## Contents

### Chapter 1

**Introduction**
- Overview
  - Information Management System (IMS)
  - The e*Way Intelligent Adapter for IMS
- Intended Reader
- Components

**Supported Operating Systems**

**System Requirements**
- Client Components

**External System Requirements**
- Hardware Requirements
- Software Requirements
- Software Requirements for IMS Connect and ITOC:

**Supporting Documents**

### Chapter 2

**Installation**

**Windows NT or Windows 2000**
- Pre-installation
- Installation Procedure

**UNIX**
- Pre-installation
- Installation Procedure

**OS/390 V2R10**

**Files/Directories Created by the Installation**

### Chapter 3

**Configuration**

**Multi-Mode e*Way Configuration**
- Creating a Multi-Mode e*Way
- Multi-Mode e*Way Configuration Parameters
Contents

JVM Settings 17
  JNI DLL Absolute Pathname 17
  CLASSPATH Prepend 18
  CLASSPATH Override 18
  CLASSPATH Append From Environment Variable 19
  Initial Heap Size 19
  Maximum Heap Size 19
  Maximum Stack Size for Native Threads 19
  Maximum Stack Size for JVM Threads 20
  Disable JIT 20
  Remote debugging port number 20
  Suspend option for debugging 20
General Settings 20
  Rollback Wait Interval 21

e*Way Connection Configuration 21
  Creating an e*Way Connection 21
  e*Way Connection Configuration Parameters 22
Connector 22
  Type 23
  Class 23
  Property.Tag 23
  Connection Establishment Mode 23
  Connection Inactivity Timeout 24
  Connection Verification Interval 24
TCP/IP Configuration 24
  Server 25
  Port 25
IRM Header 26
  IRM_LEN 26
  IRM_ID 26
  IRM_F5 (Flow Control) 27
  IRM_TIMER 27
  IRM_SOCT 27
  IRM_CLIENTID 28
  IRM_F1 (MFS MOD Names) 28
  IRM_F2 (COMMIT MODE) 28
  IRM_F3 (Sync Level) 29
  IRM_F4 (ACK/NAK/ Response) 29
  IRM_TRNCOD_SRC 30
  IRM_TRNCOD 30
  IRM_DESTID 30
  IRM_LTERM 30
  IRM_RACF_USERID 30
  IRM_RACF_PW 31
  IRM_RACF_GRNAME 31

Chapter 4

Implementation 32

IMS Sample Implementation 32
Importing the Sample Schema 33
Chapter 5:

Java Methods

The IMSClientETD Class

Methods of the IMSClientETD Class
connect
disconnect
getBandrs
getDatastoreID
getLtermName
getMessage
getPassword
getPort
getRacfGroupName
getRacfUserID
getReply
getServer
getTranCode
getTranCodeSrc
IMSSRequest
IMSSRequestWithMultiSegs
Contents

isConnected 64
setBandrs 65
setDatastoreID 65
setLtermName 66
setMessage 66
setPassword 66
setPort 67
setRacfGroupName 67
setRacfUserID 68
setServer 68
setTranCode 69
setTranCodeSrc 69
terminate 70

Index 71
Chapter 1

Introduction

This user’s guide explains how to install and configure the e*Way Intelligent Adapter for IMS (Java version).

1.1 Overview

Information Management System (IMS)

IBM’s IMS (Information Management System), is a database and transaction management system that provides an interface for users to access information in various databases via on-line transactions. IMS/TM (Transaction Manager), a message-based transaction processor, handles the execution of specific business application programs. IMS/DB (Database) is an entirely separate component providing access to the IMS hierarchical database from applications running under the IMS Transaction Manager, as well as CICS transaction monitor, and OS/390 batch jobs.

The e*Way Intelligent Adapter for IMS

The IMS e*Way enables the e*Gate Integrator to connect with IBM’s IMS/TM mainframe applications through IBM’s IMS Connect, or its predecessor, IMS TCP/IP OTMA Connection (ITOC).

Figure 1  IMS e*Way and the IMS Environment.
The e*Way provides access to the Input and Output Descriptors (MID/MOD) of the IMS applications without requiring changes to the application. By capturing the field contents before screen formatting, the e*Way is not affected by cosmetic changes to the application’s screen design.

The e*Way includes the IMS MFS Wizard conversion utility to facilitate the creation of input and output Event Type Definitions (ETDs) from IMS Message Format Service (MFS) files.

The implementation of the IMS e*Way is in accordance with the IBM Documents IMS TCP/IP OTMA Connection User’s Guide and Reference, Version 2.1.3, and IMS Connect Guide and Reference, Version 1). These documents describe the OTMA protocol and contain important prerequisite information for the configuration of ITOC and IMS Connect on the mainframe.

A sample schema for the Java enabled IMS e*Way is included on the installation CD-ROM which demonstrates how a non-conversational scenario (simple send/response) is managed.

1.1.1. Intended Reader

The reader of this guide is presumed:

- To be a developer or system administrator with the responsibility of maintaining the e*Gate system.
- To have working knowledge of Windows, or UNIX, and OS/390 operations and administration.
- To be familiar with IMS transactions and OTMA protocol.
- To be familiar with Windows-style GUI operations.

1.1.2. Components

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

- **Jar files** containing the logic required by the e*Way to gain access to IMS.
- The **IMS e*Way Connection** which provides access to information necessary for connecting to specified external connections.
- The **IMSGlobalETD.xsc** which 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.
- The Java **IMS MFS Wizard** that generates Event Type Definitions (ETDs) that map input and output message segments at the field level.

*Note:* The IBM MFS Wizard is not supported for the IMS e*Way running on the OS/390 platform.

A complete list of installed files appears in Table 1 on page 15.
1.2 Supported Operating Systems

The IMS e*Way is available on the following operating systems:
- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Windows NT 4 SP6a
- Solaris 2.6, 7, and 8
- AIX 4.3.3 and 5.1
- HP-UX 11.0 and HP-UX 11i
- Red Hat Linux 6.2
- OS/390 V2, R10

1.3 System Requirements

**Important:** Open and review the Readme.txt for the IMS e*Way regarding any additional requirements prior to installation. The Readme.txt is located on the Installation CD_ROM at setup\addons\ewims.

To use the IMS e*Way, you need the following:
- An e*Gate Participating Host, version 4.5.1 or later.

**Note:** Subcollaborations and connection management are only supported for the IMS e*Way in e*Gate 4.5.2 and above.

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

1.3.1 Client Components

The client components of IMS have their own requirements; see the subject system’s documentation for details.
1.4 External System Requirements

Note: This document uses the term RACF when referring to RACF (Resource Access Control Facility) or equivalent products.

Hardware Requirements

- A host processor capable of running the versions of the operating system, IMS, TCP/IP, and RACF detailed below.

Software Requirements

- IMS TCP/IP OTMA Connection (ITOC) version 2.1 and above or IMS Connect, Version 1 and above.

Note: IMS Connect is an SMP/E installable and maintainable priced product that replaces the cost-free ITOC. IBM recommends that ITOC only be used for “proof of concept” purposes, since support for ITOC was withdrawn on March 1, 2001. ITOC is no longer available for download.

Software Requirements for IMS Connect and ITOC:

- OS/390 Version 2 Release 6 configurations, or subsequent versions, releases, and modification levels.
- IMS Version 5, IMS Version 6, and/or IMS Version 7, with the required maintenance APARs applied for each version. See IMS Connect Guide and Reference, chapter 3, “IMS Connect and IMS Coexistence” for more information about IMS and IMS Connect coexistence.
- TCP/IP (Version 3.2, or Version 3.4 or later), APAR PQ13154, and APAR PQ38814.
- Resource Access Control Facility (RACF) Version 1.9.2 or later, or equivalent product.

Important: IMS Connect and ITOC do not support TCP/IP Version 3.3.
1.5 Supporting Documents

The following SeeBeyond documents are designed to work in conjunction with the *e*Way Intelligent Adapter for IMS User’s Guide and provide additional information that may prove useful:

- Creating an End-to-end Scenario with *e*Gate Integrator
- *e*Gate Integrator Collaboration Services Reference Guide
- *e*Gate Integrator Installation Guide
- *e*Gate Integrator Intelligent Queue Services Reference Guide
- *e*Gate Integrator System Administration and Operations Guide
- *e*Gate Integrator User’s Guide
- Standard *e*Way Intelligent Adapters User’s Guide
- Readme.txt file on the *e*Gate Installation CD-ROM.
Chapter 2

Installation

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

Windows NT or Windows 2000 on page 12
UNIX on page 14
OS/390 V2R10 on page 14
Files/Directories Created by the Installation on page 15

2.1 Windows NT or 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.
- Open and review the Readme.txt for the IMS eWay regarding any additional requirements prior to installation. The Readme.txt is located on the Installation CD_ROM at setup\addons\ewims.

2.1.2. Installation Procedure

To install the IMS e*Way on a Windows 2000 or NT system

1. Log in as an Administrator on the workstation on which the e*Way will be installed.
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 Select Sub-components dialog box appears.

10. Select the IMS e*Way as shown in Figure 2. Click the continue button to return to the Select Components dialog box, then click Next.

![Figure 2](IMS e*Way Addon)

11. Follow the rest of the on-screen instructions to install the IMS 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 IMS 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:** Install the e*Way files in the suggested client installation directory. The installation utility detects the appropriate installation directory. Do not change the suggested “installation directory” settings unless directed to do so by your SeeBeyond support representative.

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

**Note:** Once you have installed and configured this e*Way, 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 directions for OS/390 V2R10 can be found in the e*Gate Integrator Installation Guide.
2.4 Files/Directories Created by the Installation

The IMS e*Way installation process will install the following files (see Table 1) 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

<table>
<thead>
<tr>
<th>e*Gate Directory</th>
<th>File(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>stcewims.ctl</td>
<td></td>
</tr>
<tr>
<td>/bin/java/</td>
<td>stcjintegra.jar</td>
</tr>
<tr>
<td></td>
<td>xerces.jar</td>
</tr>
<tr>
<td>/classes/</td>
<td>stcutil.jar</td>
</tr>
<tr>
<td></td>
<td>stcexception.jar</td>
</tr>
<tr>
<td></td>
<td>stcims.jar</td>
</tr>
<tr>
<td>/configs/imsclientetd/</td>
<td>IMSClientETD.def</td>
</tr>
<tr>
<td>/etd/</td>
<td>IMSClientETD.ctl</td>
</tr>
<tr>
<td>/etd/imsclientetd/</td>
<td>IMSClientETD.xsc</td>
</tr>
</tbody>
</table>
Configuration

This chapter describes how to configure the following components of the IMS e*Way.

- Multi-Mode e*Way Configuration on page 16
- e*Way Connection Configuration on page 21

3.1 Multi-Mode e*Way Configuration

A Multi-Mode e*Way is a multi-threaded component used to route and transform data within e*Gate. Unlike traditional e*Ways, Multi-Mode e*Ways can use multiple simultaneous e*Way Connections to communicate with several external systems, as well as IQs or JMS IQ Managers. The following describes how to create and configure the Multi-Mode e*Way component for the IMS e*Way. Multi-Mode e*Way properties are set using the Enterprise Manager.

3.1.1 Creating a Multi-Mode e*Way

1. Select the Navigator’s Components tab.
2. Open the host on which you want to create the e*Way.
3. On the Palette, click on the icon to create a new e*Way.
4. Enter the name of the new e*Way, then click OK.
5. Select the new e*Way component, right-click, and select Properties. The e*Way Properties dialog box opens.
6. The Executable File field defaults to stceway.exe. (stceway.exe is located in the “bin\” directory).
7. Type any additional command line arguments that the e*Way may require in the Additional Command Line Arguments field, taking care to insert them at the end of the existing command-line string. Be careful not to change any of the default arguments unless you have specific need to do so.
8. Click New under the Configuration File field to create a new configuration file, Find to select an existing configuration file, or Edit to edit the currently selected file. The Editor opens to edit settings for the Multi-Mode e*Way. The Multi-Mode e*Way Configuration Editor opens. The following section provides more information on these parameters.
After selecting the desired parameters, **Save** the configuration file and select **Promote to Run Time**. Click **OK** to close the e*Way Properties Window.

For more information on Multi-Mode e*Way settings and properties see the *e*Gate Integrator User’s Guide, the Standard e*Way Intelligent Adapter User’s Guide or consult the e*Way Editor’s online Help.

### 3.1.2. Multi-Mode e*Way Configuration Parameters

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

- JVM Settings on page 17
- General Settings on page 20

### 3.1.3. JVM Settings

The JVM Settings control basic Java Virtual Machine settings. The JVM Settings section contains the following parameters:

- JNI DLL Absolute Pathname on page 17
- CLASSPATH Prepend on page 18
- CLASSPATH Override on page 18
- CLASSPATH Append From Environment Variable on page 19
- Initial Heap Size on page 19
- Maximum Heap Size on page 19
- Maximum Stack Size for Native Threads on page 19
- Maximum Stack Size for JVM Threads on page 20
- Disable JIT on page 20
- Remote debugging port number on page 20
- Suspend option for debugging on page 20

#### 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:
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.

<table>
<thead>
<tr>
<th>OS</th>
<th>Java 2 JNI DLL Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows NT 4.0, Windows2000</td>
<td>jvm.dll</td>
</tr>
<tr>
<td>Solaris 2.6, 7, or 8</td>
<td>libjvm.so</td>
</tr>
<tr>
<td>OS/390</td>
<td>libjvm.so</td>
</tr>
<tr>
<td>HP-UX</td>
<td>libjvm.sl</td>
</tr>
<tr>
<td>AIX</td>
<td>libjvm.a</td>
</tr>
<tr>
<td>Red Hat Linux 6.2</td>
<td>libjvm.so</td>
</tr>
</tbody>
</table>
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 No.

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.

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.

Remote debugging port number

Description
Specifies the port number for the remote debugging of the JVM.

Required Values
An integer between 2000 and 65536.

Suspend option for debugging

Description
Specifies whether the option for debugging will be enabled or suspended upon JVM startup.

Required Values
YES or NO.

3.1.4. General Settings

General Settings controls the period of time the workslice waits before it re-posts a message once a rollback has occurred. The General Settings section contains the following parameter:

- Rollback Wait Interval on page 21
Rollback Wait Interval

Description
Specifies the period of time in milliseconds, that the workslice waits before it re-posts a message once a rollback has occurred.

Required Values
An integer between 0 and 9999999.

3.2 e*Way Connection Configuration

e*Way Connections are the encoding of access information for specific external connections. The e*Way Connection configuration file contains the parameters necessary for connecting with a specific external system. e*Way Connection parameters are set using the Enterprise Manager.

3.2.1. Creating an e*Way Connection

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 and click OK.
5 From the **e*Way Connection Type** drop-down box, select **IMSClientETD**.

6 Enter the **Event Type “get” interval** in the dialog box provided. The configured default is 10000 milliseconds.

7 Click **New** under the **e*Way Connection Configuration File** field to create a new configuration file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file. The e*Way Connection Configuration Editor opens. The following section provides more information on these e*Way Connection parameters.

8 After selecting the desired parameters, **Save** the configuration file and select **Promote to Run Time**. Click **OK** to close the e*Way Connection Properties Window.

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

### 3.2.2. e*Way Connection Configuration Parameters

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

- **Connector** on page 22
- **TCPIP Configuration** on page 24
- **IRM Header** on page 26

### 3.2.3. Connector

The Connector section contains the following top level parameters:
- **Type** on page 23
- **Class** on page 23
- **Property.Tag** on page 23
- **Connection Establishment Mode** on page 23
- **Connection Inactivity Timeout** on page 24
- **Connection Verification Interval** on page 24

**Type**

**Description**

Specifies the connector type.

**Required Values**

String-set. The value is always **IMSClientETD** by default for IMSClientETD connections.

**Class**

**Description**

Specifies the class name of the ETD connector object.

**Required Values**

String-set. A valid package name. The default is `com.stc.eways.ims.IMSClientETDConnector`.

**Property.Tag**

**Description**

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

**Required Values**

String-set. A valid data source package name.

**Connection Establishment Mode**

**Description**

Specifies how the connection with the external system is initiated and ended.

- **Automatic**: indicates that the connection is automatically established when the collaboration is started, and it keeps the connection alive as needed.

- **OnDemand**: indicates that the connection will be established on demand as business rules requiring a connection to the external system are performed. The connection will be closed after the methods are completed.


- **Manual**: indicates that the user will explicitly call the connection connect and disconnect methods in their collaboration as business rules.

**Required Values**


**Note**: This parameter is ignored when run on e*Gate 4.5.1.

### Connection Inactivity Timeout

**Description**

Specifies the timeout (in milliseconds) for the **Automatic** option for the Connection Establishment Mode parameter.

Specifies the minimum period of time (in milliseconds) between checks for the connection status with the database server.

- If the connection to the server is detected to be down during verification, the user Collaboration’s **onConnectionDown** method is called.
- If the connection comes from a previous connection error, the user Collaboration’s **onConnectionUp** method is called.
- If no value is specified, 60000 ms (milliseconds) is used by default.

**Required Values**

String-set. A valid number (number of milliseconds).

**Note**: This parameter is ignored when run on e*Gate 4.5.1.

### Connection Verification Interval

**Description**

Specifies the timeout (in milliseconds) for the **Automatic** option for the Connection Establishment Mode parameter.

- If the value is left blank or set to 0 the connection will not timeout (be brought down) due to inactivity. The connection is always kept alive. If the connection goes down, re-establishing connection will be attempted automatically.
- If a non-zero value is specified, the connection manager attempts to monitor for inactivity and the connection is ended when the specified timeout value is reached.

**Required Values**

String-set. A valid number (number of milliseconds).

This parameter is ignored when run on e*Gate 4.5.1.

#### 3.2.4. TCPIP Configuration

The TCPIP Configuration section contains information for connecting to Portal Infranet. This section contains the following top level parameters:
- **Server** on page 25
- **Port** on page 25

**Server**

**Description**

Specifies the name of the server host. This parameter is mandatory.

**Required Values**

String-set. The computer name of the server.

**Port**

**Description**

Specifies the port that IMS Connect is listening on. This parameter is mandatory.

**Required Values**

String-set. The number for the port on which IMS Connect is listening.
3.2.5. **IRM Header**

The IRM (IMS Request Message) Header section contains the following top level parameters:

- **IRM_LEN** on page 26
- **IRM_ID** on page 26
- **IRM_F5 (Flow Control)** on page 27
- **IRM_TIMER** on page 27
- **IRM_SOCT** on page 27
- **IRM_CLIENTID** on page 28
- **IRM_F1 (MFS MOD Names)** on page 28
- **IRM_F2 (COMMIT MODE)** on page 28
- **IRM_F3 (Sync Level)** on page 29
- **IRM_F4 (ACK/NAK/ Response)** on page 29
- **IRM_TRNCOD_SRC** on page 30
- **IRM_TRNCOD** on page 30
- **IRM_DESTID** on page 30
- **IRM_LTERM** on page 30
- **IRM_RACF_USERID** on page 30
- **IRM_RACF_PW** on page 31
- **IRM_RACF_GRNAME** on page 31
- **IRM_RACF_USERID** on page 30

**Note:** For a full description of the IRM header, see IBM’s IMS Connect Guide and Reference (SC27-0946-00).

**IRM_LEN**

**Description**

Specifies the length of the IRM structure. The minimum size of the IRM for user written exits is 32. HWSIMSO0 and HWSSMPL0 have a minimum IRM length of 80.

**Required Values**

String-set. A valid IRM structure length. The configured default is 80.

**IRM_ID**

**Description**

Specifies the identifier (character string) of the user exit that is to be driven after the complete message has been received. The IMS Connect-supplied user message exits reserve and use these IDs:

- *IRMREQ* for HWSIMSO0
- *SAMPLE* for HWSSMPL0

**Required Values**

String-set. One of the three identifier character strings. The configured default is *SAMPLE*.
IRM_F5 (Flow Control)

Description

Specifies Flow Control properties.

**Note:** SeeBeyond recommends using the default value **No_Auto_Flow.** Contact SeeBeyond Support before using any value other than **No_Auto_Flow.**

- **Client_Headers:** OTMA headers are built by the client.
- **Client_Translation:** Translation is done by the client.
- **Single_Message:** Only one message is returned on receive following the resumetpipe.
- **No_Auto_Flow:** No auto flow of messages (see meaning for No_Auto_Flow_Out).
- **Auto_Flow_Out:** Auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a large value. Use only if the client is a dedicated output client.
- **No_Auto_Flow_Out:** No auto flow of messages. All current messages will be returned, one at a time, and the last receive waits for the next message for the IRM_TIMER value. Set the IRM_TIMER to be a small value. Use only if the client is a dedicated output client. This value is similar to Auto_Flow_Out, as described above, except that the IRM_TIMER will cause the last receive to terminate.

Required Values

The recommended setting is **No_Auto_Flow** (default).

IRM_TIMER

Description

Specifies the time delay for the receive to the datastore after an ACK.

- **.25 SEC:** .25 seconds.
- **No_Wait:** Does not set the timer (no wait occurs).
- **Block:** The receive waits indefinitely.

Required Values

Select one of the three options. The configured default is **.25 SEC.**

IRM_SOCT

Description

Specifies the socket connection type.

- **Transaction:** Transaction socket. The socket connection lasts across a single transaction.
- **Persistent**: Persistent socket. The socket connection lasts across multiple transactions.
- **NonPersistent**: Non-persistent socket. The socket connection lasts for a single exchange consisting of one input and one output.

  **Note**: This socket type is not recommended when implementing conversational transactions due to the multiple connects and disconnects that would occur as a result.

**Required Values**

Select one of the three options. The configured default is Transaction.

**IRM_CLIENTID**

**Description**

Specifies the name of the client ID (character string) that is used by IMS Connect. If this string is not supplied from the client, then the user exit must generate it. The client ID is returned to IMS Connect from the exit in the EXIT PARMLIST field, EXPREA_CLID.

**Required Values**

String-set. A valid client ID used by IMS Connect. The configured default is STCEWAY.

**IRM_F1 (MFS MOD Names)**

**Description**

Specifies whether the MFS Message Output Descriptor (MOD) should be returned as part of the output.

- **MFS**: The user requests MFS MOD name to be returned.
- **NO_MFS**: The user requests no MFS MOD name to be returned.

When MFS is specified, a Request Mod Message (RMM) will be returned as the first structure of the output message. This structure will have an ID of *REQMOD* followed by the MFS MOD name. For details, see IBM’s IMS Connect Guide and Reference, (SC27-0946-00), page 59.

**Required Values**

Select one of the two options. The configured default is NO_MFS.

**IRM_F2 (COMMIT MODE)**

**Description**

Specifies the **Commit Mode**. If this value is not supplied from the client, the user exit must use a default value.

- **COMMIT_MODE_0**:
- **COMMIT_MODE_1**:
Required Values

Select one of the two options. The configured default is COMMIT_MODE_0.

IRM_F3 (Sync Level)

Description

Specifies the Sync Level. If this value is not supplied from the client, the user exit must use a default value. I

- **SYNC_LEVEL_CONFIRM**: Must be used when the IRM_F2 parameter (commit mode) is set to COMMIT_MODE_0.

- **SYNC_LEVEL_NONE**:

Required Values

Select one of the two options. If COMMIT_MODE_0 (IRM_F2) is used, the Sync level must be SYNC_LEVEL_CONFIRM. The configured default is SYNC_LEVEL_CONFIRM.

IRM_F4 (ACK/NAK/ Response)

Description

Specifies the ACK/NAK/response expression sent to IMS Connect and forwarded to IMS. When the value is received and passed to the user exit, the exit builds the appropriate OTMA structure and returns it to IMS Connect. The ACK/NAK/DEALLOCATE/RESUME [A/N/D/R] must be sent to IMS Connect with no data element.

- **ACK**: Positive acknowledgment, used in response to a message sent to the client where the SYNC level is CONFIRM (SYNC_LEVEL_CONFIRM).

- **NACK**: Negative acknowledgment, used in response to a message sent to the client where the SYNC level is CONFIRM (SYNC_LEVEL_CONFIRM).

- **DEALLOCATE**: Deallocate connection is used to terminate rather than complete the conversation.

- **RESUME**: Resume TPIPE is used to request Asynchronous output data from IMS. (Not yet implemented)

- **SENDONLY**: Send only, used for a non-response transaction and for sending data to IMS.

- **NO_ACK**: No request for acknowledgment or deallocation. When a response mode transaction or conversational transaction is being sent to IMS Connect, IRM_F4 should be set to NO_ACK.

Required Values

Select one of the six options. The configured default is NO_ACK.
IRM_TRNCOD_SRC

Description
Specifies where the transaction code is taken.
- **CFG**: The transaction code is to be taken from the configuration file.
- **MESSAGE**: The transaction code is the first 8 bytes of the message.

Required Values
Select one of the two options. The configured default is CFG.

IRM_TRNCOD

Description
Specifies the default IMS transaction code.

Required Values
String-set. A valid transaction code.

IRM_DESTID

Description
Specifies the Datastore name (IMS destination ID). This field must be specified by the client.

Required Values
String-set. A valid Datastore name/IMS destination ID (character string).

IRM_LTERM

Description
Specifies the IMS LTERM override. This field can be set to a valid name or to blanks.

Required Values
String-set. A valid LTERM name or blanks.

IRM_RACF_USERID

Description
Specifies the RACF User ID. The client must provide the RACF user ID if RACF is to be used.

Required Values
String-set. A valid RACF user ID.
IRM_RACF_PW

Description

Specifies the RACF PASSTICKET. The client must provide the RACF PASSTICKET if RACF is to be used.

Required Values

String-encrypt. A valid RACF PASSTICKET.

IRM_RACF_GRNAME

Description

Specifies the RACF Group Name. The client must provide the RACF group name if RACF is to be used.

Required Values

String-set. A valid RACF group name.
Implementation

This chapter presents information pertinent to implementing the Java-enabled IMS e*Way in a production environment. It provides an introduction to the e*Way components by demonstrating an implementation of the sample schemas. Two samples are provided for the Java enabled IMS e*Way; Java_IMS.zip that demonstrates the e*Way on all supported platforms other than OS/390, and Java_IMS_os390.zip which is specifically for the OS/390 platform.

The following assumptions apply to this implementation:

- The IMS e*Way has been successfully installed.
- The executable and the configuration files have been appropriately assigned.
- All necessary .jar files are accessible.

4.1 IMS 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 IMS 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 and outbound e*Ways using stcewfile.exe.
  - The Multi-Mode e*Way component as described in Chapter 3.
  - The e*Way Connection as described in Chapter 3.
  - Event Type Definitions used to package data to be exchanged with the external system.
  - Collaboration Rules to process Events.
  - 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.
4.2 Importing the Sample Schema

Sample schemas for the Java enabled IMS e*Way is available at ..\samples\ewims.. on the Installation CD-ROM.

To import a sample schema 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 are then prompted to select a schema. Click New. The New Schema dialog box opens. (Schemas can also be imported or opened from the e*Gate File menu by selecting New Schema or Open Schema.)
4. Enter a name for the new Schema. In this case, enter IMS_Sample, or any name as desired.
5. To import the sample schema select Create from Export, and use Find to locate and select the sample .zip file on the CD-ROM. For Windows or UNIX operating systems select Java.IMS.zip. Click Open. For OS/390 operating systems select Java.IMS_os390.zip.

The e*Gate Enterprise Manager opens to the new schema. You are now ready to make any configuration changes that may be necessary for this sample schema to run on your specific system.

The following pages provide a description of how the sample schemas’s e*Way components are created and configured manually. For information on implementing the OS/390 sample schema see Sample Schema Implementation for OS/390 on page 55.

4.3 Creating and Configuring the e*Ways

Component e*Ways connect with external systems to poll or send data. They also transform and route data. Multi-Mode e*Ways are used to run Java Collaborations that utilize e*Way Connections to send and receive Events to and from multiple external systems. The following sections provide directions for creating the component e*Way used by the sample schema.
4.3.1. Creating the Inbound e*Way (Feeder)

1. Select the Navigator's Components tab.
2. Open the host on which you will 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 “ew_Request_in”), 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 stcewwfile.exe as the executable file.

![Inbound e*Way Properties dialog box](image)

8. Under the Configuration File field, click the New (or Edit for saved e*Ways) button. The Configuration Editor opens. Select the following settings for this configuration file:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Settings (unless otherwise stated, leave settings as default)</td>
<td></td>
</tr>
<tr>
<td>AllowIncoming</td>
<td>YES</td>
</tr>
<tr>
<td>AllowOutgoing</td>
<td>NO</td>
</tr>
<tr>
<td>Outbound Settings</td>
<td>Default</td>
</tr>
<tr>
<td>Poller Inbound Settings</td>
<td></td>
</tr>
<tr>
<td>PollDirectory</td>
<td>C:\INDATA (input file folder)</td>
</tr>
</tbody>
</table>
When the desired parameters have been entered, save the configuration file as `ew_req_in.cfg`.

Click File, Promote to Run Time. This will close the Configuration Editor.

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.

Select OK to close the e*Way Properties window.

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

1 On the palette, click the Create a New e*Way button to create another e*Way.

2 Enter the name of the new e*Way (in this case “ew_Response_out”), then click OK.

3 Select the new e*Way, right-click and select Properties. Click Find under the Executable File field and select `stcewfile.exe` as the executable file.

4 Under the Configuration File field, click the New button. The Configuration Editor opens. Select the following settings for this configuration file.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InputFileExtension</code></td>
<td>*.fin (input file extension)</td>
</tr>
<tr>
<td><code>PollMilliseconds</code></td>
<td>1000</td>
</tr>
<tr>
<td><code>Remove EOL</code></td>
<td>YES</td>
</tr>
<tr>
<td><code>MultipleRecordsPerFile</code></td>
<td>No</td>
</tr>
<tr>
<td><code>MaxBytesPerLine</code></td>
<td>4096</td>
</tr>
<tr>
<td><code>BytesPerLineIsFixed</code></td>
<td>NO</td>
</tr>
<tr>
<td><code>File Records Per eGate Event</code></td>
<td>1</td>
</tr>
<tr>
<td><strong>Performance Testing</strong></td>
<td>Default</td>
</tr>
</tbody>
</table>

9 When the desired parameters have been entered, save the configuration file as `ew_req_in.cfg`.

10 Click File, Promote to Run Time. This will close the Configuration Editor.

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.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Settings (unless otherwise stated, leave settings as default)</strong></td>
<td></td>
</tr>
<tr>
<td>AllowIncoming</td>
<td>NO</td>
</tr>
<tr>
<td>AllowOutgoing</td>
<td>YES</td>
</tr>
<tr>
<td><strong>Outbound Settings</strong></td>
<td></td>
</tr>
<tr>
<td>OutputDirectory</td>
<td>C:\DATA</td>
</tr>
</tbody>
</table>
5 Save the .cfg file as `ew_resp_out.cfg`, and click **Promote to Run Time** to move the file to the runtime environment and close the Configuration Editor.

6 In the e*Way - Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for the e*Way.

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

### 4.3.3. Creating the Multi-Mode e*Way

1 Select the **Control Broker** that will manage the new e*Way.

2 On the palette, click the **Create a New e*Way** button.

3 Enter the name of the new e*Way (in this case, “ew_IMS_interface”), then click **OK**.

4 Right-click the new e*Way and select **Properties** to edit its properties.

5 When the e*Way Properties window opens, click the **Find** button beneath the **Executable File** field, and confirm that `stceway.exe` is the executable file (`stceway.exe` is the default).

6 To edit the JVM Settings, select **New** (or **Edit** if you are editing the existing .cfg file) under Configuration file. The Multi-Mode e*Way Configuration Editor opens.

   See **Multi-Mode e*Way Configuration Parameters** on page 17 for details on the parameters associated with the Multi-Mode e*Way.

7 Save the .cfg file (`ew_ims_interface.cfg`), and **Promote to Run Time**.

8 In the e*Way Properties window, use the **Startup**, **Advanced**, and **Security** tabs to modify the default settings for each.

9 Click **OK** to close e*Way Properties window.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutputFileName</td>
<td>output%d.dat</td>
</tr>
<tr>
<td>MultipleRecordsPerFile</td>
<td>YES</td>
</tr>
<tr>
<td>MaxRecordsPerFile</td>
<td>10000</td>
</tr>
<tr>
<td>AddEOL</td>
<td>YES</td>
</tr>
<tr>
<td>Poller Inbound Settings</td>
<td>Default</td>
</tr>
<tr>
<td>Performance Testing</td>
<td>Default</td>
</tr>
</tbody>
</table>
### 4.4 Creating the e*Way Connection

An e*Way Connection is the encoding of access information for an external connection. The e*Way Connection configuration file contains the settings necessary for connecting with IMS.

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, “ecp_stc_ims”), then click OK.
4. Double-click the new e*Way Connection to edit its properties.
5. The e*Way Connection Properties window opens. Select IMSClientETD 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. (For the purpose of this sample, set the “get” interval to 100.)
7. Under e*Way Connection Configuration File, click the New (Edit) button.
8. The e*Way Connection Editor opens. Select the following parameters listed in Table 4. For more information on the IMS e*Way Connection parameters, see e*Way Connection Configuration Parameters on page 22.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>connector (unless otherwise stated, leave settings as default)</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>IMSClientETD</td>
</tr>
<tr>
<td>class</td>
<td>com.stc.eways.ims.IMSClientETDConnector</td>
</tr>
<tr>
<td>Connection Establishment Mode</td>
<td>Manual</td>
</tr>
<tr>
<td>TCPIP Configuration</td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td>server name</td>
</tr>
<tr>
<td>Port</td>
<td>port number</td>
</tr>
<tr>
<td>IRM Header</td>
<td></td>
</tr>
<tr>
<td>IRM_Len</td>
<td>80</td>
</tr>
<tr>
<td>IRM_ID</td>
<td><em>SAMPLE</em></td>
</tr>
<tr>
<td>IRM_CLIENTID</td>
<td>STCEWAY</td>
</tr>
<tr>
<td>IRM_TRNCOD</td>
<td>PART</td>
</tr>
<tr>
<td>IRM_DESTID</td>
<td>IVP1</td>
</tr>
<tr>
<td>IRM_RACF_USERID</td>
<td>USER99</td>
</tr>
<tr>
<td>IRM_RACF_PW</td>
<td>PASSWORD</td>
</tr>
</tbody>
</table>
### 4.5 Event Types

Each packet of data within e*Gate is referred to as an Event. Event Types are data labels that allow e*Gate to process and route specific Events differently. e*Gate uses Event Type Definitions (.xsc files) to parse, validate, and (if necessary) transform Events.

#### 4.5.1. Creating an Event Type Using the Custom ETD Wizard

The following procedure shows how to create an ETD using the Custom ETD Wizard.

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 to create a new Event Type. Enter a name for the Event Type, then click OK.
3. Double-click the new event type to edit its properties.
4. When the Properties window opens, click the New button. The ETD Editor opens.
5. Select New from the File menu on Task Manager. The New Event Type Definition window opens.
6. Select the appropriate wizard. (For this Event Type, select Custom ETD Wizard. See Figure 5.)

![Figure 5 New Event Type Definition - Custom ETD Wizard](image)

7. Enter a root node name. (For this example type **GenericEvent**.)

---

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRM_RACF_GRNAME</td>
<td>SYS1</td>
</tr>
</tbody>
</table>
8 Enter a package name where the ETD Editor can place all the generated Java classes associated with the created ETD. (For this example, use `GenericEventPackage` as the package name.) Click OK and Finish to accept the names and open the ETD Editor to the new ETD.

9 Select `GenericEvent` in the Event Type Definition pane. Right click, and select Add Field, and as Child Node.

10 Select `GenericIn` and change the structure value under the Properties pane from delim to fixed.

11 Select `Field1` under `GenericIn` in the Event Type Definition pane, and change the structure value under the Properties pane from delim to fixed. (See Figure 6)

![Figure 6 Event Type Definition Editor](image)

12 Click File, Compile And Save. After the file has compiled the methods are displayed in the Event Type Definition pane. Promote to Run Time and close the ETD Editor.
4.5.2. IMSClientETD.xsc

The **IMSClientETD** Event Type Definition (ETD) can be used as a Source and/or Destination Event Type in the Collaboration Rules. The **IMSClientETD.xsc** is “read-only” by design. The structure and properties can be displayed, but it cannot be modified in any way. To open the IMSClientETD in the ETD Editor do the following:

1. From the e*Gate toolbar, click the ETD Editor button. The ETD Editor opens.

2. Click the Open button on the ETD Editor toolbar, find, and select **IMSClientETD.xsc** (etd\IMSClientETD\IMSClientETD.xsc). The ETD opens.

![Figure 7 IMSClientETD Event Type Definition](image)

3. **Save** the file and **Promote to Run Time** to move the file to the run time environment.
4.5.3. The IMS MFS Wizard and the MFS Converter

The IMS MFS file converter parses Message Formatting Services files and creates ETDs from the MID/MOD MSG definitions. The converter examines MSG, SEG, and MFLD statements and generates ETD node names and lengths.

- **MSG statements**: node generated, using Label and Type (Input or Output) to create the node name.
- **SEG statements**: a SEG subnode is created that contains all the MFLD statements before the next SEG or MSGEND statement.
- **MFLD statements**: the converter uses field names and the LTH= parameter to generate fixed length nodes.
- **FMT**, **PDB**, and **TABLE** sections of the file are not pertinent and are ignored.
- The following parameters are ignored: EJECT, PRINT, SPACE, STACK, UNSTAC, TITLE, ALPHA, EQU, COPY, RESCAN, ATTR, (SCA) and FILL.

**Considerations**

- Be sure the MFS definitions are syntactically valid, and can be assembled by the IBM MFS language utility without errors. Invalid syntax may cause the converter to yield unpredictable results.
- Column positioning is critical: the converter assumes that Labels begin in column 0, Operations in column 10, and Operands in column 16.
- The converter does not currently support the parameters: DO, and MFLD continuation lines.
- The IBM MFS Wizard is not supported for the IMS e*Way running on the OS/390 platform.

**Creating an Event Type Using the IMS MFS Wizard**

The MFS Wizard is used to generate ETDs from an IMS MFS (Message Format Service) file, that map input and output message segments at the field level. A sample MFS data file, MFSSAMP.mfs, is included on the Installation CD-ROM in the ..\samples\ewims directory. To create an ETD using the MFS Wizard do the following:

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. Enter a name for the Event Type, then click **OK**.
3. Double-click the new Event Type to edit its properties. When the **Properties** window opens, click the **New** button. The ETD Editor opens.
4. Select **New** from the ETD Editor’s **File** menu. The **New Event Type Definition** window opens.
Figure 8  New Event Type Definition - IMS MFS Wizard

5 Select the MFSWizard and click OK. The MFS Wizard opens.

6 On page two of the MFS Wizard, Enter the **Package Name** where the MFS Builder will place all generated Java classes associated with the created ETD.

7 Enter the **MFS File Name** by using the **Browse** button to navigate to the IMS MFS file that will be used to create the ETD. For the sample, navigate to and select the MFSSAMP.mfs file provided on the Installation CD-ROM. (See Figure 9.)

Figure 9  IMS MFS Wizard - Page 2

8 Click Next. Confirm that the information entered is correct and click **OK**. The Generated ETD opens in the ETD Editor.
The ETD is now available to configure as needed. When finished, click File, Compile and Save, and Promote to Run Time.

**Note:** For more information on the creating and modifying Java-enabled ETDs, see the “e*Gate Integrator User’s Guide” or consult e*Gate’s online help.

**Sample Input File — MFSSAMP.mfs**

The sample MFS data file, **MFSSAMP.mfs**, is included on the Installation CD-ROM in the ..\samples\ewims directory. The text of the sample appears as follows:

```
PRINT ON,NOGEN 00020000
IVTCV MSG TYPE=OUTPUT,SOR=(IVTCVF,IGNORE),NXT=IVTCVM11 00440000
SEG MSG,LTH=40 00460000
MFLD MSG,LTH=40 00460000
MFLD CMD,LTH=8 00470000
MFLD NAME1,LTH=10 00480000
MFLD NAME2,LTH=10 00480000
MFLD EXTN,LTH=10 00500000
```
The next step is to create and associate Intelligent Queues (IQs). IQs manage the exchange of information between components within the e*Gate system and provide non-volatile storage for data as it passes from one component to another. IQs use IQ Services to transport data. IQ Services provide the mechanism for moving Events between IQs, and for handling the low-level implementation of data exchange (such as system calls to initialize or reorganize a database).
The Java implementation of the IMS e*Way uses the SeeBeyond JMS (Java Message Service) IQ Service. For more information on the SeeBeyond JMS IQ see the SeeBeyond JMS Intelligent Queue User’s Guide.

4.6.1. Creating the Intelligent Queue

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. The Service defaults to STC_JMS_IQ and cannot be changed. This is true for any IQ created for an IQ Manager with the type SeeBeyond_JMS.
9. Set the initialization string and the Event Type Get Interval if necessary. The default Event Type Get Interval of 100 Milliseconds is satisfactory for the purposes of this implementation.

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

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

4.7 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.

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.

- cr_Request_in (Pass Through)
- cl_Response_out (Pass Through)
- cr_ims_interchange (Java)
4.7.1. Creating the Collaboration Rules

**cr_Request_in Collaboration (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 “cr_Request_in”), then click **OK**.
5. Double-click the new Collaboration Rules Component. The **Collaboration Rules Properties** window opens.
6. The **Service** field defaults to **Pass Through**.
7. Go to the **Subscriptions** tab. Select **GenericEvent** 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_Request_to_ims** under **Available Output Event Types**, and click the right arrow to move it to **Selected Output Event Types**. Make sure that **etd_Request_to_ims** is selected as default.

**Figure 11**  Collaboration Properties

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

**cr_Response_out Collaboration (Pass Through)**

1. To create the **cr_Response_out Collaboration Rules** repeat steps 1–9, substituting:
**cr_ims_interchange Collaboration (Java)**

1. To create the Java Collaboration Rules select the **Collaboration Rules** folder.
2. On the palette, click the **Create New Collaboration Rules** button.
3. Enter the name of the new Collaboration Rule, then click **OK** (for this sample, use **cr_ims_interchange**).
5. From the **Service** field drop-down box, select **Java**. The **Collaboration Mapping** tab is now enabled, and the **Subscriptions** and **Publications** tabs are disabled.
6. In the **Initialization string** field, enter any required initialization string that the Collaboration Service may require. This field can be left blank.
7. Select the **Collaboration Mapping** tab.

![Collaboration Rules - Collaboration Mapping](image)

8. Using the **Add Instance** button, create instances to coincide with the Event Types. For this sample, do the following:
   
   A. In the **Instance Name** column, enter **req_in** for the instance name.
   B. Click **Find**, navigate to and double-click **GenericEvent.xsc**. This adds **GenericEvent.xsc** to the **ETD** column for this instance.

- **cr_Response_out** for the name in step 4.
- **etd_Response_from_ims** for the Selected Input Event in step 7.
- **GenericEvent** for the Selected Output Event in step 8.
C  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.

D  In the **Trigger** column, make sure that the checkbox is selected (triggered).

E  In the **Manual Publish** column, make sure the value in N/A.

9  Repeat step 8 again using the following values:

   - Instance Name — **resp_out**
   - ETD — **GenericEvent.xsc**
   - Mode — **Out**
   - Trigger — N/A
   - Manual Publish - clear

10  Repeat step 8 using the following values:

   - Instance Name — **ims_work**
   - ETD — **IMSClientETD.xsc**
   - Mode — **In/Out**
   - Trigger — clear
   - Manual Publish - clear

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

12  For optimum viewing expand the window to full size, and expand the Source and Destination Events as well. The following section describes how the business logic for the schema is set up using the Java Collaboration Rules Editor.

---

### 4.8 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 installation CD-ROM. Please refer to the completed class, **cr_ims_interchange.class** when completing the Collaboration Rules Properties.

1  As stated in the above section, the Java Collaboration Rules Editor opens to the **cr_ims_interchange.xpr** from the Collaboration Rules Properties dialog box by clicking the **New** or **Edit** button under the Collaboration Rules field. Expand to full size for optimum viewing, expanding the Source and Destination Events as well.
Select `retBoolean` in the Business Rules pane. All of the user-defined business rules are added as part of this method. To view the created code in the Business Rules window, click View on the Menu bar and select Display Code.

Each new rule is created by clicking the rule button on the Business Rules toolbar, or by dragging and dropping one object to another from the Source and Destination Events panes. For additional information on using the Java Collaboration Rules Editor, see the e*Gate Integrator User’s Guide. The `cr_ims_interchange.xpr` business rules are created as follows.
### Figure 14 Collaboration Rules Editor – Business Rules

<table>
<thead>
<tr>
<th>Business Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGate.collabTrace()</td>
</tr>
<tr>
<td>Drag <strong>TranCode</strong> located under <strong>ims_work</strong> (Source Events) into the set of parentheses to create the following code: EGate.collabTrace(getims_work().getTranCode())</td>
</tr>
</tbody>
</table>

3. The **Display tran code to log** rule is a trace statement created by typing the following in the Rule Properties, Rule window:

   EGate.collabTrace()

   Drag **TranCode** located under **ims_work** (Source Events) into the set of parentheses to create the following code:

   EGate.collabTrace(getims_work().getTranCode())

4. The **Connect to IMS** rule is created by dragging the **connect** method located under **ims_work** (Source Events) into the Rule Properties, Rule window.

5. The **Move request message** rule is created by dragging and dropping **Field1** under **req_in** in the Source Events pane, to **Message** under **ims_work** in the Destination Events pane. The following code is created:

   getims_work().setMessage(getreq_in().getField1())

6. The **Display request in log** rule is another trace statement created by typing the following in the Rule Properties, Rule window:

   EGate.collabTrace()

   Drag **Message** located under **ims_work** (Source Events) into the set of parentheses to create the following code:

   EGate.collabTrace(getims_work().getMessage())

7. The **Execute IMS call** rule is created by dragging the IMSRequest under **ims_work** (Source Events) into the Rule Properties, Rule window to create the following code:

   getims_work().IMSRequest()

8. The **Display response in log** rule is another trace statement created by typing the following in the Rule Properties, Rule window:

   EGate.collabTrace()

   Drag the **getReply** method located under **ims_work** (Source Events) into the set of parentheses to create the following code:

   EGate.collabTrace(getims_work().getReply())
The **Move response to outbound etd** rule is created by dragging **Field1** under **req_in** in the Destination Events pane to the Rule Properties, Rule window to create the following code:

```java
getresp_out().setField1()
```

Drag the **getReply** method located under **ims_work** (Source Events) into the set of parentheses to create the following code:

```java
getresp_out().setField1(getims_work().getReply())
```

The **Disconnect from IMS** rule is created by dragging the **disconnect** method located under **ims_work** (Source Events) into the Rule Properties, Rule window to create the following code:

```java
getims_work().disconnect()
```

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. For further information on using the Collaboration Rules Editor see the *e*Gate Integrator User’s Guide.

### Creating the 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.

#### 4.9.1. Creating the Inbound col_Request_in 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, and select the **Control Broker**.
3. Select the **ew_Request_in** e*Way to assign its Collaboration.
4. On the palette, click the **Create a New Collaboration** button.
5. Enter the name of the new Collaboration (for the sample, “**col_Request_in**”) then click **OK**.
6. Double-click the new Collaboration to edit its properties. The **Collaboration Properties** dialog box opens.
7. From the **Collaboration Rules** drop-down list box, select the Collaboration Rules file that you created previously (for the sample, “**cr_Request_in**”).
8. 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 Generic **Event Type** “**GenericEvent**.”
B Select the **Source** from the **Source** drop-down list box. In this case, it should be `<EXTERNAL>`.

9 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** provided with the sample “etd_Request_to_ims.”

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_Request_in

4.9.2. Creating the col_IMS_interchange Multi-Mode Collaboration

1 Select the Multi-Mode e*Way to assign the Collaboration (for this sample, “ew_IMS_interface”).

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

3 Enter the name of the new Collaboration, then click **OK**. (For the sample, “col_IMS_interchange”)

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

5 From the **Collaboration Rules** drop-down list box select the Collaboration Rules file that you created previously (for the sample, “cr_ims_interchange”).

6 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 “req_in.”

B From the Event Type drop-down list box, select the Event Type “etd_Request_to_ims” provided with the sample schema.

C From the Source drop-down list box, select the source (for this sample “col_Request_in”).

7 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 “resp_out.”

B From the Event Types drop-down list box, select the Event Type “etd_Response_from_ims” provided with the sample schema.

C Select the publication destination from the Destination drop-down list box. In this case, it should be “IQ1.”

D The value in the Priority column defaults to 5.

Figure 16  Collaboration Properties - col_IMS_interchange

8 Again, 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 “ims_work.”

B From the Event Types drop-down list box, select the generic Event Type “GenericEvent.”

C Select the publication destination from the Destination drop-down list box. In this case, it should be “ecp_IMS.”
The value in the **Priority** column defaults to 5.

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

### 4.9.3. Creating the col_Response_out e*Way Collaboration

To create the `col_Response_out` e*Way Collaboration follow the steps under **Creating the Inbound col_Request_in e*Way Collaboration** on page 51, substituting the following properties as displayed in Figure 17:

- Subscriptions Event Type — `etd_Response_from_ims`
- Subscriptions Source — `col_IMS_interchange`
- Publications Event Type — `GenericEvent`
- Publications Destination — `<EXTERNAL>`
- Publications Priority — 5

**Figure 17** Collaboration Properties col_Response_out

---

### 4.10 Executing the Schema

To execute the IMS_Sample schema, do the following:

1. Go to the command line prompt, and enter the following:
   ```
   stccb -rh hostname -rs IMS_Sample -un username -up user password -ln hostname_cb
   ```
   Substitute `hostname`, `username` and `user password` as appropriate.
2 Start the e*Gate Monitor. Specify the server that contains the Control Broker you started in Step 1 above.

3 Select the IMS_Sample schema.

4 Verify that the Control Broker is connected. To do this, select and right-click the Control Broker in the e*Gate Monitor, and select Status. (The message in the Control tab of the console will indicate command succeeded and status as up.)

5 Select the IQ Manager, hostname_igmgr, then right-click and select Start. (This will already be started if Start automatically is selected in the IQ Manager properties.)

6 Select each of the e*Ways, right-click select Start. (These will already be started if Start automatically is selected in the e*Way’s properties.)

7 To view the output, copy the output file (specified in the Outbound e*Way configuration file). Save to a convenient location, open.

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

4.11 Error Messages

Errors will return an abbreviated error message to the e*Way log and output data file. A more detailed message is sent to the MVS System. For example, Running the IMS e*Way with (a) a valid user ID and non-valid password, and (b) a non-valid user ID and a valid password returns the following message to the e*Way:

*REQSTS*4, *REQSTS*40

The following message appears in the MVS Systems log:

ICH408I USER(P390 ) GROUP(SYS1 ) NAME(P390 ) 017
LOGON/JOB INITIATION - INVALID PASSWORD
IRRO13I VERIFICATION FAILED. INVALID PASSWORD GIVEN.
HWSP1500E security violation; R=8, C=STCEWAY , RACFRC=8, RACFRS=0, M=SDRC
ICH408I USER(BART ) GROUP(SYS1 ) NAME(???) 020
LOGON/JOB INITIATION - USER AT TERMINAL NOT RACF-DEFINED
IRRO12I VERIFICATION FAILED. USER PROFILE NOT FOUND.
HWSP1500E security violation; R=8, C=STCEWAY , RACFRC=4, RACFRS=0, M=SDRC

4.12 Sample Schema Implementation for OS/390

The OS/390 implementation of the sample schema Java_IMS_os390 is similar to the sample implementation provided in the preceding sections with the following exceptions:

1 The IQ Manager Type is set to SeeBeyond Standard, and the IQ Service is STC Standard. (See Intelligent Queues on page 44.)
In the Collaboration Rules, the `cr_ims_interchange` initialization string value has been changed from (null) to `jnidl /usr/lpp/java/J1.3/bin/classic/libjvm.so`. (See Creating the Collaboration Rules Class on page 48.)

For the Event Type Definition `GenericEvent.xsc`, the encoding is set to iso-8859-1. (See Event Types on page 38.)

In the `cr_ims_interchange` Collaboration a "Convert EBCDIC data" has been added to the userInitialize() method, as seen in Figure 18.

**Figure 18** Collaboration Rules - `cr_ims_interchange` for OS/390 Sample
Java Methods

The IMS e*Way contains Java methods that are used to extend the functionality of the e*Way. These methods are contained in the following class

5.1 The IMSClientETD Class

java.lang.Object
  | + -- com.stc.eways.ims.EwayConnectionETDImpl
  |     + -- com.stc.eways.ims.IMSClientETD.

public class IMSClientETD

Extends com.stc.eway.ims.EwayConnectionETDImpl.
Implements com.stc.jcsre.ETDExt.

Methods of the IMSClientETD Class

These methods are described in detail on the following pages:

  connect on page 58  IMSRequestWithMultiSegs on page 64
  disconnect on page 58  isConnected on page 64
  getBandrs on page 59  setBandrs on page 65
  getDatastoreID on page 59  setDatstoreID on page 65
  getLtermName on page 59  setLtermName on page 66
  getMessage on page 60  setMessage on page 66
  getPassword on page 60  setPassword on page 66
  getPort on page 61  setPort on page 67
  getRacfGroupName on page 61  setRacfGroupName on page 67
  getRacfUserID on page 61  setRacfUserID on page 68
**connect**

**Description**
Connect to server. Connection management is only supported in e*Gate 4.5.2 and above.

**Syntax**
```java
public void connect()
```

**Parameters**
None

**Return Values**
None

**Throws**
```java
com.stc.common.collabService.CollabConnException
```

---

**disconnect**

**Description**
Disconnect from server. Connection management is only supported in e*Gate 4.5.2 and above.

**Syntax**
```java
public void disconnect()
```

**Parameters**
None

**Return Values**
None

**Throws**
```java
com.stc.common.collabService.CollabConnException
```
getBandrs

Description
Called in the Collaboration to get the Bandrs (IRM_F4) attribute.

Syntax
public char getBandrs()

Parameters
None

Return Values
char
The BANDRS (NoAck, ACK, NACK, Deallocate, Resume, SendOnly).

Throws
None

getDatastoreID

Description
Called in the Collaboration to get the DatastoreID attribute.

Syntax
public java.lang.String getDatastoreID()

Parameters
None

Return Values
java.lang.String
The DatastoreID.

Throws
None

getLtermName

Description
Called in the Collaboration to get the LtermName attribute.

Syntax
public java.lang.String getLtermName()

Parameters
None
Return Values

java.lang.String
The LtermName.

Throws
None

---

getMessage

Description
Called in the Collaboration to get the Message attribute.

Syntax

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

Parameters

None

Return Values

java.lang.String
The Message.

Throws
None

---

getPassword

Description
Called in the Collaboration to get the Password attribute.

Syntax

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

Parameters

None

Return Values

java.lang.String
The Password.

Throws
None
getPort

Description
Called in the Collaboration to get the Port attribute.

Syntax
public java.lang.String getPort()

Parameters
None

Return Values
java.lang.String
The server Port.

Throws
None

getRacfGroupName

Description
Called in the Collaboration to set the RacfGroupName attribute.

Syntax
public java.lang.String getRacfGroupName()

Parameters
None

Return Values
java.lang.String
The RacfGroupName.

Throws
None

getRacfUserID

Description
Called in the Collaboration to get the RacfUserID attribute.

Syntax
public java.lang.String getRacfUserID()

Parameters
None
Return Values

java.lang.String  
   The RacfUserID.

Throws

None

---

### getReply

**Description**

Called in the Collaboration to get the reply from the server.

**Syntax**

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

**Parameters**

None

**Return Values**

java.lang.String  
   The reply from the server.

**Throws**

com.stc.common.collabService.CollabConnException
com.stc.common.collabService.CollabDataException
com.stc.common.collabService.CollabResendException

---

### getServer

**Description**

Called in the Collaboration to get the Server attribute.

**Syntax**

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

**Parameters**

None

**Return Values**

java.lang.String  
   The Server host.

**Throws**

None
getTranCode

Description

Called in the Collaboration to get the TransCode attribute.

Syntax

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

Parameters

None

Return Values

```
java.lang.String
```

The transaction code (TransCode).

Throws

None

getTranCodeSrc

Description

Called in the Collaboration to get the trans code source attribute.

Syntax

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

Parameters

None

Return Values

```
java.lang.String
```

The transaction code source (TransCodeSource).

Throws

None

IMSRequest

Description

Called in the Collaboration to send multi-segmented messages to the server.

Syntax

```
public void IMSRequest(java.lang.String inputMessage)
```
Chapter 5
Java Methods
Section 5.1
The IMSClientETD Class

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputMessage</td>
<td>java.lang.String</td>
<td>Message segment to be sent to IMS.</td>
</tr>
</tbody>
</table>

Return Values
None

Throws
com.stc.common.collabService.CollabConnException
com.stc.common.collabService.CollabDataException
com.stc.common.collabService.CollabResendException

IMSRequestWithMultiSegs

Description
Called in the Collaboration to send multi-segmented messages to the server.

Syntax
public void IMSRequestWithMultiSegs(java.lang.String[] inputMessages)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputMessages</td>
<td>java.lang.String</td>
<td>Message segments to be sent to IMS.</td>
</tr>
</tbody>
</table>

Return Values
None

Throws
com.stc.common.collabService.CollabConnException
com.stc.common.collabService.CollabDataException
com.stc.common.collabService.CollabResendException

isConnected

Description
Checks for a connection to the server. Connection management is only supported in e*Gate 4.5.2 and above.

Syntax
public boolean isConnected()
Return Values
   Boolean
Throws
   com.stc.common.collabService.CollabConnException

**setBandrs**

Description
   Called in the Collaboration to set the Bandrs (IRM_F4) attribute.

Syntax
   public void setBandrs(char msg)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>char</td>
<td>blank = NoAck, A = ACK, N = NACK, D = Deallocate, R = Resume, S = SendOnly</td>
</tr>
</tbody>
</table>

Return Values
   None
Throws
   None

**setDatastoreID**

Description
   Called in the Collaboration to set the datastoreID attribute.

Syntax
   public void setDatastoreID(java.lang.String msg)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>DatastoreID</td>
</tr>
</tbody>
</table>

Return Values
   None
Throws
   None
**setLtermName**

Description
---
Called in the Collaboration to set the ltermName attribute.

Syntax
---
public void setLtermName(java.lang.String msg)

Parameters
---
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>ltermName</td>
</tr>
</tbody>
</table>

Return Values
---
None

Throws
---
None

**setMessage**

Description
---
Called in the Collaboration to set the message attribute.

Syntax
---
public void setMessage(java.lang.String msg)

Parameters
---
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>Message</td>
</tr>
</tbody>
</table>

Return Values
---
None

Throws
---
None

**setPassword**

Description
---
Called in the Collaboration to set the password attribute.

Syntax
---
public void setPassword(java.lang.String msg)
### Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>Password</td>
</tr>
</tbody>
</table>

### Return Values

None

### Throws

None

### setPort

**Description**

Called in the Collaboration to set the port attribute.

**Syntax**

```java
public void setPort(java.lang.String port)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>java.lang.String</td>
<td>Server Port</td>
</tr>
</tbody>
</table>

### Return Values

None

### Throws

None

### setRacfGroupName

**Description**

Called in the Collaboration to get the racfGroupName attribute.

**Syntax**

```java
public void setRacfGroupName(java.lang.String msg)
```

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>RacfGroupName</td>
</tr>
</tbody>
</table>

### Return Values

None
setRacfUserID

Description
Called in the Collaboration to set the racfUserID attribute.

Syntax

    public void setRacfUserID(java.lang.String msg)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>RacfGroupName</td>
</tr>
</tbody>
</table>

Return Values
None

Throws
None

setServer

Description
Called in the Collaboration to set the server attribute.

Syntax

    public void setServer(java.lang.String server)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>java.lang.String</td>
<td>Server host</td>
</tr>
</tbody>
</table>

Return Values
None

Throws
None
setTranCode

Description
Called in the Collaboration to set the trans code attribute.

Syntax
public void setTranCode(java.lang.String msg)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>TranCode</td>
</tr>
</tbody>
</table>

Return Values
None

Throws
None

setTranCodeSrc

Description
Called in the Collaboration to set the trans code source attribute. If tranCodeSrc is set to "MESSAGE", the transaction ID will be picked up from the first eight bytes of the Message. Otherwise, it will be picked up from the config file.

Syntax
public void setTranCodeSrc(java.lang.String msg)

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>java.lang.String</td>
<td>TranCodeSrc</td>
</tr>
</tbody>
</table>

Return Values
None

Throws
None
**terminate**

**Description**
Closes the external connection if NOT in a sub-collaboration. If in sub-collaboration, release resources used in the sub-collaboration, but do not close the external connection.

**Syntax**

```java
public void terminate()
```

**Parameters**
None

**Return Values**
None

**Throws**

```java
com.stc.common.collabService.CollabConnException
```

**Specified by**

```java
terminate in interface com.stc.jcsre.ETD
```

**Overrides**

```java
terminate in class com.stc.eways.ims.EwayConnectionETDImpl
```
Index

B
business rules
 creating 48
 sample 48

C
Collaboration Rules 45
 .xpr files 48
 creating 45, 46
 editor 48
 Java 48
 properties 45
 configuration 16
 Multi-Mode e*Way 16
 connection management 33
 Custom ETD Wizard 38

E
e*Way Connection
 creating 21, 37
 get interval 22
 parameters 21
 Class 23
 Connection Establishment Mode 23
 Connection Inactivity Timeout 24
 Connection Verification Interval 24
 Connector 22
 IRM Header 26
 IRM_CLIENTID 28
 IRM_DESTID 30
 IRM_F1 28
 IRM_F2 28
 IRM_F3 29
 IRM_F4 29
 IRM_F5 27
 IRM_ID 26
 IRM_LEN 26
 IRM_LTERM 30
 IRM_RACF_GRPNAME 31
 IRM_RACF_PW 31
 IRM_RACF_USERID 30
 IRM_SOCT 27

IRM_TIMER 27
IRM_TRNCOD 30
IRM_TRNCOD_SRC 30
Port 25
Property Tag 23
Server 25
TCPIIP Configuration 24
Type 23
e*Ways
 creating and configuring 33
 Inbound e*Way 34
 Multi-Mode e*Way 36
 Outbound e*Way 35
error messages
 logs 55
ETDs 38
 Event Type Definitions 38
 Event Types 38

I
implementation 32
 overview 32
 importing the sample schema 33
 IMSClientETD.xsc
 opening 40
 installation
 directories and files created 15
 UNIX 14
 Windows 2000 and NT 12
 Intelligent Queues 44
 creating 44
 IQs 44
 creating 44

J
JVM settings 17

M
methods 57
 IMSClientETD Class 57
 disconnect 58
 getBandrs 59
 getDatastoreID 59
 getLtermName 59
 getMessage 60
 getPassword 60
 getPort 61
 getRacfGroupName 61
 getRacfUserID 61
 getReply 62
Index

getServer 62
getTranCode 63
getTranCodeSrc 63
IMSRequest 63
IMSRequestWithMultiSegs 64
isConnected 64
setBandrs 65
setDatastoreID 65
setLtermName 66
setMessage 66
setPassword 66
setPort 67
setRacfGroupName 67
setRacfUserID 68
setServer 68
setTranCode 69
setTranCodeSrc 69
terminate 70

Multi-Mode e*Way
creating 16
parameters 17
CLASSPATH Append From Environment Variable 19
CLASSPATH Override 18
CLASSPATH Prepend 18
Disable JIT 20
initial Heap Size 19
JNI DLL Absolute Pathname 17
JVM Settings 17
Maximum Heap Size 19
Maximum Stack Size for JVM Threads 20
Maximum Stack Size for Native Threads 19
Remote debugging port number 20
Rollback Wait Interval 21
Suspend option for debugging 20

S
sample schema
executing the schema 54
importing 33
schema
importing 33
Subcollaborations 33
Supporting Documents 11

P
parameters
Class 23
CLASSPATH Append From Environment Variable 19
CLASSPATH Override 18
CLASSPATH Prepend 18
Connection 23
Connection Inactivity Timeout 24
Connection Verification Interval 24
Connector 22
Disable JIT 20
initial Heap Size 19
IRM_CLIENTID 28
IRM_DESTID 30
IRM_F1 28
IRM_F2 28
IRM_F3 29
IRM_F4 29
IRM_F5 27
IRM_ID 26
IRM_LEN 26
IRM_LTERM 30
IRM_RACF_GRNAME 31
IRM_RACF_PW 31
IRM_RACF_USERID 30
IRM_SOCT 27
IRM_TIMER 27
IRM_TRNCOD 30
IRM_TRNCOD_SRC 30
JNI DLL Absolute Pathname 17
Maximum Heap Size 19
Maximum Stack Size for JVM Threads 20
Maximum Stack Size for Native Threads 19
Multi-Mode e*Way 18
JNI DLL absolute pathname 17
JVM settings 17
Port 25
Property Tag 23
Remote debugging port number 20
Rollback Wait Interval 21
Server 25
Suspend option for debugging 20
Type 23