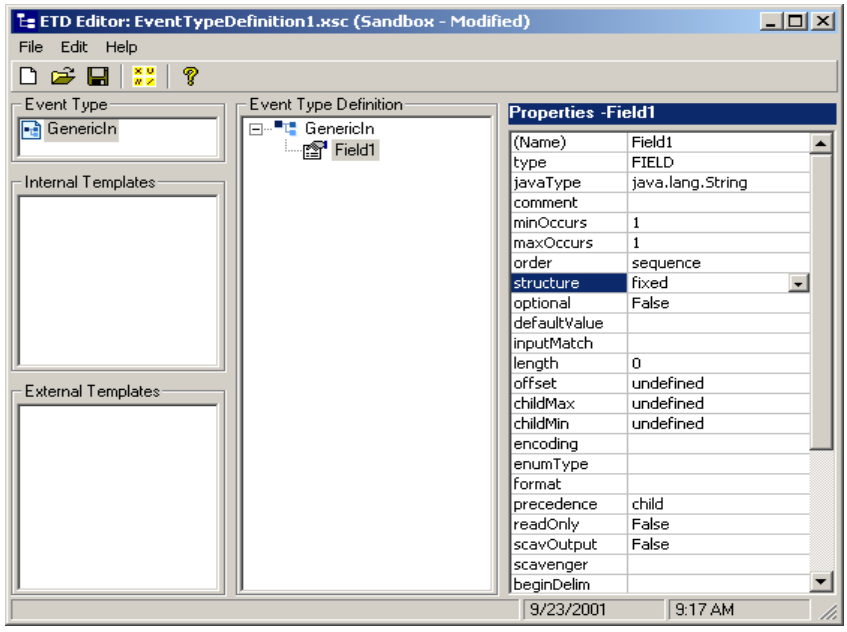
14 Select **GenericIn** and change the **length** value under the **Properties** pane to **-1** in the Properties window.

**Figure 3** Event Type Definition Editor




15 Select **Field1** and change the **structure** value under the **Properties** pane to **fixed** in the Properties window.

**Figure 4** ETD Editor Properties




16 Click **File, Compile and Save**, saving the file as **BlobGeneric**. When the file has been compiled the methods will appear in the **Event Type Definition** pane.

17 Click **File**, **Promote to Run Time**. The ETD Editor closes

### Creating an Event Type From an Existing .xsc

For the purpose of this example, the following procedure shows how to create an **Event Type Definition** (**ETD**) from an existing **.xsc** file using **cicsclient.xsc** as the input file. The **cicsclient.xsc** comes with the CICS e*Way and is used when creating all Schemas.

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 "**etd_CICSClient**.")
4. Double-click the new **Event Type** to edit its properties.
5. When the **Properties** window opens, click the **Find** button.
6. Browse to and select **cicsclient.xsc** (provided as the default destination **.xsc** file).
7. Click **Apply** and **OK** to close the Event Type Properties dialog box.

## 5.2.3 Creating and Configuring the e*Ways

The first components to be created are the following e*Ways.

- **"Creating the Inbound e*Way (Feeder)" on page 31**
- **"Creating the Outbound e*Way (Eater)" on page 32**
- **"Creating the Multi-Mode e*Way (CICSClient)" on page 33**

The following sections provide instructions for creating each e*Way.

#### Creating the Inbound e*Way (Feeder)

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 will manage 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 "**Feeder**"), 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 following settings for this configuration file.

:

**Table 2** Configuration Parameters for the Inbound e*Way

| Parameter | Value |
|---|---|
| **General Settings (unless otherwise stated, leave settings as default)** | |
| AllowIncoming | YES |
| AllowOutgoing | NO |

**Table 2** Configuration Parameters for the Inbound e*Way

| Parameter | Value |
|---|---|
| **Outbound Settings** | Default |
| **Poller Inbound Settings** | |
| PollDirectory | C:Indata (input file folder) |
| InputFileExtension | *.fin (input file extension) |
| PollMilliseconds | 1000 |
| Remove EOL | YES |
| MultipleRecordsPerFile | YES |
| MaxBytesPerLine | 4096 |
| BytesPerLineIsFixed | NO |
| File Records Per eGate Event | 1 |
| **Performance Testing** | Default |

9 After selecting the desired parameters, save the **configuration** file (as "**Feeder.cfg**").

10 Click **File**, **Promote to Run Time**. This will close 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.

### Creating the Outbound e*Way (Eater)

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 will manage 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 "**Eater**"), then click **OK**.

6 Select the new e*Way, right-click and select **Properties** to edit its properties.

7 When the **e*Way Properties** window opens, click the **Find** button beneath the **Executable File** field. Select **stcewfile.exe** as the executable file.

8 Under the **Configuration File** field, click the **New** button. The **Edit Settings** window opens. Select the following settings for this configuration file.

**Table 3** Configuration Parameters for the Outbound e*Way

| Parameter | Value |
|---|---|
| **General Settings (unless otherwise stated, leave settings as default)** | |
| AllowIncoming | NO |
| AllowOutgoing | YES |
| **Outbound Settings** | |
| OutputDirectory | C:\DATA |
| OutputFileName | output%d.dat |
| MultipleRecordsPerFile | NO |
| MaxRecordsPerFile | 10000 |
| AddEOL | YES |
| **Poller Inbound Settings** | Default |
| **Performance Testing** | Default |

9 Save the .cfg file (**Eater.cfg**), and click **File**, **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 the e*Way.

11 Use **Security** to view or set privilege assignments.

12 Click **OK** to close the e*Way Properties window.

### Creating the Multi-Mode e*Way (CICSClient)

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 will manage 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, "**CICSClient**"), 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.

8 To edit the JVM Settings, select **New** (or **Edit** if you are editing the existing .cfg file) under Configuration file.

See **"Multi-Mode e*Way Configuration" on page 14** for details on the parameters associated with the Multi-Mode e*Way.

9 Save the **.cfg** file (**CICSClient.cfg**), and click **File**, **Promote to Run Time**.

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.

## 5.2.4 Create the e*Way Connection

The e*Way Connection configuration file contains the connection information along with the information needed to communicate using CICS.

### 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, "**eWc_CICSClient**"), then click **OK.**

4 Double-click the new e*Way Connection to edit its properties.

5 The e*Way Connection Properties window opens. Select **CICS** from the **e*Way Connection Type** drop-down menu.

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.

7 Under e*Way Connection Configuration File, click the **New** button.

8 The e*Way Connection Editor opens. Select the following parameters listed in Table 4. For more information on the CICS e*Way Connection parameters, see **"e*Way Connection Configuration" on page 19**.

**Table 4** e*Way Connection Configuration Parameters

| Parameter | Value |
|---|---|
| **connector (unless otherwise stated, leave settings as default)** | |
| type | CICS |
| class | com.stc.eways.cics.CicsClientConnector |
| **CICS Gateway** | |
| Port | 8888 |
| AddEOL | YES |
| **CICS Client** | Default |
| CICS Program | STCPROGB |

**Table 4** e*Way Connection Configuration Parameters

| Parameter | Value |
|---|---|
| COMMAREA length | 1000 |
| ECI extend mode | No |
| ECI LUW token | 0 |
| Message qualifier | 0 |
| Encoding | cp500 |
| **Performance Testing** | |
| Level | 0 |
| Filename | CICSJava_Trace.txt |
| Truncation Size | 100 |
| Dump Offset | 0 |
| Timing | On |

9 Save the .cfg file (**eWc_CICSClient.cfg**)and click **File**, **Promote to Run Time**.

## 5.2.5 Intelligent Queues

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 CICS 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 "**IQ1**"), 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 **SeeBeyond_Standard** IQ Service provides sufficient functionality for most applications. If specialized services are required, custom IQ Service DLLs may be created.

***Note:*** *When running the CICS e*Way on the OS/390 platform, always select an IQ Manager Type of SeeBeyond Standard. The SeeBeyond JMS IQ Manager is not available on the OS/390.*

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

11 For this schema, repeat steps 1 through 10 to create an additional IQ (for this sample, "**IQ2"**).

## 5.2.6 Collaboration Rules

The next step is to create the Collaboration Rules that will 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**. The default should be set to **Java**.

The sample schema calls for the creation of three collaboration Rules files.

- **PassThru** (Pass Through)
- **CICSClient** (Java)

### Creating Collaboration Rules files

#### Pass Through

1 Select the Navigator's **Components** tab in the e*Gate Enterprise Manager.

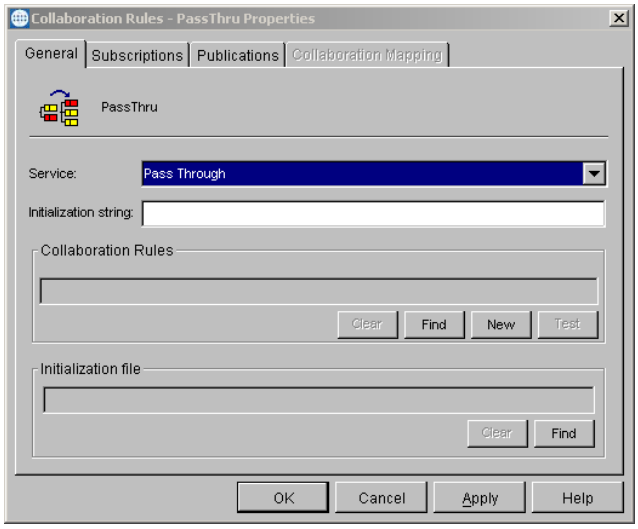
2 In the Navigator, select the **Collaboration Rules** folder.

3 On the palette, click the **Create New Collaboration Rules** button.

4 Enter the name of the new Collaboration Rule Component (for this case "**PassThru**"), then click **OK**.

5 Double-click the new Collaboration Rules Component. The **Collaboration Rules Properties** window opens.

**Figure 5** Collaboration Rules Properties - Pass Through

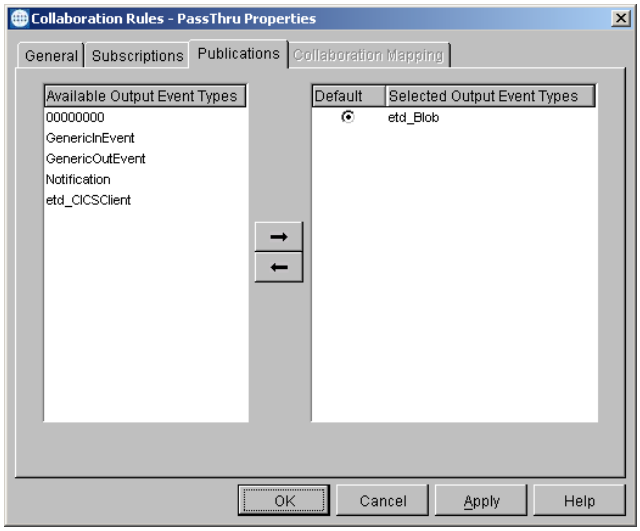



6 The **Service** field defaults to **Pass Through**.

7 Go to the **Subscriptions** tab. Select **etd_Blob** under **Available Input Event Types**, and click the right arrow to move it to **Selected Input Event Types**. The box under **Triggering Event** should be checked.

8 Go to the **Publications** tab. Select **etd_Blob** under **Available Output Event Types**, and click the right arrow to move it to **Selected Output Event Types**. Make sure that **etd_Blob** is selected as the default.

**Figure 6** Collaboration Properties




9 Click **OK** to close the **Collaboration Rules - Pass_In Properties** window.

## Java (CICSClient)

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, use **CICSClient**).

5 Double-click the new Collaboration Rules Component to edit its properties. The Collaboration Rules Properties window opens.

6 From the **Service** field drop-down 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:

10 In the Instance Name column, enter **CICSOut** for the instance name.

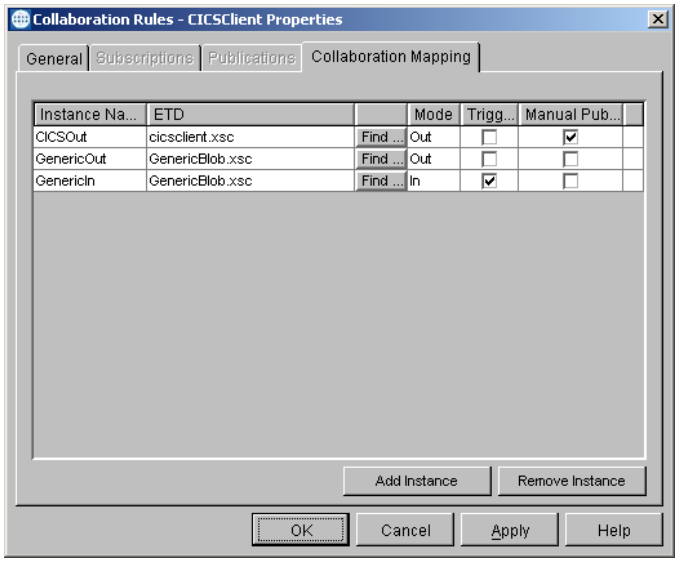
11 Click **Find,** navigate to and double-click **cicsclient.xsc**. This adds **cicsclient.xsc** to the **ETD** column for this instance.

12 In the **Mode** column, select **In** from the drop–down list box. To access the drop-down list box, click the right portion of the **Mode** field for this instance.

13 In the **Trigger** column, make sure that the checkbox is cleared (no trigger).

14 In the **Manual Publish** column, make sure the checkbox is selected.

**Figure 7** Collaboration Rules - Collaboration Mapping




15 Repeat steps 9–13 using the following values:

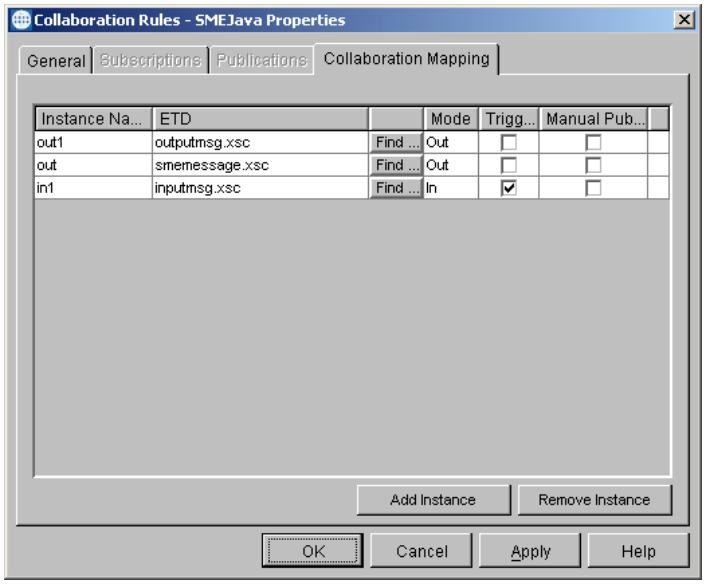
- Instance Name — **GenericOut**
- ETD — **GenericBlob.xsc**
- Mode — **Out**

- Trigger — clear
- Manual Publish - clear

16 Repeat steps 9–13 again using the following values:

- Instance Name — **GenericIn**
- ETD — **GenericBlob.xsc**
- Mode — **In**
- Trigger — select
- Manual Publish - clear

**Figure 8** Collaboration Rules - Collaboration Mapping Properties




17 Select the **General** tab, under the Collaboration Rule box, select **New**. The **Collaboration Rules Editor** opens.

18 Expand to full size for optimum viewing, expanding the Source and Destination Events as well. The following section describes the setting up the collaboration rules for **CICSclient** using the Java Collaboration Rules Editor.

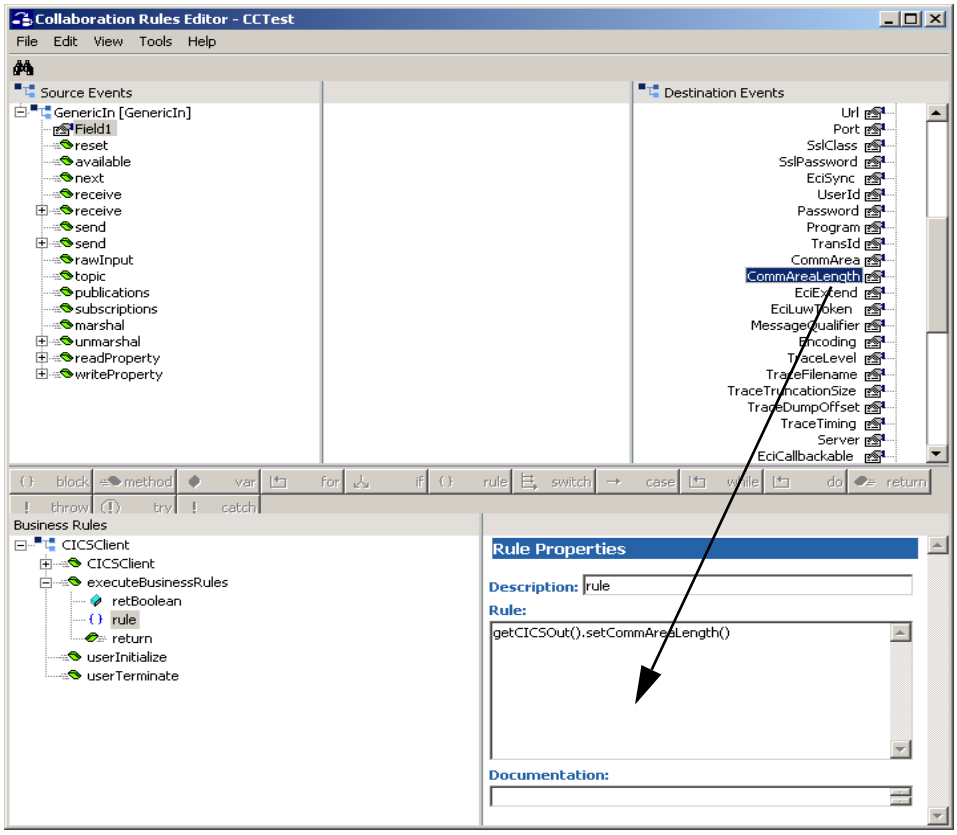
## Creating the Collaboration Rules Class

The section is given as an example of how to create the Collaboration Rules Class using the Java Collaboration Rules Editor. The completed Collaboration Rules .xpr file is included with the sample schema on the CD. The following section gives a number of examples that demonstrate how these rules were setup. Please refer to the completed class, **CICSClient.class** when completing the Collaboration Rules Properties.

1 As stated in the above section, the Java Collaboration Rules Editor opens from the Collaboration Rules Properties dialog box when the Collaboration Rules field, New or Edit button is clicked. Expand to full size for optimum viewing, expanding 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.

**Figure 9** The Collaboration Rules Editor




3 From the Collaboration Rules Editor toolbar, click on **rule**. This places a **rule** "space" in the Business Rules pane, to which the user can add the Java expression. A **rule** space is now available under **retBoolean** in the Business Rules window. Select the new **rule**.

4 With the new Rule selected, drag-and-drop **CommAreaLength** from the **Destination Events** pane into the Rule Properties, Rules window. This enters the Java code in the Rule Properties, Rules window.

5 In the Rules window, place the curser between the parenthesis following **CommAreaLength** and type **1100**. This sets the sets the length to 1100 bytes.

6 Select retBoolean again and click on **rule** in the toolbar. An additional Rule space is placed in the Business Rules above the earlier Rule. Select the new Rule.

7 In Rule Properties, Description, type **logging**. This now appears as the name of the rule.

8 Place the curser in the Rule Properties, Rules window. Type the following:

```
EGate.traceln (EGate.TRACE_EWAY, EGate.TRACE_EVENT_DEBUG, "Inside
Rule")
```

9 Select **logging** in the Business Rules pane, right-click and select **copy**.

10 Select **rule** in the Business Rules pane, right-click and select **paste**. **logging** is pasted below the rule.

11 With the new "logging" rule selected, go to the Rule Properties, Rules window, and change `the code from "Inside Rule"` to `"Finish Set Array"`.

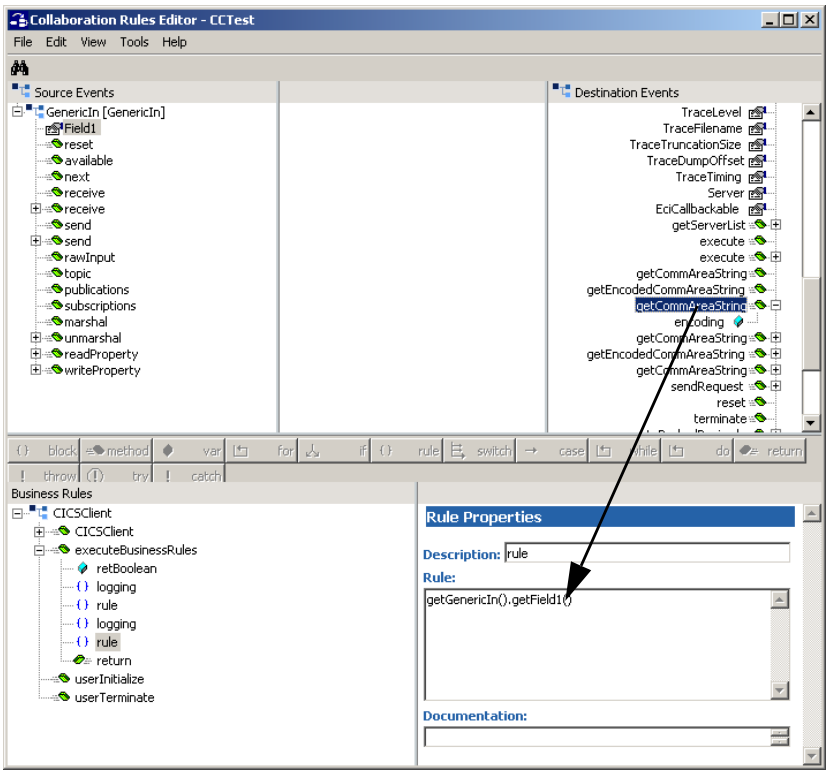
12 Click **rule** on the toolbar again. A new rule is placed under **logging** in the Business Rules pane. Select the new rule.

13 With the new rule selected, drag-and-drop **Field1** from the Source Events pane to the Rules window. this places the following code in the Rules window:

```
getGenericIn().getField1()
```

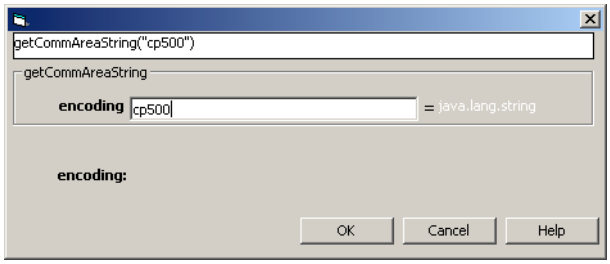
14 From the Destination Events pane, drag and drop the **getCommAreaString** method (under getEncodedCommAreaString) into the second set of parenthesis in the Rules window (see **Figure 10 on page 41**).

**Figure 10** The Collaboration Rules Editor - Rules window




15 A Properties dialog box opens for **getCommAreaString encoding**. Type in **cp500** and click **OK**. (See **Figure 11 on page 42**)

**Figure 11** Properties window - getCommAreaString encoding




16 The code in the Rules window should now appear as follows:

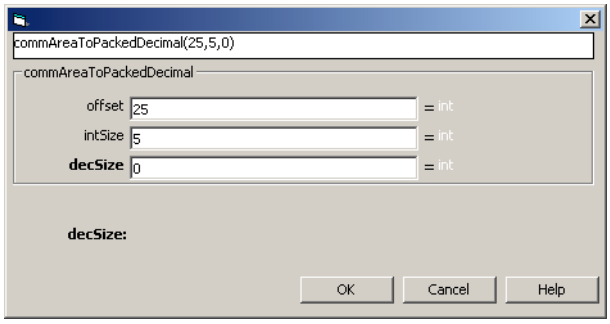
```
getGenericOut().setField1(getCICSOut().getCommAreaString("cp500"))
```

17 In the **Business Rules** pane, select the last **rule**. From the toolbar, click the **var** button to add a variable. In the **Rule Properties** pane, **Description** field, type **pd**.

18 In the **Rule Properties** pane, **Name** field, type **pd**.

19 From the Destination Events pane, drag-and-drop the commAreaPackedDecimal to the **Rule Properties** pane, **Initial Value** field. This opens a Properties window for **commAreaPackedDecimal**.

**Figure 12** Properties window - commAreaPackedDecimal




20 In the **commAreaPackedDecimal** method Properties window, enter **25** for the **offset** integer, **5** for the **intSize** integer, and **0** for the **decSize** integer, and click **OK**.

21 In the **Rule Properties** pane, **Type** field drop-down list box, select **com.stc.eways.cics.PackedDecimal**.

22 Select the **pd** variable in the **Business Rules** pane. Click on the rule button on the toolbar. A **rule** appears under the **pd** variable.

23 Select the new Rule. From the Source Events pane, drag-and-drop Field1 to the Rule Properties, Rules window. The following code appears.

```
getGenericIn().getField1()
```

24 From the Destination Events pane, drag-and-drop the packedDecimaltoString method into the second set of parenthesis in the Rules Window.

25 A **packedDecimaltoString** Properties window opens. Type **pd** in the **packedDec** field, and click **OK**. The code in the Rules window should appear as follows:

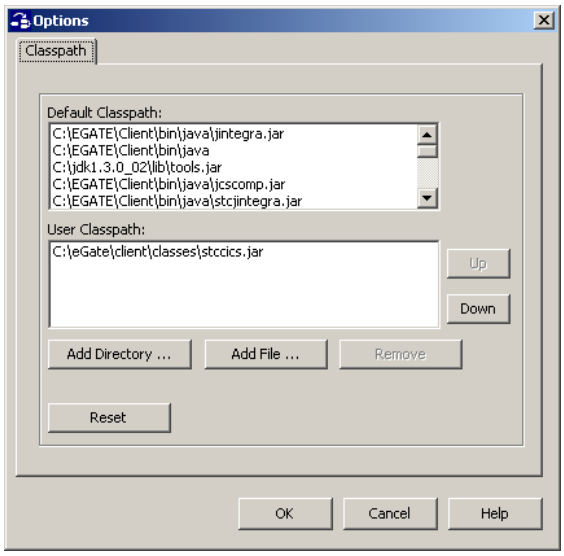
```
getGenericIn().getField1(getCICSOut().packedDecimalToString(pd))
```

26 When the Collaboration Rules are completed, click **File**, **Compile** to compile the new collaboration.

27 Before compiling the code, select **Tools**, **Options**,.

28 Verify that all necessary **.jar** files are included. Add **stccics.jar**.

**Figure 13** Business Rules




29 When all the business logic has been defined, the code can be compiled by selecting **Compile** from the **File** menu. The **Save** menu opens, provide a name for the **.**xpr file.

***Important:*** *This is not a complete collaboration, but an example of how the various components of the collaboration are setup. For the sample schema, select CICSClient.class in the Collaboration Rules - CICSClient Properties dialog box to use the completed CICSClient.xpr file. For further information on using the Collaboration Rules Editor see the* ***e*Gate Integrator User's Guide****.*

## 5.2.7 Collaborations

Collaborations are the components that receive and process Event Types and forward the output to other e*Gate components or to 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.

### Create the CICIS_Multi_Mode 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 an e*Way to assign the Collaboration (for this sample, "**CICSClient**").

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, "**col_CICSclient**")

7 Double-click the new **Collaboration** to edit its properties. The **Collaboration Properties** dialog box opens.

8 From the **Collaboration Rules** drop-down list box select the Collaboration Rules file that you created previously (for the sample, "**CICSclient**").

9 In the **Subscriptions** area, click **Add** to define the input Event Types to which this Collaboration will subscribe.

- A From the **Instance Name** field drop-down list box, select the Instance Name that you previously defined "**GenericIn**."
- B From the **Event Type** drop-down list box, select the **Event Type** that you previously defined "**etd_Blob**."
- C From the **Source** drop-down list box, select the source (for this sample "**eWc_CICSClient**").

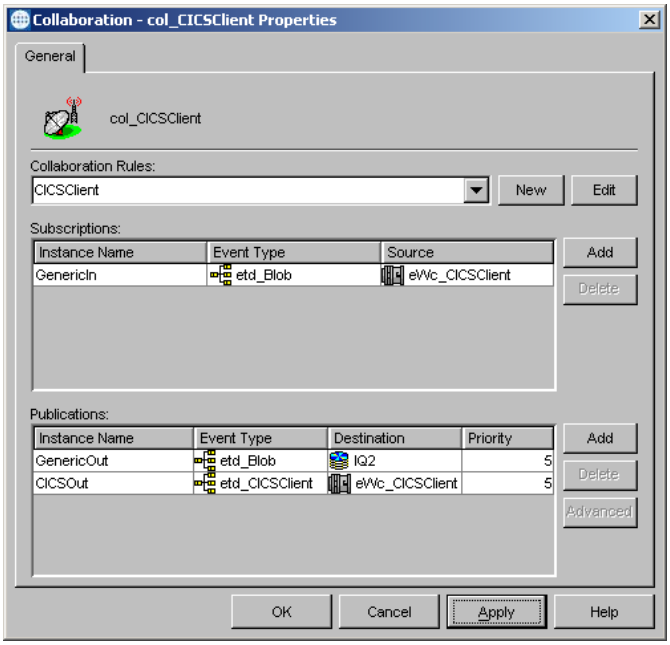
10 In the **Publications** area, click **Add** to define the output **Event Types** that this Collaboration will publish.

- A From the **Instance Name** drop-down list box, select the **Instance Name** that you previously defined "**GenericOut**."
- B From the **Event Types** drop-down list box, select the **Event Type** that you previously defined "**etd_Blob**."
- C Select the publication destination from the **Destination** drop-down list box. In this case, it should be "**IQ2**."
- D The value in the **Priority** column defaults to **5**.

11 In the **Publications** area, click **Add** to add an additional instance.

- A From the **Instance Name** drop-down list box, select the **Instance Name** that you previously defined "**CICSOut**."
- B From the **Event Types** drop-down list box, select the **Event Type** that you previously defined "**etd_CICSClient**."
- C Select the publication destination from the **Destination** drop-down list box. In this case, it should be "**eWc_CICSClient**."
- D The value in the **Priority** column defaults to **5**.

**Figure 14** Collaboration Properties - col_CICSClient




12 Click the **Apply** button and click **OK** to close the Collaboration Properties dialog box.

### Create the Inbound 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 **Feeder** e*Way to assign its Collaboration.

5 On the palette, click the **Create a New Collaboration** button.

6 Enter the name of the new Collaboration (for the sample, "**col_Feeder**") then click **OK.**

7 Double-click the new Collaboration to edit its properties. The **Collaboration Properties** dialog box opens.

8 From the **Collaboration Rules** drop-down list box, select the Collaboration Rules file that you created previously (for the sample, "**PassThru**").

9 In the **Subscriptions** area, click **Add** to define the input Event Types to which this Collaboration will subscribe.

- A From the **Event Type** drop-down list box, select the **Event Type** that you previously defined "**etd_Blob**."
- B Select the **Source** from the **Source** drop-down list box. In this case, it should be **<External>**.

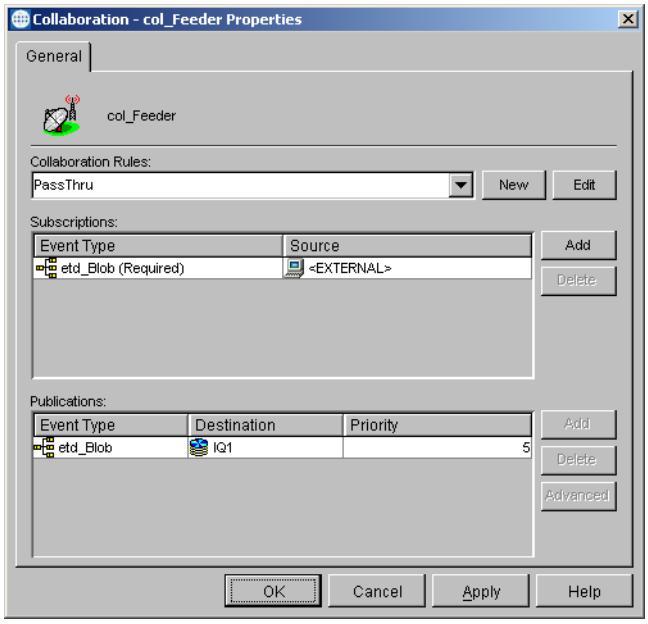
10 In the **Publications** area, click **Add** to define the output **Event Types** that this Collaboration will publish.

A From the **Event Types** list, select the **Event Type** that you previously defined "**etd_Blob**."

B Select the publication destination from the **Destination** list. In this case, it should be "**IQ1**."

C The value in the **Priority** column defaults to **5**.

**Figure 15** Collaboration Properties_col_Feeder




11 Click the **Apply** button and click **OK** to close the Collaboration Properties dialog box.

### Create the Outbound 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 **Eater** e*Way to assign its Collaboration.

5 On the palette, click the **Create a New Collaboration** button.

6 Enter the name of the new Collaboration (for this sample, "**col_Eater**"), then click **OK**.

7 Double-click the new **Collaboration** to edit its properties.

8 From the **Collaboration Rules** drop-down list box, select the Collaboration Rules file that you created previously (for the sample, "**PassThru**").

9 In the **Subscriptions** area, click **Add** to define the input Event Types to which this Collaboration will subscribe.

A From the **Event Type** drop-down list box, select the **Event Type** that you previously defined "**etd_Blob**."

B Select the **Source** from the **Source** list. In this case, it should be "**col_CICSClient**."

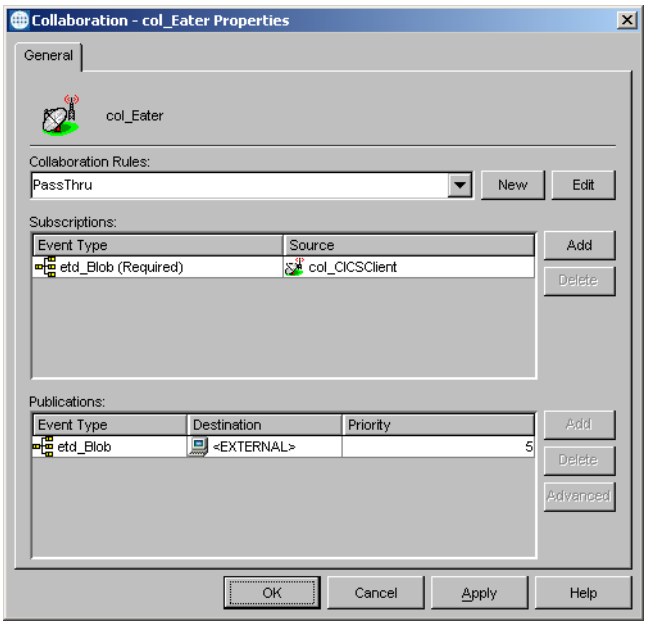
10 In the **Publications** area, click **Add** to define the output **Event Types** that this Collaboration will publish.

A From the **Event Types** list box, select the **Event Type** that you previously defined "**etd_Blob**."

B Select the publication destination from the **Destination** list. In this case, it should be **<External>**.

C The value in the **Priority** column defaults to **5**.

**Figure 16** Collaboration Properties_col_Feeder




11 Click the **Apply** button and click **OK** to close the Collaboration Properties dialog box.

## 5.3 Sample Schema

The previous sections provided the basics for implementing the CICS e*Way. This section describes how to use the CICS e*Way within a sample Schema. It is assumed that the CICS e*Way has been installed properly, and that all of the necessary files and scripts are located in the default location. When running the e*Way on OS/390, use the CICSJava_os390 sample schema.

This implementation will consist of two file-based e*Ways, one Multi-Mode e*Way, two Event Types, two Collaboration Rules, two Intelligent Queues and three Collaborations, as follows:

- **Feeder** - This e*Way will receive input from an external source, apply pass through Collaboration Rules, and publish the information to an Intelligent Queue.
- **CICSClient** - This Multi-Mode e*Way applies extended Java Collaboration Rules to an inbound Event to perform the desired business logic, in this case encryption and decryption.
- **Eater** - This e*Way will receive information from the Multi-Mode e*Way and publish to the external system.
- **etd_CICSClient** - This Event Type contains the methods to be used to perform the necessary transformation.
- **etd_Blob** - This Event Type describes an Event that is input to the extended Java Collaboration Service.
- **PassThru** - This Collaboration Rule is associated with the *ewInbound* e*Way, and is used for receiving the input Event and sending the Event to the External.
- **CICSClient** - This Collaboration Rule is associated with the CICSClient Multi-Mode e*Way, and is used to perform the transformation process.
- **IQ1** - This Intelligent Queue is an STC_JMS IQ, and forwards data to the *CICSClient* Multi-Mode e*Way.
- **IQ2** - This Intelligent Queue is a STC_JMS IQ, and forwards data to the outbound Eater e*Way.

### 5.3.1 Execute the Schema

To execute the CICSSample schema, do the following:

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

   ```
   stccb -rh hostname -rs CICSSample -un username -up user password
   -ln hostname_cb
   ```

   Substitute ***hostname***, ***username*** and ***user password*** as appropriate.
2. Exit from the command line prompt, and start the e*Gate Monitor GUI.
3. When prompted, specify the ***hostname*** which contains the Control Broker you started in Step 1 above.
4. Select the CICSSample schema.
5. After you verify that the Control Broker is connected (the message in the Control tab of the console will indicate command *succeeded* and status as *up*), highlight the IQ Manager, ***hostname***_igmgr, then click on the right button of the mouse, and select **Start**.
6. Select each of the e*Ways, right-click the mouse, and select **Start**.
7. To view the output, copy the output file (specified in the Outbound e*Way configuration file). Save to a convenient location, open.

***Note:*** *While the schema is running, opening the destination file, will cause errors.*

Chapter 6

# Java Methods

The e*Way's Java Methods fall into the following categories:

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 Cicsclient Class** on page 49

## 6.1 The Cicsclient Class

```
java.lang.Object
    com.stc.jcsre.SimpleETDImpl(implements com.stc.jcsre.ETD)
        com.stc.eways.cics.CicsClient
```

public class **CicsClient** extends com.stc.jcsre.

***Note:*** *Methods that include an "encoding" parameter should specify encoding as "ISO-8859-1" when the e*Way is run on the OS/390 platform.*

### Methods of the Cicsclient Class

These methods are described in detail on the following pages:

## CicsClient

### Description

Constructor.

### Syntax

```
public CicsClient()
```

### Parameters

None.

### Return Values

None.

### Throws

None.

## commAreaToPackedDecimal

### Description

Converts a packed decimal (Cobol comp-3) commarea field to a PackedDecimal object.

### Syntax

```
public com.stc.eways.cics.PackedDecimal commAreaToPackedDecimal(int
offset, int intSize, int decSize)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| offset | integer | Offset of the packed decimal field relative to the start of the commarea (a field starting in byte 1 would have an offset of 0) |
| intSize | integer | The number of integer digits in the resulting object. |
| decSize | integer | The number of decimal digits in the resulting object. |

### Return Values

**com.stc.eways.cics.PackedDecimal**
Returns the new PackedDecimal object.

### Throws

None

## commAreaZonedToString

### Description

Converts a zoned decimal (Cobol PIC S9) commarea field to a string.

### Syntax

```
public java.lang.String commAreaZonedToString(int offset, int length)

public java.lang.String commAreaZonedToString(int offset, int length,
java.lang.String encoding)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| offset | integer | Offset of the zoned decimal field relative to the start of the commarea (a field starting in byte 1 would have an offset of 0). |

| Name | Type | Description |
|---|---|---|
| length | integer | The length of the zoned decimal field. |
| encoding | java.lang.String | The desired character encoding type (ASCII, cp500 for EBCDIC, etc.) |

***Note:*** *Methods that include an "encoding" parameter should specify encoding as "ISO-8859-1" when the e*Way is run on the OS/390 platform."*

### Return Values

**java.lang.String**
Returns the new string.

### Throws

None.

## execute

### Description

Executes the CICS program.

### Syntax

```
public void execute()

public void execute(boolean eciSynCall, java.lang.String
cicsServerName, java.lang.String cicsUserId, java.lang.String
cicsPassword, java.lang.String cicsProgram, java.lang.String
cicsTransId, byte[] byteArray, int length, boolean eciExtendMode, int
eciLUWToken, int msgQualifier, com.stc.eways.cics.Callbackable
eciCallbackableObj)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| eciSynCall | boolean | A Boolean value indicating whether to use ECI Synchronous Call. |
| cicsServerName | java.lang.String | The CICS server name. |
| cicsUserId | java.lang.String | The user id. |
| cicsPassword | java.lang.String | The password associated with the specified user id. |
| cicsProgram | java.lang.String | The CICS Program name to be executed. |
| cicsTransId | java.lang.String | The CICS transaction id. |
| byteArray | byte [] | A byte array for the COMMAREA length. |
| length | integer | The COMMAREA length |

| Name | Type | Description |
|---|---|---|
| eciExtendMode | boolean | A Boolean value indicating whether to implement ECI extend mode. |
| eciLUWToken | integer | An ECI LUW token (Logical Unit of Work token) |
| msgQualifier | integer | Application provided identifier |
| eciCallbackableObj | com.stc.eways.cics.Callbackable | ECI callbackable object. This may be null if no callback is required. |

### Return Values

None.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a connection error.

**com.stc.common.collabService.CollabDataException**
Indicating a data error.

## getCommArea

### Description

Constructs a CommArea object.

### Syntax

```
public byte[] getCommArea()
```

### Parameters

None.

### Return Values

**byte array**
Returns the CommArea byte array.

### Throws

None.

## getCommAreaLength

### Description

Constructs the CommArea length.

### Syntax

```
public int getCommAreaLength()
```

### Parameters

None

### Return Values

**integer**
Returns the CommArea length.

### Throws

None.

---

## getCommAreaString

### Description

Constructs a CommArea String by converting the CommArea array of bytes using the platform's default character encoding, or:

Constructs a CommArea String by converting the CommArea array of bytes with offset and length using the platform's default character encoding, or;

Construct a CommArea String by converting the CommArea array of bytes with offset and length using the character encoding specified as an argument, or;

Constructs a CommArea String by converting the CommArea array of bytes using the character encoding specified as an argument.

### Syntax

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

public java.lang.String getEncodedCommAreaString(int offset, int
length)

public java.lang.String getCommAreaString(int offset, int length,
java.lang.String encoding)

public java.lang.String getCommAreaString(java.lang.String encoding)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| offset | integer | Offset of the area to be converted relative to the start of the commarea (a field starting in byte 1 would have an offset of 0). |
| length | integer | The length of the area to be converted. |
| encoding | java.lang.String | The desired character encoding type (ASCII, cp500 for EBCDIC, etc.) |

***Note:*** *Methods that include an "encoding" parameter should specify encoding as "ISO-8859-1" when the e*Way is run on the OS/390 platform."*

### Return Values

**java.lang.String**
Returns the CommArea string.

### Throws

None.

## getEciCallbackable

### Description

Gets the ECI callbackable object.

### Syntax

```
public com.stc.eways.cics.Callbackable getEciCallbackable()
```

### Parameters

None.

### Return Values

**com.stc.eways.cics.Callbackable**
Returns the ECI callbackable value.

### Throws

None.

## getEciExtend

### Description

Determines whether the ECI LUW has been set to extended.

### Syntax

```
public boolean getEciExtend()
```

### Parameters

None

### Return Values

**boolean**
Returns true to indicate that the extended request is implemented; otherwise, returns false.

### Throws

None.

## getEciLuwToken

### Description

Gets the ECI LUW token value.

### Syntax

```
public int getEciLuwToken()
```

### Parameters

None.

### Return Values

**integer**
Returns the ECI LUW token value.

### Throws

None.

## getEciSync

### Description

Queries whether the state is set to synchronous.

### Syntax

```
public boolean getEciSync()
```

### Parameters

None.

### Return Values

**boolean**
Returns true to indicate that the ECI state is set to synchronous.

### Throws

None.

## getEncodedCommAreaString

### Description

Constructs a CommArea String by converting the CommArea array of bytes using the character encoding specified earlier for the ETD, or:

Constructs a CommArea String by converting the CommArea array of bytes with offset and length using the character encoding specified earlier for the ETD.

### Syntax

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

```
public java.lang.String getEncodedCommAreaString(int offset, int
length)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| offset | integer | Offset of the area to be converted relative to the start of the commarea (a field starting in byte 1 would have an offset of 0). |
| length | integer | The length of the area to be converted. |

### Return Values

**java.lang.String**
Returns the encoded CommArea string value.

### Throws

**java.io.UnsupportedEncodingException**
Indicating unsupported encoding.

## getEncoding

### Description

Gets the encoding key.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the encoding type.

### Throws

None.

## getHexString

### Description

Gets the hexidecimal string.

### Syntax

```
public static java.lang.String getHexString(byte[] byteArray)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| byteArray | byte [] | |

### Return Values

**java.lang.String**
Returns the hexidecimal string.

### Throws

None

## getMessageQualifier

### Description

Gets the Message Qualifier information.

### Syntax

```
public int getMessageQualifier()
```

### Parameters

None

### Return Values

**integer**
Returns the Message Qualifier information.

### Throws

None.

## getPassword

### Description

Gets the password and decrypts it.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the password.

### Throws

None.

## getPort

### Description

Gets the port information.

### Syntax

```
public int getPort()
```

### Parameters

None.

### Return Values

**integer**
Returns the port information.

### Throws

None.

## getProgram

### Description

Gets the name of the CICS program.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the CICS program name.

### Throws

None.

## getServer

### Description

Gets the CICS server information.

### Syntax

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

### Parameters

None

### Return Values

**java.lang.String**
Returns the name of the CICS server.

### Throws

None.

## getServerList

### Description

Gets a list of CICS servers defined.

### Syntax

```
public java.lang.String[] getServerList(int maxNumSystems)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| maxNumSystems | integer | The maximum number of systems. |

### Return Values

**java.lang.String[]**
Returns a list of the defined CICS servers.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a connection error.

**com.stc.common.collabService.CollabDataException**
Indicating a data error.

## getSslClass

### Description

Gets the name of the SSL class.

### Syntax

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

### Parameters

None

### Return Values

**java.lang.String**
Returns the name of the SSL class.

### Throws

None.

## getSslPassword

### Description

Gets the SSL password.

### Syntax

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

### Parameters

None

### Return Values

**java.lang.String**
Returns the SSL password.

### Throws

None.

## getTraceDumpOffset

### Description

Gets the trace dump offset value.

### Syntax

```
public int getTraceDumpOffset()
```

### Parameters

None.

### Return Values

**integer**
Returns the trace dump offset value.

### Throws

None.

## getTraceFilename

### Description

Gets the trace filename.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the trace filename.

### Throws

None.

---

## getTraceLevel

### Description

Gets the defined trace level value.

### Syntax

```
public int getTraceLevel()
```

### Parameters

None.

### Return Values

**integer**
Returns the trace level.

### Throws

None.

---

## getTraceTiming

### Description

Gets the defined trace timing information.

### Syntax

```
public boolean getTraceTiming()
```

### Parameters

None

### Return Values

**boolean**
Returns true to indicate the timing trace mask is implemented.

### Throws

None.

## getTraceTruncationSize

### Description

Gets the trace truncation size of the hex dumps.

### Syntax

```
public int getTraceTruncationSize()
```

### Parameters

None.

### Return Values

**integer**
Returns the size of the trace truncation setting.

### Throws

None.

## getTransId

### Description

Gets the transaction ID of the current transaction.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the transaction ID for the current transaction.

### Throws

None.

## getUrl

### Description

Gets the URL of the CICS Transaction Gateway.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the URL of the CICS Transaction Gateway.

### Throws

None.

## getUserId

### Description

Gets the user ID associated with the terminal.

### Syntax

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

### Parameters

None.

### Return Values

**java.lang.String**
Returns the user ID associated with the terminal or null if the user id is set to null or it is a basic terminal.

### Throws

None.

## handleConfigValues

### Description

Implements the values assigned in the configuration file for the e*Way Connection.

### Syntax

```
protected void handleConfigValues(java.util.Properties props)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| props | java.util.Properties | The configuration property values. |

### Return Values

None.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a communication error.

**com.stc.common.collabService.CollabDataException**
Indicating a data error.

## handleTrace

### Description

Implements the trace flags based on parsed configuration values.

### Syntax

```
public void handleTrace()
```

### Parameters

None.

### Return Values

None.

### Throws

None.

## initialize

### Description

Initializes the Event Type Definition.

### Syntax

```
public void initialize(com.stc.common.collabService.JCollabController
cntrCollab, java.lang.String sKey, int iMode)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| cntrCollab | com.stc.common.collabService.JCollabController | The Java CollabConroller object. |
| sKey | java.lang.String | The key to a JMsgObject |
| iMode | integer | Mode for the ETD. The possible values are:<br>IN_MODE<br>OUT_MODE<br>IN_OUT_MODE |

### Return Values

None.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a connection error.

**com.stc.common.collabService.CollabDataException**
Indicating a data error.

### Additional Information

Overrides initialize in class com.stc.jcsre.SimpleETDImpl.

## initJavaGateway

### Description

Initializes the Java Gateway object to allow the flow of data.

### Syntax

```
public void initJavaGateway()
```

### Parameters

None.

### Return Values

None.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a connection error.

**com.stc.common.collabService.CollabDataException**
Indicating a data error.

## packedDecimalToString

### Description

Converts a PackedDecimal object to a string.

### Syntax

```
public static java.lang.String
packedDecimalToString(com.stc.eways.cics.PackedDecimal pd)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| pd | com.stc.eway.cics.PackedDecimal | The PackedDecimal object to be converted. |

### Return Values

**java.lang.String**

### Throws

None.

## reset

### Description

Resets the data content of an ETD.

### Syntax

```
public boolean reset()
```

### Parameters

None.

### Return Values

**boolean**

Returns true if the reset clears the data content of the ETD; otherwise, returns false if the ETD does not have a meaningful implementation of reset( ), in which case it is necessary to create a new ETD.

### Throws

None.

### Additional Information

Overrides reset in class com.stc.jcsre.SimpleETDImpl

## sendRequest

### Description

Sends a flow of data contained in the ECI Request object to the Gateway, and determines whether the send has been successful by checking the return code.

### Syntax

```
public void sendRequest(com.stc.eways.cics.ECIRequest request)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| request | com.stc.eway.cics.ECIRequest | The ECI Request object to be sent. |

### Return Values

None.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a connection error.

**com.stc.common.collabService.CollabDataException**
Indicating a data error.

## setCommArea

### Description

Sets the CommArea to be made available to CICS.

### Syntax

```
public void setCommArea(byte[] byteArray)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| byteArray | byte[] | A byte array containing the information required to set the CommArea. |

### Return Values

None.

### Throws

None.

## setCommAreaLength

### Description

Sets the Commarea length.

### Syntax

```
public void setCommAreaLength(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | Description |

### Return Values

None.

### Throws

None.

## setEciCallbackable

### Description

Sets the ECI callbackable value.

### Syntax

```
public void setEciCallbackable(com.stc.eways.cics.Callbackable c)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| c | com.stc.eway.cics.Callbackable | ECI callbackable value. |

### Return Values

None.

### Throws

None.

## setEciExtend

### Description

Sets the ECI Extend Mode.

### Syntax

```
public void setEciExtend(boolean b)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| b | boolean | true sets the mode to extended. |

### Return Values

None.

### Throws

None.

## setEciLuwToken

### Description

Sets the ECI LUW token value.

### Syntax

```
public void setEciLuwToken(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | The application identifier. |

### Return Values

None.

### Throws

None.

## setEciSync

### Description

Sets the ECI to synchronous.

### Syntax

```
public void setEciSync(boolean b)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| b | boolean | true sets the mode to synchronous. |

### Return Values

None.

### Throws

None.

## setEncoding

### Description

Sets the encryption type for encoding purposes.

### Syntax

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

### Parameters

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

### Return Values

None.

### Throws

None.

## setMessageQualifier

### Description

Sets the Message Qualifier associated with this request.

### Syntax

```
public void setMessageQualifier(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | The application identifier. |

### Return Values

None.

### Throws

None.

## setPassword

### Description

Sets the password associated with the terminal.

### Syntax

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

### Parameters

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

### Return Values

None.

### Throws

None.

### Additional Information

Invoking this method automatically flags the terminal as an extended type of terminal. The password will not be picked up until another send is completed or the terminal is connected.

## setPort

### Description

Sets the port number necessary to communicate with the Gateway.

### Syntax

```
public void setPort(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | The Gateway port number. |

### Return Values

None.

### Throws

None.

## setProgram

### Description

Sets the CICS program identity.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The name of the CICS program. |

### Return Values

None.

### Throws

None.

## setServer

### Description

Sets the server identity on which the CICS program is running.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The name of the server on which CICS resides. |

### Return Values

None.

### Throws

None.

## setSslClass

### Description

Sets the identity of the SSL class.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The SSL class name. |

### Return Values

None.

### Throws

None.

## setSslPassword

### Description

Sets the password required to access SSL information.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The SSL password. |

### Return Values

None.

### Throws

None.

## setTraceDumpOffset

### Description

Sets the offset value for trace dumping.

### Syntax

```
public void setTraceDumpOffset(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | The offset amount. |

### Return Values

None.

### Throws

None.

## setTraceFilename

### Description

Sets the name of the trace file to be used.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The name of the trace file. |

### Return Values

None.

### Throws

None.

## setTraceLevel

### Description

Sets the debugging trace level.

### Syntax

```
public void setTraceLevel(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | The trace level to be set. |

### Return Values

None.

### Throws

None.

## setTraceTiming

### Description

Sets the debugging trace timing.

### Syntax

```
public void setTraceTiming(boolean b)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| b | boolean | true sets trace timing to On. |

### Return Values

None.

### Throws

None.

## setTraceTruncationSize

### Description

Sets the trace truncation size.

### Syntax

```
public void setTraceTruncationSize(int i)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| i | integer | The truncation size to be set. |

### Return Values

None.

### Throws

None.

## setTransId

### Description

Sets the CICS transaction id.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The CICS transaction id. |

### Return Values

None.

### Throws

None.

## setUrl

### Description

Sets the URL to the Transaction Gateway.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The URL for the Transaction Gateway. |

### Return Values

None.

### Throws

None.

## setUserId

### Description

Sets the used ID associated with the terminal.

### Syntax

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

### Parameters

| Name | Type | Description |
|---|---|---|
| s | java.lang.String | The terminal user ID. |

### Return Values

None.

### Throws

None.

## terminate

### Description

Terminates the ETD.

### Syntax

```
public void terminate()
```

### Parameters

None

### Return Values

None.

### Throws

**com.stc.common.collabService.CollabConnException**
Indicating a connection error.

### Additional Information

Overrides terminate in class com.stc.jcsre.SimpleETDImpl

## toPackedDecimal

### Description

Converts a string to a PackedDecimal object.

### Syntax

```
public static com.stc.eways.cics.PackedDecimal
toPackedDecimal(java.lang.String number, int intSize, int decSize)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| number | java.lang.String | Decimal String representation to be converted |
| intSize | integer | The number of integer digits in the resulting object. |
| decSize | integer | The number of decimal digits in the resulting object. |

### Return Values

**com.stc.eways.cics.PacedDecimal**

### Throws

**java.lang.NumberFormatException**

## toZoned

### Description

Converts a number to zoned decimal (Cobol PIC S9).

### Syntax

```
public static byte[] toZoned(java.lang.String number)

public static byte[] toZoned(java.lang.String number,
java.lang.String encoding)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| number | java.lang.String | The number to be converted |
| encoding | java.lang.String | The encryption type. |

### Return Values

**byte []**
Returns a byte array containing the new zoned data.

### Throws

**java.lang.NumberFormatException**
Indicating an error occurred as a result of a numeric format exception.

## zonedToString

### Description

Converts zoned decimal (Cobol PIC s9) byte array to a string.

### Syntax

```
public static java.lang.String zonedToString(byte[] zoned)

public static java.lang.String zonedToString(byte[] zoned,
java.lang.String encoding)
```

### Parameters

| Name | Type | Description |
|---|---|---|
| zoned | byte[] | Description |
| encoding | java.lang.String | The character encoding type (ASCII, cp500 for EBCDIC, etc.) of the number to be converted. |

***Note:*** *Methods that include an "encoding" parameter should specify encoding as "ISO-8859-1" when the e*Way is run on the OS/390 platform.*

### Return Values

**java.lang.String**
Returns the new converted string.

### Throws

**java.lang.NumberFormatException**

# Index