SeeBeyond ICAN Suite

TCP/IP eWay Intelligent Adapter User's Guide

Release 5.0.2

SEEBEYOND®

Version 20040604102534.

Contents

Chapter 1

Introducing the TCP/IP eWay 6

Overview 6
TCP/IP eWay 6
Setting Properties 7

Supported Operating Systems 7

System Requirements 7

External System Requirements 7

HP NonStop Server Requirements 8

Chapter 2

Installing the TCP/IP eWay 9

Installation Procedures 9
Before installing the eWay 9
Installing the TCP/IP eWay on an eGate-supported system 9

After Installation 10

Chapter 3

Setting TCP/IP eWay Properties 11

TCP/IP eWay Properties Dialog Box 11

Settings Properties on the Connectivity Map 13
Setting eWay Properties in Server Mode 14
Envelope Message (Server) 15
Bytes to Read 15
Custom Enveloped Class Name 15
Customer Defined Property 15
Envelope Type 16
Ignore Until Char Value 19
Numeric Representation 19
Store Until Char Value 19
Width of Length 19
General Server Settings 19
Dedicated Session Mode 20

Max Data Size 20
TCP/IP Server Base Settings 20
Keep Alive 21
Receive Buffer Size 21
Send Buffer Size 21
Server Socket Factory Implementation Class 21
ServerSoTimeout 22
SoLinger 22
SoLinger Timeout 22
SoTimeout 22
TcpNoDelay 22
Setting eWay Properties in Client Mode 23
General Client Settings 24
Max Data Size 24
TCP/IP Base Settings 24
Keep Alive 24
Receive Buffer Size 24
Send Buffer Size 25
Socket Factory Implementation Class 25
SoLinger 25
SoLinger Timeout 25
SoTimeout 26
TcpNoDelay 26
Envelope Message (Client) 26

Setting Properties in the Project Environment 26
Setting eWay Properties in Server Mode 26
ServerPort 27
Setting eWay Properties in Client Mode 28
Host 28
Port 29

Chapter 4

Reviewing the Sample Project 30

eGate Project Description 30
Projects and the Enterprise Designer 30
Importing Sample Projects 31
Basic eWay Components 32
TCP/IP eWay Properties 32
TCP/IP OTDs 32
Customized Enveloping 32

Overview: Sample Project 36
Project: General Scenarios 36
Client Mode 36
Server Mode 37
Sample Project General Operation 37
Project Components 37
Project Operation 38
Input and Output Data 38
Handshake Protocol 38
Custom Envelope 39

Summary: Sample Java Collaboration Project 39
Using the eWay With Java-based Collaborations 39
Creating Business Rules Within Collaboration Definitions 39
Collaboration Editor (Java) Window 40
Creating the Project's Environment 40
Setting eWay Properties 40

Deploying a Project 43
Basic Steps 43
For and While Rules for Monitoring 44
Alerting and Logging 44
Deploying on HP NonStop Server Systems 44
Parallel Library TCP/IP 44
Using the File 45

Using the TCP/IP OTDs 46
OTD Overview 46
OTD Components 47
Basic OTD Operation 47
TCP/IP OTD Features 48

Chapter 5

Using eWay Java Methods 50

TCP/IP eWay Methods and Classes: Overview 50
Relation to eWay Configuration 50
TCP/IP eWay Javadoc 50

eWay Java Classes/Interfaces 51

Appendix A

Methods Not Used in the eWay 52

Index 55

Chapter 1

Introducing the TCP/IP eWay

This guide explains how to install, set properties for, and operate the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) TCP/IP eWay Intelligent Adapter, referred to as the TCP/IP eWay throughout this guide.

This chapter provides a brief overview of operations, components, general features, and system requirements of the TCP/IP eWay.

Chapter Topics

- **"Overview" on page 6**
- **"Supported Operating Systems" on page 7**
- **"System Requirements" on page 7**
- **"External System Requirements" on page 7**
- **"HP NonStop Server Requirements" on page 8**

1.1 Overview

The TCP/IP eWay enables the eGate Integrator to communicate with client applications using TCP/IP, and provides real-time, reliable data transfer for systems that support TCP/IP.

For details on operating and using eGate Integrator and its user interface, see the *eGate Integrator User's Guide.*

1.1.1 TCP/IP eWay

The TCP/IP eWay allows you to create a client interface to the server or implement a server in eGate, using an eGate Collaboration framework created using the eGate Enterprise Designer. This interface allows your system to communicate via TCP/IP.

The TCP/IP Object Type Definition (OTD) enables the creation of any messaging protocol capable of running over TCP/IP, and also utilizes the common eWay services available in eGate.

The eWay also allows you to select your desired message enveloping, using predefined envelope types. If these types do not meet your needs, you can customize your own enveloping, using specialized eWay interfaces designed for this purpose.

1.1.2 Setting Properties

The TCP/IP eWay properties provide the necessary parameters for the operation of the TCP/IP eWay. These properties are adopted into the OTD's functions.

See **Chapter 3** for details on how to set the TCP/IP eWay properties.

1.2 Supported Operating Systems

The TCP/IP eWay is available for the following operating systems:

- Windows XP, Windows 2000, and Windows Server 2003
- HP-UX 11.0, 11i (PA-RISC), and 11i V2 (11.23)
- IBM AIX 5.1L and 5.2
- Red Hat Linux 8 (Intel) and Advanced Server 2.1 (Intel)
- Sun Solaris 8 and 9
- HP NonStop Server, G06.22

1.3 System Requirements

To use the TCP/IP eWay, you need:

- eGate Logical Host, version 5.0 or later.
- TCP/IP network connection.

Logical Host Requirements

The eWay must be set up and administered using the Enterprise Designer. For complete information on the eGate Enterprise Designer system requirements, see the *SeeBeyond ICAN Suite Installation Guide*.

1.4 External System Requirements

To enable the eWay to communicate properly with the TCP/IP system, you need:

- Host on which the server is running (host name)
- Port on which the server is listening (port number)

1.5 HP NonStop Server Requirements

For a list of the HP NonStop server requirements, see the *SeeBeyond ICAN Suite Installation Guide*.

Chapter 2

Installing the TCP/IP eWay

This chapter explains how to install the TCP/IP eWay.

Chapter Topics

- **"Installation Procedures" on page 9**
- **"After Installation" on page 10**

2.1 Installation Procedures

During the ICAN Suite installation process, the Enterprise Manager, a Web-based application, is used to select and upload eWay and add-on files (.**sar** files) from the ICAN installation CD-ROM to the Repository.

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

Before installing the eWay

Open and review the **Readme.txt** file (located in the root directory of the ICAN installation's Repository CD-ROM) for the latest information, before installing the eWay.

Installing the TCP/IP eWay on an eGate-supported system

The TCP/IP eWay can be installed during the installation of eGate. The eGate installation process includes the following operations:

- Installing the eGate Repository
- Uploading products to the Repository
- Downloading the components (including the eGate Enterprise Designer and the Logical Host)
- Viewing the product information home pages

Follow the instructions for installing the ICAN Suite found in the *SeeBeyond ICAN Suite Installation Guide*, and include the following steps:

1. After the eGate or eInsight core products are uploaded to the Repository using the Enterprise Manager, select and upload the **FileeWay.sar**. The File eWay is used to by the eWay's Project sample. You must upload the File eWay (**FileeWay.sar**.) before uploading the TCP/IP eWay (**TCPIPeWay.sar**).
2. After the File eWay is uploaded, upload **TCPIPeWay.sar** to install the TCP/IP eWay.
3. Next, upload the **TCPIPeWayDocs.sar**. This file contains the eWay user's guide, **Javadoc**, and sample Project files.

   To obtain these files, follow the instructions provided by the user interface.
4. If needed, continue installing eGate as instructed in the *SeeBeyond ICAN Suite Installation Guide.*

2.2 After Installation

Once the eWay is installed and configured, it must then be incorporated into a Project before it can perform its intended functions. See the *eGate Integrator User's Guide* for more information on incorporating the eWay into an eGate Project.

Chapter 3

Setting TCP/IP eWay Properties

This chapter explains how to set the properties for the TCP/IP eWay.

Chapter Topics

- **"TCP/IP eWay Properties Dialog Box" on page 11**
- **"Settings Properties on the Connectivity Map" on page 13**
- **"Setting Properties in the Project Environment" on page 26**

3.1 TCP/IP eWay Properties Dialog Box

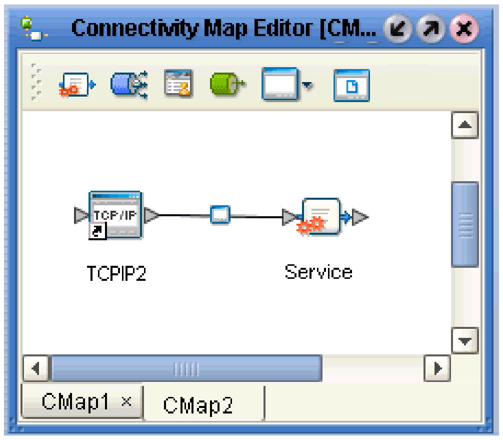
When you install the TCP/IP eWay, a default properties template for the eWay is also installed. The template's default properties are accessible via the eGate Enterprise Designer. These default settings apply to all TCP/IP eWays you use within your current Project or Business Process.

You can set properties for each individual eWay using the Enterprise Designer's eWay **Properties** dialog box. This section describes general procedures on how to change these default properties for the eWay. For details on these steps, see the *eGate Integrator User's Guide*.

To set properties for the TCP/IP eWay on the Connectivity Map

1. From the eGate Enterprise Designer's **Project Explorer** create at least one Connectivity Map.
2. Create the desired external systems for your one or more Connectivity Maps.
3. Select the external application whose default eWay properties you want to change by clicking the **eWay** icon. This icon is located on the link between an **External Application** icon and a **Service** icon on the Connectivity Map canvas. See **Figure 1 on page 12**.

Figure 1 eWay Icon



The eWay **Properties** dialog box appears. **Figure 2 on page 14**, and **Figure 3 on page 23** show the eWay's default properties available from the **Project Explorer** and Connectivity Map. You can use this window to modify the current eWay's properties settings.

To set properties for the TCP/IP eWay using on the Project Environment

1. From the Enterprise Designer, create your external systems.
2. Click the **Environment Explorer** tab (at the bottom of the left pane).
3. Create an environment for your project, then create external systems on the Environment canvas to correspond to the systems you created using the **Project Explorer**.
4. Select the external system whose default eWay properties you want to change by right-clicking the desired system's icon in the **Environment Explorer**.

   The eWay **Properties** dialog box appears. **Figure 4 on page 27** shows the eWay's default properties available from the **Environment Explorer**. You can use this dialog box to modify the eWay properties associated with the current external system.
5. Click **OK** then **Save All** to save your changes.

To use the eWay Properties dialog box

- The TCP/IP eWay's properties are set using the eGate Enterprise Designer's eWay **Properties** dialog box. The default properties are automatically provided.
- Clicking the **Configuration** (Connectivity Map) or **Environment Configuration** folder in the left pane displays properties group subfolders in the right pane. Click any subfolder to display the eWay's editable properties.
- Many of the entries allow you to enter text. Click the desired text box, then click the ellipsis (**...**) that appears, to open a dialog box for this purpose.

Note: *Even if you do not change the eWay's properties, you must open each **Properties** dialog box for every eWay and click **OK** to activate the eWay.*

The rest of this chapter explains all of the eWay's properties in detail, under the following sections:

- **"Settings Properties on the Connectivity Map" on page 13**
- **"Setting Properties in the Project Environment" on page 26**

3.2 Settings Properties on the Connectivity Map

This section explains in detail the eWay's editable properties accessible via the eGate Enterprise Designer's **Project Explorer** and Connectivity Map. You can set these properties using the eWay **Properties** dialog box.

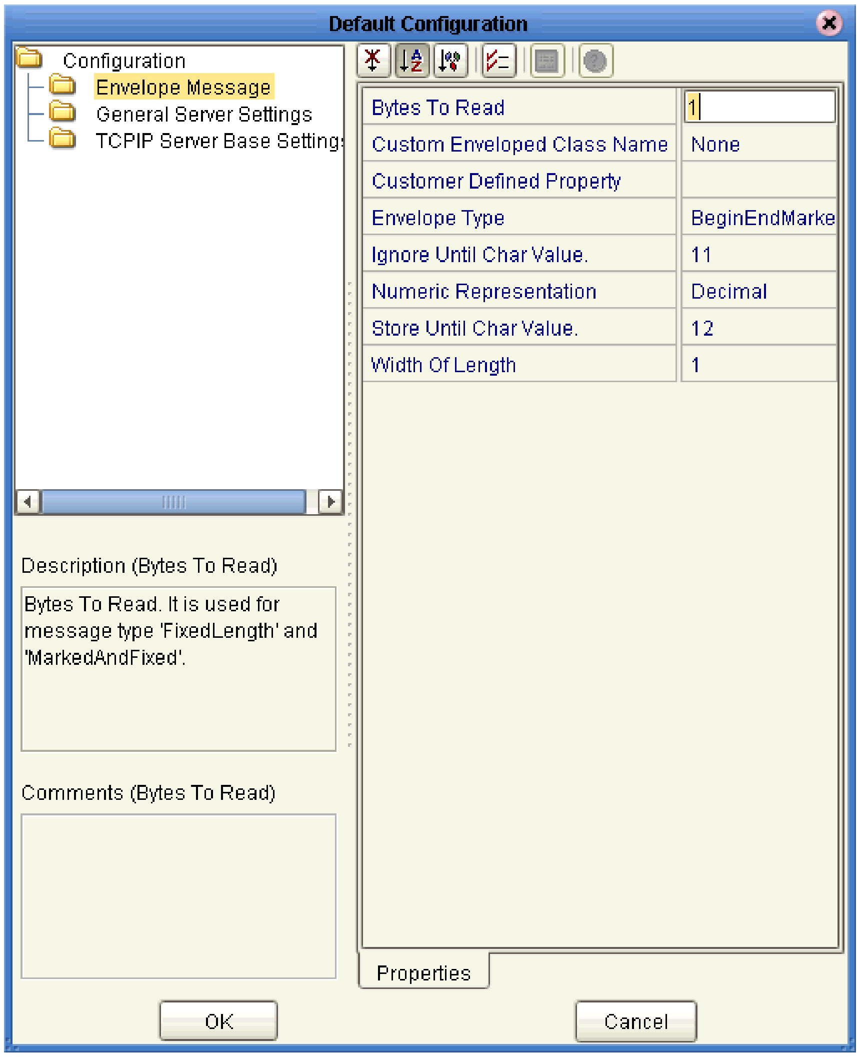
The TCP/IP eWay operates in two modes, client (outbound) and server (inbound). The properties relating to each of these modes are explained under the following sections:

- **"Setting eWay Properties in Server Mode" on page 14**
- **"Setting eWay Properties in Client Mode" on page 23**

3.2.1 Setting eWay Properties in Server Mode

The server (inbound) properties settings determine the eWay's behavior for input operations. See Figure 2.

Figure 2 eWay Properties Dialog Box: Server (Inbound) Connectivity Map



Bytes To Read	1
Custom Enveloped Class Name	None
Customer Defined Property	
Envelope Type	BeginEndMarke
Ignore Until Char Value.	11
Numeric Representation	Decimal
Store Until Char Value.	12
Width Of Length	1

Description (Bytes To Read)

Bytes To Read. It is used for message type 'FixedLength' and 'MarkedAndFixed'.

Comments (Bytes To Read)

These eWay server mode properties are organized under the following subfolders:

- **"Envelope Message (Server)" on page 15**
- **"General Server Settings" on page 19**
- **"TCP/IP Server Base Settings" on page 20**

3.2.2 Envelope Message (Server)

This section explains the envelope message format properties for the server. These properties are all associated with TCP/IP enveloping. The properties in this section are:

- **“Bytes to Read” on page 15**
- **“Custom Enveloped Class Name” on page 15**
- **“Customer Defined Property” on page 15**
- **“Envelope Type” on page 16**
- **“Ignore Until Char Value” on page 19**
- **“Numeric Representation” on page 19**
- **“Store Until Char Value” on page 19**
- **“Width of Length” on page 19**

Bytes to Read

Description

Specifies the number of bytes to read.

Required Values

An integer; the range is 1 to 2 billion, and the default is 1.

Custom Enveloped Class Name

Description

Specifies the Java class name to be used when the **Envelope Type** property is set to **Custom**.

If you are using a custom envelope you have created, using a Java class, you can import the Java **.jar** file containing the class into any desired Collaboration, using the Collaboration Editor (Java)’s file import feature. For complete information on how to use this feature, see the *eGate Integrator User’s Guide*.

For more details, see **“Customized Enveloping” on page 32**.

Required Values

A valid full Java class name.

Customer Defined Property

Description

Allows you to enter any set of user-defined properties for a custom envelope, to be used when the **Envelope Type** property is set to **Custom**. You can parse this information, such as delimiters, into your customized envelope message implementation.

Required Values

Any text string.

Envelope Type

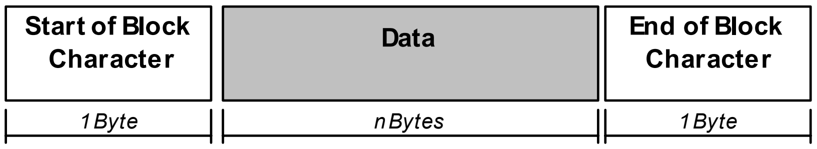
Description

Specifies the envelope type. The envelope type defines where a message starts and stops. The rest of this section explains the available envelope types and the structure of each.

Note: *For all envelope types, except **MarkedAndFixed**, the data is just the payload. See* **"MarkedAndFixed" on page 17** *for an explanation of how the data is handled by that envelope type.*

BeginEndMarked

The **BeginEndMarked** envelope has the following structure:

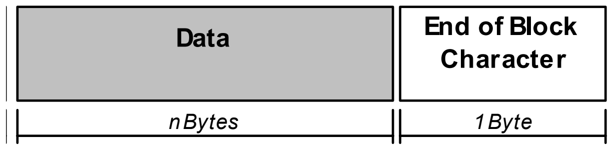


The **Start of Block Character** component of the Begin-end Marked envelope is the same as the editable **Ignore Until Character**. The **End of Block Character** component is the same as the editable **Store Until Character**.

If during the read process, the **Start of Block Character** is encountered, all read bytes are discarded and the read routine starts storing the incoming data from the last **Start of Block Character**.

EndMarked

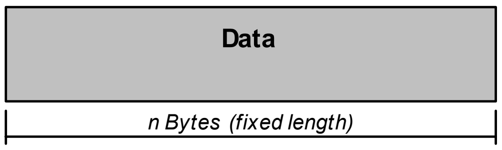
The **EndMarked** envelope has the following structure:



The **End of Block Character** component of the **EndMarked** envelope is the same as the editable **Store Until Character**.

FixedLength

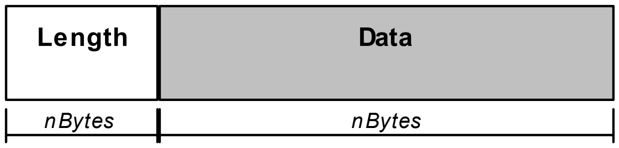
The **FixedLength** envelope has the following structure:



The **FixedLength** envelope type is set using the **Bytes to Read** editable property. It is assumed that all messages are the same length as specified by the **Bytes to Read** editable property.

LengthPrefixed

The **LengthPrefixed** envelope has the following structure:



The **Length** component of the **LengthPrefixed** envelope is an integer indicating the length of the **Data** component. The **Length** component has the following properties:

- **Numeric Representation** of the length
- **Allowed Width** of the length

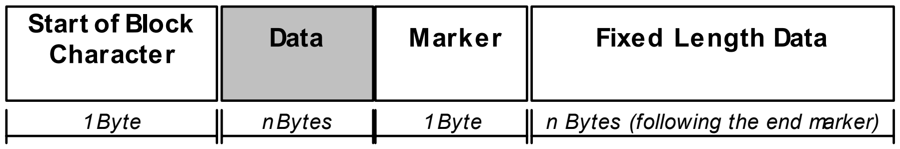
Table 1 displays the available values for each **Numeric Representation** and corresponding **Allowed Width**.

Table 1 Counter Component Values

Numeric Representation	Allowed Width
Decimal	1 to 10
Octal	1 to 8
Hexadecimal	1 to 16
Network short	2
Network long	4

MarkedAndFixed

The **MarkedAndFixed** envelope has the following structure:



The **MarkedAndFixed** envelope type is similar to the **BeginEndMarked** envelope, and is set using the properties **Ignore Until Character**, **Store Until Character**, and **Bytes to Read**.

The **Start of Block Character** in the diagram is the same as the property **Ignore Until Character**, and the **Marker** component is the same as the property **Store Until Character**. The communication client reads the marker before reading the remainder of the envelope as specified by the **Bytes to Read** property (**Fixed Length Data** in the diagram).

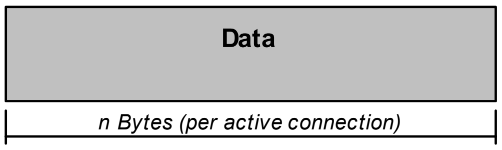
You can express this data structure in the following formula:

> ***Ignore Until Character*** *+ Marked Data +* ***Store Until Character*** *+ Fixed Data (****Bytes to Read****) = Data Structure*

The **Ignore Until Character**, **Store Until Character**, and **Bytes to Read** values are represented by ASCII number properties settings in the eWay's properties.

PerActiveConnection

The **PerActiveConnection** envelope has the following structure:



The message is not enclosed in an envelope, and the current connection is closed by the eWay when an entire message is sent. The receiver knows the entire message has been received when the sender closes the connection.

If you choose the **PerActiveConnection** envelope type, **sendEnvelopedMsg()** must be last method you use with any corresponding TCP/IP OTD. Since **sendEnvelopedMsg()** sends the message and drops the connection, if you use additional methods afterward, they only return the error message "Connection was already closed."

Custom

The **Custom** envelope type refers to an envelope type you have created yourself, using a Java class. If want to use the class for a custom envelope, that is, a class you specified under the **Custom Enveloped Class Name** property (see **"Custom Enveloped Class Name" on page 15**), you must select this setting.

Note: *For optimum performance, it is recommended that you use the method* ***receiveEnvelopedMsg()*** *with any enveloped messages, because this method uses the envelope as its ending condition. However, the other receiving methods,* ***receiveBytes()*** *and* ***receiveString()****, use a time-out as their ending condition.*

Required Values

For the **Envelope Type**, enter one of the following properties denoting the envelope type:

- **BeginEndMarked**
- **EndMarked**
- **FixedLength**
- **LengthPrefixed**
- **MarkedAndFixed**
- **PerActiveConnection**
- **Custom**

The default is **BeginEndMarked**.

Ignore Until Char Value

Description

Specifies the value for the ignore-until (same as begin block) character. All incoming characters are ignored until this character is encountered.

Required Values

A byte; the default is **11**.

Numeric Representation

Description

Specifies how the counter is represented numerically. This value is expressed in one of the following formats: decimal, hexadecimal, octal, network short, or network long.

Required Values

Enter **Decimal**, **Hexadecimal**, **Octal**, **Network short**, or **Network long**. **Decimal** representation is the default.

Store Until Char Value

Description

Specifies the store-until (same as end block) character value. All incoming characters are stored until this character is encountered.

Required Values

A byte; the default is **12**.

Width of Length

Description

Specifies the length component in the envelope. This value refers to the number of bytes used to represent the number in the length field of an envelope.

Required Values

An integer; the range is 1 to 10, and the default is **1**. This property must be set to **2** for **Network short** and **4** for **Network long**.

3.2.3 General Server Settings

These properties provide the Dedicated Session Mode and maximum data size message settings for the server. The properties in this section are:

- **"Dedicated Session Mode" on page 20**
- **"Max Data Size" on page 20**

Dedicated Session Mode

Description

Allows you to enable or disable the eWay's Dedicated Session Mode. When the Dedicated Session Mode is enabled in a server, the current client's request can exclusively hold the server port that it connects to.

For example, if this property is enabled, and the client connects to a server, it only serves that client until its work is finished, and the session is disconnected. If another client tries to connect to the server during that time, it cannot until the session is done.

Required Values

True or **False**. **True** enables the option; the default is **False**.

Max Data Size

Description

Allows you to define the maximum size of the data that the programs can hold internally.

Required Values

The valid range is from 1 to 2 GB (the maximum value of the Java integer).

3.2.4 TCP/IP Server Base Settings

These properties allow you to set the basic TCP/IP values for the server. The properties in this section are:

- **"Keep Alive" on page 21**
- **"Receive Buffer Size" on page 21**
- **"Send Buffer Size" on page 21**
- **"Server Socket Factory Implementation Class" on page 21**
- **"ServerPort" on page 27**
- **"ServerSoTimeout" on page 22**
- **"SoLinger" on page 22**
- **"SoLinger Timeout" on page 22**
- **"SoTimeout" on page 22**
- **"TcpNoDelay" on page 22**

Note: *For complete information on options referred to by these base settings, for example, **SO_KEEPALIVE**, see the appropriate Sun Microsystems Java documentation.*

Keep Alive

Description

Specifies whether the server's **SO_KEEPALIVE** option is enabled or disabled; used for the accepted client socket.

Note: *For some properties, the server socket itself does not have direct properties settings associated with it. Instead, the properties map to the accepted client socket.*

Required Values

True or **False**. **True** enables the option.

Receive Buffer Size

Description

Allows you to set or get the value of the server's **SO_RCVBUF** option for the current socket, that is, the buffer size used by the platform for input on the socket; used for the accepted client socket.

Required Values

The receive buffer size; the default is **8192**.

Send Buffer Size

Description

Allows you to set or get the value of the server's **SO_SNDBUF** option for the current socket, that is, the buffer size used by the platform for output on the socket; used for the accepted client socket.

Required Values

The send buffer size; the default is **8192**.

Server Socket Factory Implementation Class

Description

Enter the name of the Java class that implements the server socket factory. This class is used for creating the server socket.

If you have provided your own server socket implementation, you must enter the name here, of the Java class that contains this implementation. The factory implementation class must implement the following interface:

com.stc.connector.tcpip.model.factory.TCPIPSocketFactory

Required Values

A valid Java class name; the default is:

com.stc.connector.tcpip.model.factory.TCPIPSocketFactoryImpl

ServerSoTimeout

Description

Allows you to set or get the server **SO_TIMEOUT** value, in milliseconds.

Required Values

The server's **SO_TIMEOUT** value in milliseconds; the default is **10,000** milliseconds (10 seconds).

SoLinger

Description

Specifies whether the server's **SO_LINGER** option is enabled or disabled; used for the accepted client socket.

Required Values

True or **False**. **True** enables the option.

SoLinger Timeout

Description

Specifies the server's linger time-out in seconds. The maximum time-out value is platform specific. The setting only affects the socket close; used for the accepted client socket.

Required Values

The linger time-out in seconds; the default is **-1** seconds, meaning that the **SO_LINGER** option has been disabled.

SoTimeout

Description

Allows you to set or get the server's **SO_TIMEOUT** value, in milliseconds; used for the accepted client socket.

A time-out of 0 (zero) is an infinite time-out. If you specify this value, the eWay goes into an infinite read. If this action happens, it is recorded in the eWay's log file.

Required Values

The **SO_TIMEOUT** value in milliseconds; the default is **10,000** milliseconds (10 seconds).

TcpNoDelay

Description

Specifies whether the server's **TCP_NODELAY** option (that is, Nagle's algorithm) is enabled or disabled; used for the accepted client socket.

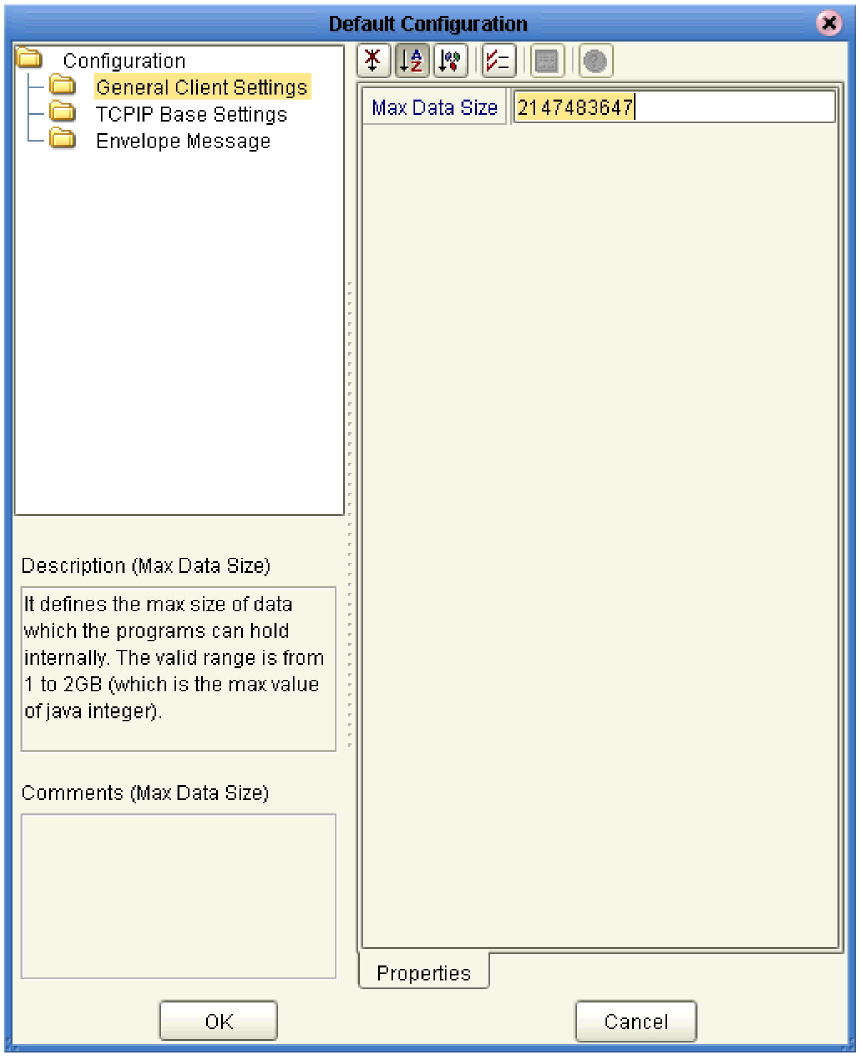
Required Values

True enables the option, and **False** disables it.

3.2.5 Setting eWay Properties in Client Mode

The client (outbound) properties settings determine the eWay's behavior for output operations. See Figure 3.

Figure 3 eWay Properties Dialog Box: Client (Outbound) Settings/Connectivity Map



Max Data Size: 2147483647

Description (Max Data Size)

It defines the max size of data which the programs can hold internally. The valid range is from 1 to 2GB (which is the max value of java integer).

Comments (Max Data Size)

These eWay client mode properties are organized under the following folders:

- **"General Client Settings" on page 24**
- **"TCP/IP Base Settings" on page 24**
- **"Envelope Message (Client)" on page 26**

3.2.6 General Client Settings

Specifies the maximum data size for the client. The property in this section is:

- **"Max Data Size" on page 24**

Max Data Size

Description

Allows you to define the maximum size of the data that the programs can hold internally.

Required Values

The valid range is from 1 to 2 GB (the maximum value of the Java integer).

3.2.7 TCP/IP Base Settings

Specifies the basic TCP/IP values for the client. The properties in this section are:

- **"Keep Alive" on page 24**
- **"Receive Buffer Size" on page 24**
- **"Send Buffer Size" on page 25**
- **"Socket Factory Implementation Class" on page 25**
- **"SoLinger" on page 25**
- **"SoLinger Timeout" on page 25**
- **"SoTimeout" on page 26**
- **"TcpNoDelay" on page 26**

Note: *For complete information on options referred to by these base settings, for example, **SO_KEEPALIVE**, see the appropriate Sun Microsystems Java documentation.*

Keep Alive

Description

Specifies whether the client's **SO_KEEPALIVE** option is enabled or disabled.

Required Values

True or **False**. **True** enables the option.

Receive Buffer Size

Description

Allows you to set or get the value of the client's **SO_RCVBUF** option for the current socket, that is, the buffer size used by the platform for input on the socket.

Required Values

The receive buffer size; the default is **8192**.

Send Buffer Size

Description

Allows you to set or get the value of the client's **SO_SNDBUF** option for the current socket, that is, the buffer size used by the platform for output on the socket.

Required Values

The send buffer size; the default is **8192**.

Socket Factory Implementation Class

Description

Enter the name of the Java class that implements the client socket factory. This class is used for creating a client socket.

If you have provided your own client socket implementation, you must enter the name here, of the Java class that contains this implementation. The factory implementation class must implement the following interface:

com.stc.connector.tcpip.model.factory.TCPIPSocketFactory

Required Values

A valid Java class name; the default is:

com.stc.connector.tcpip.model.factory.TCPIPSocketFactoryImpl

SoLinger

Description

Specifies whether the client's **SO_LINGER** option is enabled or disabled.

Required Values

True or **False**. **True** enables the option.

SoLinger Timeout

Description

Specifies the client's linger time-out in seconds. The maximum time-out value is platform specific. The setting only affects the socket close.

Required Values

The linger time-out in seconds; the default is **-1** seconds, meaning that the **SO_LINGER** option has been disabled.

SoTimeout

Description

Allows you to set or get the client's **SO_TIMEOUT** value, in milliseconds.

Required Values

The **SO_TIMEOUT** value in milliseconds; the default is **10,000** milliseconds (10 seconds).

TcpNoDelay

Description

Specifies whether the client's **TCP_NODELAY** option (that is, Nagle's algorithm) is enabled or disabled.

Required Values

True enables the option, and **False** disables it.

3.2.8 Envelope Message (Client)

These properties are the envelope message format settings for the client. These properties operate in the same way as those for the server (inbound) eWay properties. See **"Envelope Message (Server)" on page 15** for details.

3.3 Setting Properties in the Project Environment

This section explains in detail the eWay's editable properties accessible via the eGate Enterprise Designer's **Environment Explorer**. You can set these properties using the eWay **Properties** dialog box.

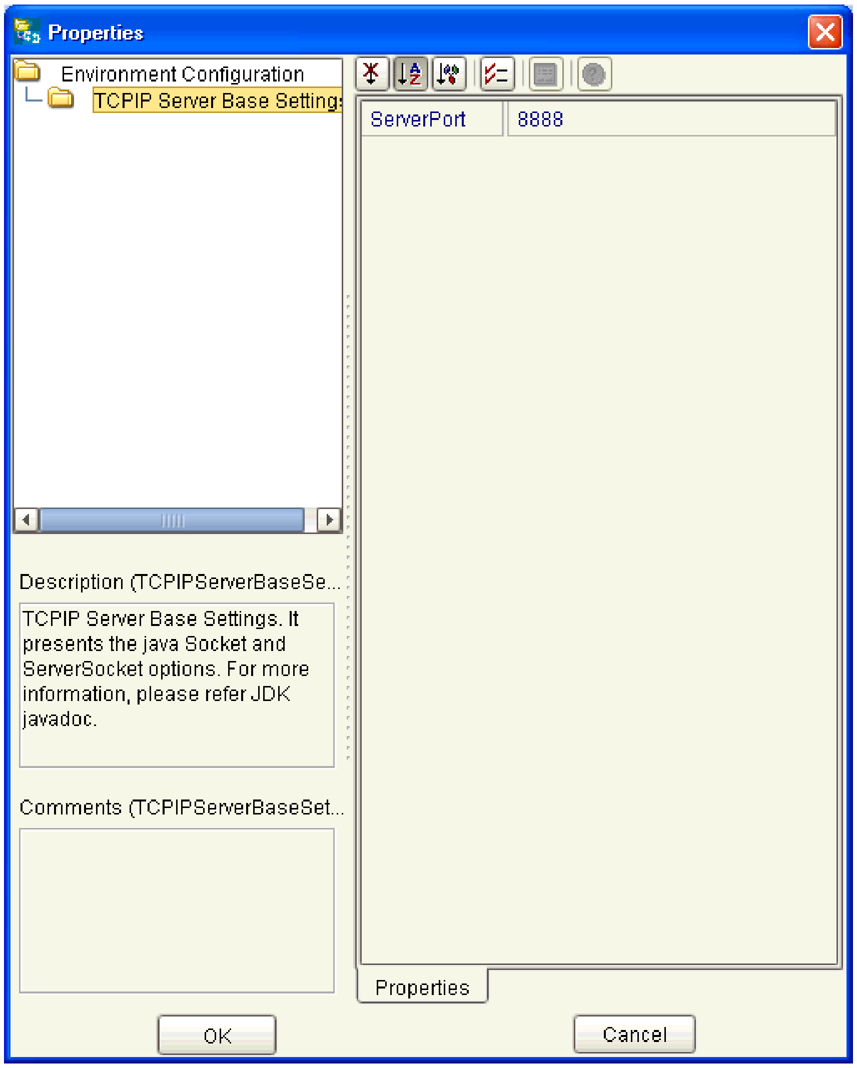
The TCP/IP eWay operates in two modes, client (outbound) and server (inbound). The properties relating to each of these modes are explained under the following sections:

- **"Setting eWay Properties in Server Mode" on page 26**
- **"Setting eWay Properties in Client Mode" on page 28**

3.3.1 Setting eWay Properties in Server Mode

The server (inbound) properties settings determine the eWay's behavior for input operations. See **Figure 4 on page 27**.

Figure 4 eWay Properties Dialog Box: Server (Inbound) Settings/Environment



ServerPort 8888

Description (TCPIPServerBaseSe...

TCPIP Server Base Settings. It presents the java Socket and ServerSocket options. For more information, please refer JDK javadoc.

Comments (TCPIPServerBaseSet...

The eWay server property in the **Environment Explorer** is:

- **“ServerPort” on page 27**

ServerPort

Description

Specifies the TCP/IP port number of the server.

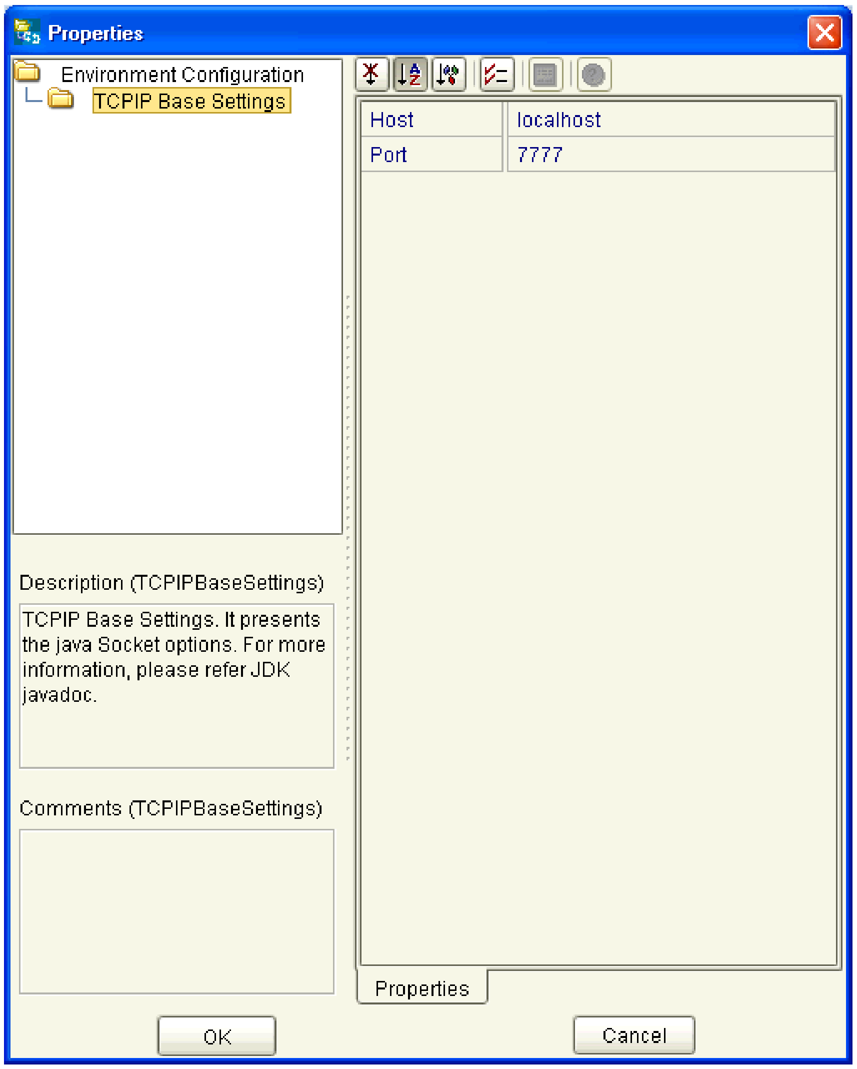
Required Values

A valid port number; the default is **8888**.

3.3.2 Setting eWay Properties in Client Mode

The client (outbound) properties settings determine the eWay's behavior for output operations. See Figure 5.

Figure 5 eWay Properties Dialog Box: Client (Outbound) Settings/Environment



Description (TCPIPBaseSettings)

TCPIP Base Settings. It presents the java Socket options. For more information, please refer JDK javadoc.

Comments (TCPIPBaseSettings)

Host	localhost
Port	7777

These eWay client properties in the **Environment Explorer** are:

- **"Host" on page 28**
- **"Port" on page 29**

Host

Description

Specifies the default server host used (in the client mode) to establish a TCP/IP connection. You can use either the host name or the IP address.

Note: *In the server mode, this property is not necessary because the server gets the client's host name automatically.*

Required Values

A valid TCP/IP host name or IP address; the default is **localhost**.

Port

Description

Specifies the TCP/IP port number of the host. This number represents the port the client connects to. In the server mode, this is the port number the server listens to for inbound connections.

Required Values

A valid port number; the default is **7777**.

Chapter 4

Reviewing the Sample Project

This chapter describes how to implement the TCP/IP eWay by way of reviewing sample Project included with the eWay. The chapter assumes that you are already familiar with eGate concepts and that you understand the basics of creating a Project using the eGate Enterprise Designer.

For a complete explanation of eGate terminology and concepts, including how to use the eGate Enterprise Designer to create and set properties for the components of an eGate Project, see the *eGate Integrator User's Guide*.

Chapter Topics

- **"eGate Project Description" on page 30**
- **"Overview: Sample Project" on page 36**
- **"Summary: Sample Java Collaboration Project" on page 39**
- **"Creating the Project's Environment" on page 40**
- **"Setting eWay Properties" on page 40**
- **"Deploying a Project" on page 43**
- **"Using the TCP/IP OTDs" on page 46**

4.1 eGate Project Description

This section provides an overview of the eGate sample Project for the TCP/IP eWay and how to import the Project into eGate.

4.1.1 Projects and the Enterprise Designer

A Project contains all of the eGate components you designate to perform one or more desired processes in eGate. Each eGate Project is created using the Enterprise Designer.

The Project canvas contains windows that represent the various stages of your Project. The types of windows in your Project canvas area include:

- **Connectivity Map Canvas**: Contains the eGate business logic components, such as Collaborations, Topics, Queues, and eWays, that you include in the structure of the Project.
- **OTD Editor**: Contains the source files used to create Object Type Definitions (OTDs) to use with a Project.
- **Collaboration Editor (Java)**: Allows you to create and/or modify Business Rules to implement the business logic of a Project's Java-enabled Collaboration Definitions.

4.1.2 Importing Sample Projects

Before you can view or work with a sample Project, you must first import it into eGate, using the Enterprise Designer.

Note: *The sample **.zip** file you first download may contain more than one Project and/or additional files. If this is the case, you must unzip this file first, find the desired Project file, then import the Project file. For information on how to obtain this file, see* **Chapter 2**. *For complete instructions on downloading and installing the eWay and related files, see the **SeeBeyond ICAN Suite Installation Guide**.*

The container file you are looking for is **TCPIP_eWay_Sample.zip**. The Project file name is:

TCPIP_SampleProject.zip: Java-based Collaboration Project

You can name imported Projects as desired.

To import a sample Project

1. Save any changes not saved previously.
2. From the Enterprise Designer's **Project Explorer** pane, right-click the desired Repository and select **Import**.
3. On the **Import Manager** window, browse to the directory that contains the sample Project **.zip** file.
4. Select the sample Project file and click **Open**.
5. Click **Import**. If the import was successful, click **OK** on the **Import Status** dialog box.
6. Close the **Import Manager** window.

Important: *An imported Project does not contain an environment or a deployment profile. After importing a Project, you must use the Enterprise Designer to create these functions for the Project. See* **"Creating the Project's Environment" on page 40** *and* **"Deploying a Project" on page 43**. *For additional information, see the **eGate Integrator User's Guide** and **SeeBeyond ICAN Suite Deployment Guide**.*

You must check out the major eGate components before you can change them. For details, see the *eGate Integrator User's Guide*.

4.1.3 Basic eWay Components

To use the TCP/IP eWay, you must:

- Set the desired properties settings.
- Use the TCP/IP client and server OTDs.

The sections that describe how to build the Project explain more about how to do these operations.

TCP/IP eWay Properties

The properties for the TCP/IP eWay allow you to edit values used to connect with a specific external system. These properties are set using the eGate eWay **Properties** dialog box. For more information about setting TCP/IP eWay properties and the eWay **Properties** dialog box, see **Chapter 3**.

TCP/IP OTDs

The TCP/IP client and server OTDs are part of the eWay and ready-made for your use in a Project. See **"Using the TCP/IP OTDs" on page 46** for an explanation of the features available to you with these OTDs.

4.1.4 Customized Enveloping

The eWay properties settings provide ready-made message enveloping you can use if desired. However, if these predefined envelopes do not meet your needs, you can create your own Java class defining a custom message envelope.

Note: *See the **Javadoc** provided with this eWay for complete information on the eWay's Java classes and methods.*

For more information on the eWay's enveloping feature, see **"Envelope Type" on page 15**.

The eWay allows you to set properties to implement your customized enveloping. These properties are explained under:

- **"Custom Enveloped Class Name" on page 15**
- **"Customer Defined Property" on page 15**

To create a custom envelope

1. Using the Java Software Developer's Kit (SDK), create the class that defines your custom message envelope, including the desired custom features. This new class must implement the following Java interfaces provided in the eWay:
    - **com.stc.connector.tcpip.ext.msg.EnvelopedMsgReceiver**
    - **com.stc.connector.tcpip.ext.msg.EnvelopedMsgSender**

2 For your implemented Java interface (see the previous list), you can import/use the following Java classes:

- **com.stc.connector.appconn.tcpip.ext.TCPIPEXTClientApplication**
- **com.stc.connector.message.envelope.EnvelopedMsg**
- **com.stc.connector.tcpip.model.exception.TCPIPApplicationException**

You can find the **.jar** files containing these classes in the following eGate installation directory:

edesigner\usrdir\modules\ext\tcpipextadapter

For example:

C:\ican50\edesigner\usrdir\ ...

Note: These classes are provided with the eWay. You can create your own classes, if desired.

3 Be sure to include the **.jar** files discussed under step 2 in the Java CLASSPATH for compiling the class.

4 Create, compile, and package a **.jar** file that contains your custom envelope class, for example, **customenvelope.jar**.

5 In the eWay Properties window for the eWay, you must set the following properties:

- **Envelope Type**: **Custom**
- **Custom Property**: Any property that needs to be defined in order for your custom envelope to work, for example, delimiters.
- **Custom Enveloped Class Name**: For example:

  com.stc.customenvelope.CustomEnvelope1

You must enter the full path location of the custom envelope class.

Note: For details on the eWay Properties window, see **Chapter 3**.

Next, you must include your custom envelope in the current eGate Project. To do this operation, you must first create a Repository object that contains your custom envelope's **.jar** file, then add that file to the CLASSPATH of any Collaboration that uses this envelope.

To use a custom envelope in a Project

1 From the Enterprise Designer's **Project Explorer** pane, right-click the desired Project and choose **File** from the pop-up menu.

2 From the resulting dialog box, choose your custom envelope's **.jar** file (see the **procedure on page 32**) and click **Import**.

This action creates a Repository object containing your custom envelope's **.jar** file and makes the file available in the **Project Explorer**.

3 From the Collaboration Editor (Java) for any Collaboration using your custom envelope, click the **File Import** button on the toolbar.

4 Again, from the resulting dialog box, choose your custom envelope's **.jar** file and click **Add**.

   This action adds your customized envelope's class to the current Collaboration's CLASSPATH.

For details on how to use the Enterprise Designer's file import features, see the *eGate Integrator User's Guide*.

Custom envelope sample

The following text provides a sample of a custom envelope:

```
package com.stc.customenvelope;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.log4j.Logger;

import com.stc.connector.appconn.tcpip.ext.TCPIPEXTClientApplication;
import com.stc.connector.message.envelope.EnvelopedMsg;
import com.stc.connector.tcpip.model.exception.TCPIPApplicationException;
import com.stc.connector.tcpip.ext.msg.EnvelopedMsgReceiver;
import com.stc.connector.tcpip.ext.msg.EnvelopedMsgSender;

/**
 * Custom envelope message sample:
 * This sample demonstrates the use of custom message
 * enveloping. The sample uses the following envelope parameters:
 * offset, len, and delimeter.
 * They are specified in the eWay configuration parameter
 * as the string "offsetdelimeterlen",
 * for example, "4&10",
 * where:
 * offset = 4   len = 10   delimeter = &.
 * You can assign values to offset and len, but delimeter must be "&".
 * The eWay parses the data and assigns
 * the given values to
 * offset and len, then reads data on the socket from the "offset" position
 * until it reaches the end-of-length value "len".
 */
public class CustomEnvelopedSample
    implements EnvelopedMsgReceiver, EnvelopedMsgSender {
    private Logger logger = Logger.getLogger("STC.eWay.TCPIP." + getClass().getName())
;
    private String logMsg;

    // class variables
    private int offset;
    private int len;

    /**
      * Constructor for CustomEnvelopedSample.
     */
    public CustomEnvelopedSample() {

        this.offset = 0;
        this.len = 0;
    }
```

```
/**
  * @see com.stc.connector.tcpip.ext.msg.EnvelopedMsgReceiver#receiveEnvelopedMsg
  * (TCPIPEXTClientApplication)
 */
public EnvelopedMsg receiveEnvelopedMsg(TCPIPEXTClientApplication app)
    throws TCPIPApplicationException, IOException {
    // do additional validation if needed.


 /* The following code parses the eWay custom property given as 4&10,
  * where 4 = offset, 10 = len, and & = delimeter.
  */

 String config = app.getMessageInfo().getCustomerDefinedProperty();
 StringTokenizer st = new StringTokenizer(config, "&");
 if (st.hasMoreTokens()) {
           this.offset = Integer.parseInt(st.nextToken());
           }
           if (st.hasMoreTokens()) {
           this.len = Integer.parseInt(st.nextToken());
      }

    int readLen = 0;


    byte[] ignore = new byte[this.offset];
    readLen = app.getSocket().getInputStream().read(ignore);

    if (readLen != this.offset) {
        logMsg =
            "CustomEnvelopedSample.receiveEnvelopedMsg(): "
                + "Invalid data/
offset, no enough data is available on the socket.";
        logger.error(logMsg);
        throw new TCPIPApplicationException(logMsg);
    }


    byte[] bytes = new byte[this.len];
    readLen = app.getSocket().getInputStream().read(bytes);

    if (readLen != this.len) {
        logMsg =
            "CustomEnvelopedSample.receiveEnvelopedMsg(): "
                + "Invalid data/len, no enough data is available on the socket.";
        logger.error(logMsg);
        throw new TCPIPApplicationException(logMsg);
    }

    return this.new MyEnvelopedMsgImpl(bytes);
}

/**
  * @see com.stc.connector.tcpip.ext.msg.EnvelopedMsgSender#sendEnvelopedMsg
  * (EnvelopedMsg, TCPIPEXTClientApplication)
 */
public void sendEnvelopedMsg(
    EnvelopedMsg msg,
    TCPIPEXTClientApplication app)
    throws TCPIPApplicationException, IOException {
    // do additional validation if needed.
    app.sendBytes(msg.getBytes());
}

/** An inner classe that implements an enveloped message.
  */
private class MyEnvelopedMsgImpl extends EnvelopedMsg {
    private byte[] data;

/** Constructor for MyEnvelopedMsgImpl.
 * @param data byte[]
 */
    public MyEnvelopedMsgImpl(byte[] data) {
        super(null);
        this.data = data;
        //this.setEnvelopeType("Custom");
    }
```

```
            /** @see com.stc.connector.message.common.AbstractMsg#getBytes()
             */
            public byte[] getBytes() throws IOException {
                return this.data;
            }
        }
    }
```

This custom envelope is used in the eWay's sample Project. For more information, see **"Custom Envelope" on page 39**.

4.2 Overview: Sample Project

The TCP/IP eWay sample Project demonstrates how the TCP/IP eWay processes information to and from a TCP/IP system. The resulting information is then written to a text file.

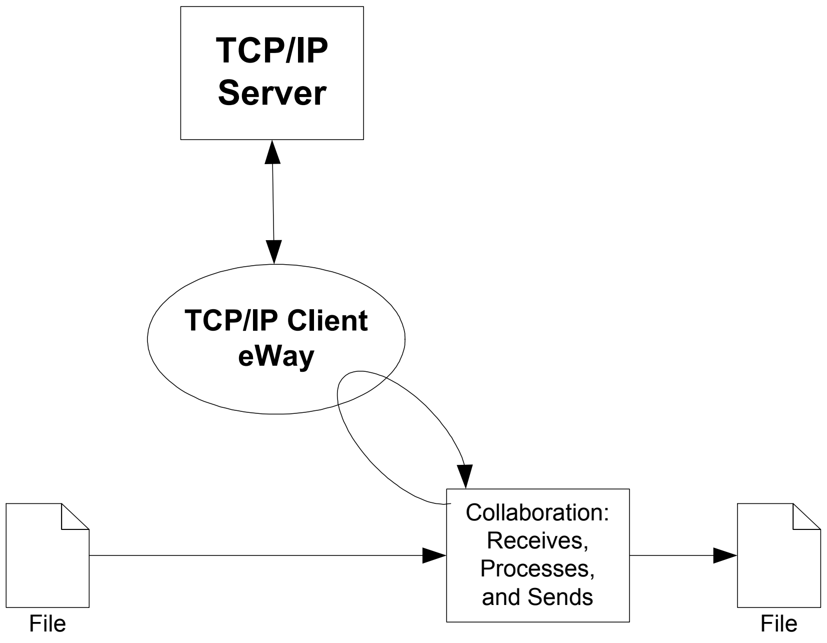
4.2.1 Project: General Scenarios

This section describes how the sample Project operates using scenarios that employ the eWay's two modes, client and server.

Client Mode

Figure 6 shows a general diagram of how the TCP/IP eWay operates in the scenario used by the Project sample, in the client mode. This sample allows eGate to communicate with both a TCP/IP server.

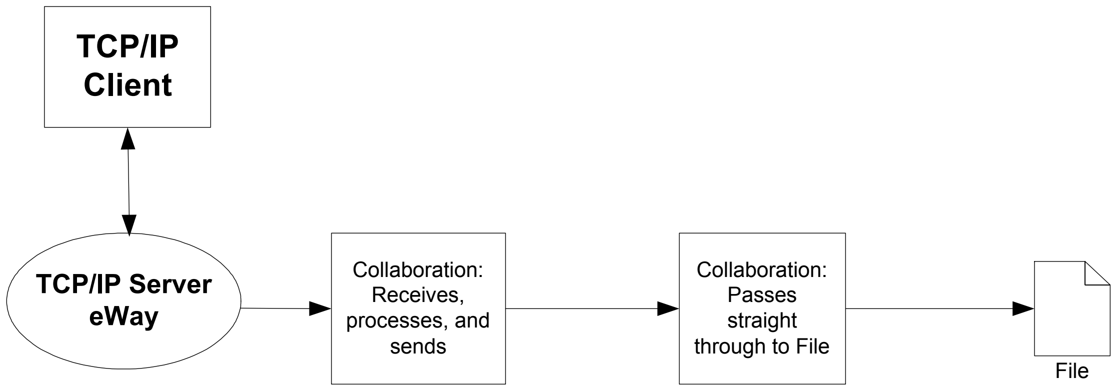
Figure 6 TCP/IP Client Mode Scenario



Server Mode

Figure 7 shows a general diagram of how the TCP/IP eWay operates in the scenario used by the Project sample, in the server mode. This sample allows eGate to communicate with both a TCP/IP client.

Figure 7 TCP/IP Server Mode Scenario


TCP/IP Client
TCP/IP Server eWay
Collaboration: Receives, processes, and sends
Collaboration: Passes straight through to File
File

Important: *To be sure the eWay can receive multiple messages in the server mode, you must set up your server Java-based Collaboration Business Rules with additional looping logic to make the Collaboration longer-lived. For example, you can add a **while** loop as an initial Business Rule, which is set to make multiple tries to receive any incoming message until a quit condition is met. This condition could be, for example, a message counter that confirms the message has been received.*

4.2.2 Sample Project General Operation

The TCP/IP eWay sample Project demonstrates how the TCP/IP eWay processes information to and from a client, as well as how it processes information from a server. The resulting information is then written to a text file.

Project Components

The Project has the following components:

- Client scenario
  - Inbound file external application: **File1**
  - Inbound File eWay
  - Collaboration (Java) for processing data: **file2TCPIPClientCollab**
  - TCP/IP eWay
  - TCP/IP external application: **TCPIP1**
  - Outbound File eWay
  - Outbound file external application: **File2**

- Server scenario
    - Inbound TCP/IP external application: **TCPIP2**
    - TCP/IP eWay
    - Collaboration (Java) for processing data: **TCPIPServer2FileCollab**
    - Outbound File eWay
    - External outbound file system: **File3**

Project Operation

The client part of the Project operates as follows:

- Data enters eGate from a file system via a File eWay; information needs to be communicated to a TCP/IP server.
- The data becomes an eGate message and is sent to a Collaboration (Java) for processing
- The message is sent to the TCP/IP server through a TCP/IP eWay.
- The desired information is sent and received; information returns to eGate through the eWay.
- The information is processed again by the same Collaboration then written to an external file system, again via a File eWay.

The server part of the Project operates as follows:

- Data enters eGate from a TCP/IP client via the TCP/IP eWay.
- The data becomes an eGate message and is passed to a Collaboration (Java) where it is processed.
- The information is written to an external file system via a File eWay.

Input and Output Data

The sample includes the following input file for both client and server scenarios:

input.~in

The sample includes the following client scenario output example file:

ClientOutput.txt

The sample includes the following server scenario output example file:

ServerOutput.txt

Handshake Protocol

The TCP/IP client checks the connection state, and if the state is null or if the state is **not connected**, the client sends the user name and password. The TCP/IP server checks the user name and password. If they are correct, the server sends a positive acknowledgement and changes the state to **connected**. Otherwise, the server sends a negative acknowledgement.

The client checks the message from the server to check whether the message is a positive or negative acknowledgement. If the state is **connected**, the client sends the data to TCP/IP server, and the server receives the data.

Custom Envelope

The sample Project uses a custom envelope. For details on the eWay's custom enveloping feature and the custom envelope used in this sample, see **"Customized Enveloping" on page 32**.

This custom envelope has two properties, offset and length. The custom envelope parses the incoming data, starting from the offset position and continuing until the stated length has been reached.

4.3 Summary: Sample Java Collaboration Project

This section explains generally how to implement the TCP/IP eWay using the eGate Project sample that includes a Java-based Collaboration. This sample is included on your installation CD-ROM.

The extracted sample Project file is named **TCPIP_SampleProject.zip**. This Project allows you to observe an end-to-end data-exchange scenario involving eGate and the TCP/IP eWay.

For instructions on how to import the sample Project, see the **procedure on page 31**. For an overview of the sample Project and what is does, see **"Overview: Sample Project" on page 36**.

See the *eGate Integrator User's Guide* for details on how to build an end-to-end Project in eGate.

4.3.1 Using the eWay With Java-based Collaborations

Java-based Collaborations in eGate use the TCP/IP OTDs to process TCP/IP data within the eGate system. Using Collaboration Definitions and these OTDs, you can allow the eWay to access the desired TCP/IP functions.

See **"Using the TCP/IP OTDs" on page 46** for details on the TCP/IP OTDs.

Creating Business Rules Within Collaboration Definitions

A Collaboration Editor (Java) allows you to create the Business Rules that implement your business logic for a Java-based Collaboration Definition.

Collaboration Editor (Java) Window

The Collaboration Editor (Java) window displays in the Enterprise Designer after you create a Java-based Collaboration Definition. You can also open this editor by right-clicking on the name of the desired Collaboration Definition in the **Project Explorer** and choosing **Open** from the shortcut menu.

To complete a Java-based Collaboration Definition, you can use this editor to create Business Rules.

You can create the desired Business Rules for your Project by dragging and dropping values from a source OTD onto the nodes of a destination TCP/IP OTD and other OTDs. TCP/IP OTD nodes represent TCP/IP functions, which are in turn able to call TCP/IP methods.

See the *eGate Integrator User's Guide* for complete information on how to use the Collaboration Editor (Java).

4.3.2 Creating the Project's Environment

This section provides general procedures for creating an Environment for your Project. For a complete explanation, see the *eGate Integrator User's Guide.*

To create an Environment

1. From the Enterprise Designer, click the **Environment Explorer** tab on the Enterprise Explorer.
2. Under the current **Repository** icon in the **Environment Explorer**, create a new Environment for your Project and name it as desired.
3. In the **Environment Explorer**, right-click the **Environment** icon and select the desired external systems from the shortcut menu. Include external systems for the TCP/IP eWay and the File eWays (inbound and outbound). Give them the same names as you did the corresponding external applications on the Connectivity Map.
4. Use the same shortcut menu to create a Logical Host for your Project, and name it as desired.
5. Click **Save** and return to the **Project Explorer** tab.

4.3.3 Setting eWay Properties

You must set the eWay properties for your specific system and for the current Project, using the eGate Enterprise Designer. For directions on accessing and using the eWay **Properties** dialog box, as well as a complete explanation of the TCP/IP eWay properties, see **Chapter 3**.

To set properties for the File eWays

1. From the **Project Explorer**, open the Connectivity Map for the sample Project.
2. To change the default properties for the inbound File eWay, double-click the **FileIn** external system's eWay icon. For this eWay, select the **Inbound** properties.

The eWay **Properties** dialog box appears. This eWay can use the current default properties settings for the sample Project. However, if you are using different file names and/or folders than the defaults, you must enter the names of the files/folders you are using.

Note: *Even if you do not change the eWay's properties, you must open each **Properties** dialog box for every eWay and click **OK** to activate the eWay.*

3 For this sample, use the **C:\temp** as the property for **Directory**, and for **Input file name**, enter ***.txt**.

4 Click **OK** to save the settings and close the eWay **Properties** dialog box.

5 To change the default properties for the outbound File eWay, double-click the **FileOut** external system's eWay icon. For this eWay, select the **Outbound** properties.

Use your file and folder names if they differ from the defaults. For all other properties, you can use the defaults.

6 Click **OK** to close the properties dialog box and save your changes.

To set the TCP/IP eWay properties

1 To begin changing the default properties for the TCP/IP eWay, click the **TCPIP1** external application's eWay icon on the Connectivity Map. For this eWay, select the **Client** properties.

The eWay **Properties** dialog box appears.

2 From the upper left pane of the eWay **Properties** dialog box, select the desired folder.

3 The properties settings appear in the **Properties** pane on the left.

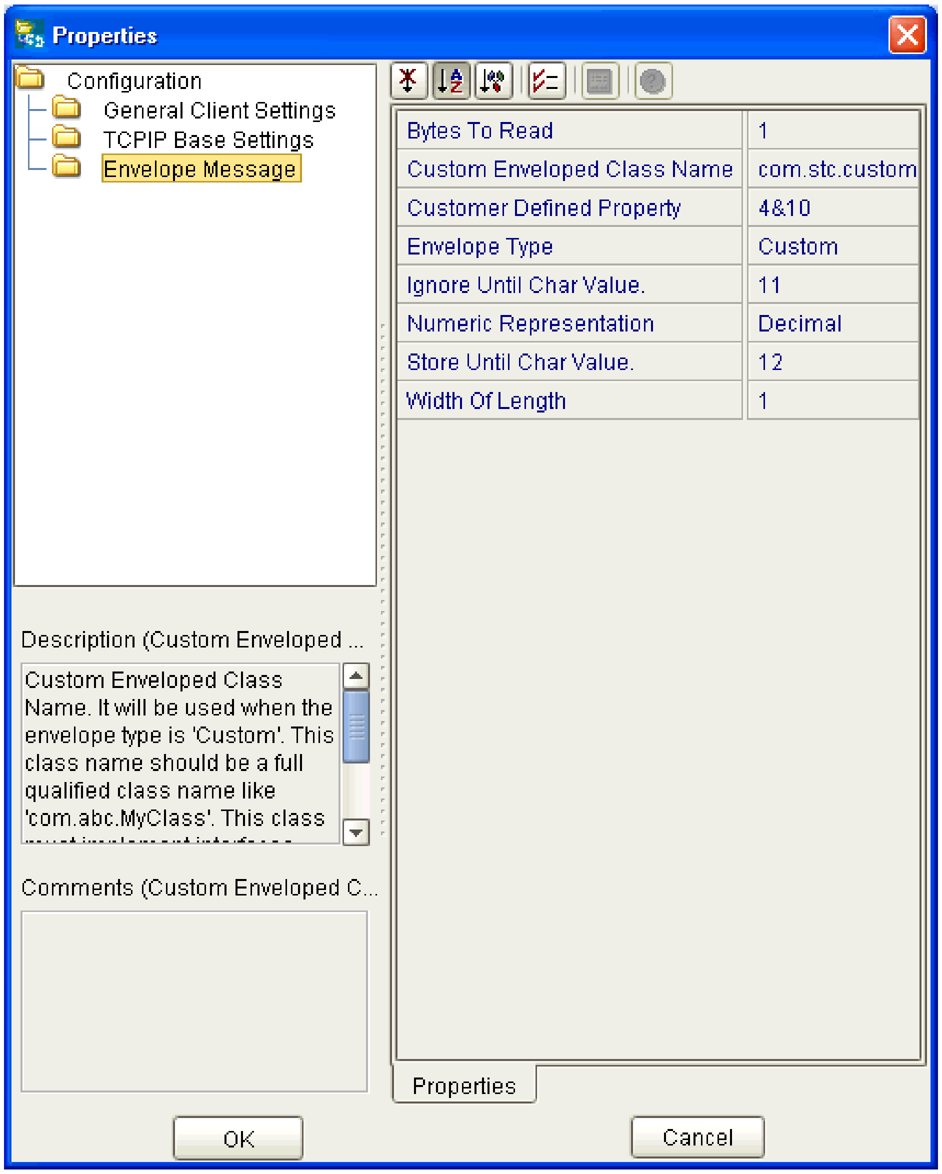
4 You can set the properties to the defaults except for the custom enveloping. Use the custom envelope file provided with the sample. For details on how to create and use this file, see **"Customized Enveloping" on page 32**.

For both the server and client eWays, set the custom envelope properties as follows:

- **Custom Enveloped Class Name: com.stc.customenvelope.CustomEnvelopedSample**
- **Customer Defined Property: 4&10**

See **Figure 8 on page 42**.

Figure 8 Custom Envelope Properties



Bytes To Read	1
Custom Enveloped Class Name	com.stc.custom
Customer Defined Property	4&10
Envelope Type	Custom
Ignore Until Char Value.	11
Numeric Representation	Decimal
Store Until Char Value.	12
Width Of Length	1

Description (Custom Enveloped ...

Custom Enveloped Class Name. It will be used when the envelope type is 'Custom'. This class name should be a full qualified class name like 'com.abc.MyClass'. This class

Comments (Custom Enveloped C...

5 Click **OK** to close the window and save.

6 To change the default properties for the next TCP/IP eWay, click the **TCPIP2** external application's eWay icon. For this eWay, select the **Server** properties.

The eWay **Properties** dialog box appears.

7 From the upper left pane of the eWay **Properties** dialog box, select the desired folder.

8 The properties settings appear in the **Properties** pane on the left.

9 Set the properties for file and folder settings in the same way as you did for the client mode. Use the same custom enveloping settings. For the other settings, you can use the defaults.

10 Click **OK** to close the window and save.

11 From the Enterprise Designer, click the **Environment Explorer** tab.

12 In the **Environment Explorer**, right-click the new Environment you created for your Project, and select **New TCPIP External System** for each system, client and server. Give these systems the same names as you gave the corresponding external applications (**TCPIP1** and **TCPIP2**) created on the Connectivity Map in the **Project Explorer**.

If you have already done these operations, you can skip this step.

13 In the left pane, right-click the **TCPIP2** (server) external application icon and select **Properties** from the pop-up menu.

The eWay **Properties** dialog box appears.

14 From the upper left pane of the eWay **Properties** dialog box, select the **Environment Configuration** folder.

15 The properties settings appear in the **Properties** pane on the left.

16 For the settings, use the defaults, *except* be sure that **ServerPort** is set to **8888**.

17 Click **OK** to close the window and save.

18 In the left pane, right-click the **TCPIP1** (client) external application icon and select **Properties** from the pop-up menu.

The eWay **Properties** dialog box appears.

19 For the settings, use the defaults, *except* be sure that **Port** is set to **8888**.

20 Click **OK** to close the window and save.

4.4 Deploying a Project

This section provides general procedures for Project deployment.

4.4.1 Basic Steps

For a complete explanation of how to deploy and run an eGate Project, see the *eGate Integrator User's Guide.*

To deploy the Project

1 From the **Project Explorer**, select the current Project and right-click, choosing **New > Deployment Profile** from the shortcut menus.

2 From the **Create a Deployment Profile** dialog box, enter the name of the current Project and select the Environment you created for this Project.

3 Click **OK**.

The Deployment Profile canvas appears as follows:

- The Project's external applications and Services show up as items on the left side of the canvas.
- The external systems and Logical Host you created under **"Creating the Project's Environment" on page 40** show up as windows on the right side of the canvas.

4. Set up your Deployment Profile by dragging the items on the left into the corresponding window on the right.
5. Click **Save All** then **Activate**.

   When the Project has been activated, a pop-up message appears stating the activation was successful.

For additional information, see the *SeeBeyond ICAN Suite Deployment Guide.*

To run the Project

For instructions on how to run a Project, see the *eGate Integrator User's Guide*.

4.4.2 For and While Rules for Monitoring

It is a good practice to create a **for** and/or **while** rule (a loop) in any Collaboration (Java) using the server mode, if you want to start and stop the eWay frequently. This addition allows you to use the eGate Monitor to start and stop the eWay without problems. For more information, see the *eGate Integrator User's Guide*.

For more information on how to create these rules, see the *eGate Integrator User's Guide*.

4.4.3 Alerting and Logging

eGate provides an alerting and logging feature. This feature allows monitoring of messages and captures any adverse messages in the order of severity, based on a configured severity level and higher. For details on how to enable the Logging feature, see the *eGate Integrator User's Guide*.

4.4.4 Deploying on HP NonStop Server Systems

For instructions on how to deploy eGate Projects on HP NonStop server systems, see the *SeeBeyond ICAN Suite Deployment Guide*.

Parallel Library TCP/IP

Parallel Library TCP/IP allows all processors in an HP NonStop server system to access a given physical TCP/IP adapter. This feature routes message packets directly to the processor containing the desired application. By routing packets directly to the correct processor from the adapter, Parallel Library TCP/IP eliminates the message-system hop that occurs between processes in conventional TCP/IP.

The eGate system for HP NonStop server lets you use Parallel Library TCP/IP with the TCP/IP eWay.

Note: *For more information on the HP NonStop server Parallel Library TCP/IP feature, see the appropriate HP documentation.*

You can set the Parallel Library TCP/IP feature using the **lh.profile.sample** file located in the following eGate directory:

eGate/logical_host/**bootstrap/bin**

Where:

- ***eGate*** is the directory where you installed eGate.
- ***logical_host*** is the directory where you installed the logical host.

Using the File

The contents of the **lh.profile.sample** file have a sectional organization. Lines that are not to be used are commented out with a "#" symbol. Refer to section **2.1 TCPIP environment** in this file for the lines that enable the Parallel Library TCP/IP feature. Conventional TCP/IP, without this feature, is the default (see section **2.1.1 CONVENTIONAL TCP/IP** in the file), so the line in the file for this option is not commented out.

The file also contains definitions and instructions for its use, as shown here:

```
####################################################################
# File:  lh.profile.sample
# Purpose:To set environment variables and =DEFINE's for a
#             logical host.
####################################################################
#
# 1.0 Environment variables
#
export JAVA_HOME=/usr/tandem/java
export NSJMS_HOME=/usr/tandem/nsjms/jms
export NSDOM_CFG_DBM=\$DATA04.UNDM.NSDCFGDB
export PATH=$JAVA_HOME/bin:$PATH
#
# 2.0 Defines
#
# To prevent loss of communication with SQLMP compiler in NSJMS
add_define =_SQL_CMP_TIMEOUT class=map file=\$DATA02.NSJMS.DURABLE
#
# 2.1 TCPIP environment: Please choose only one.
#
# 2.1.1 CONVENTIONAL TCP/IP: Set =TCPIP^PROCESS^NAME to TCPIP process
name
#
add_define =TCPIP^PROCESS^NAME class=map file=\$ZTC0
#
# 2.1.2 PARALLEL TCP/IP: Set =TCPIP^PROCESS^NAME to TCPSAM process
name
# add_define =TCPIP^PROCESS^NAME class=map file=\$TSAM2
#
####################################################################
# EOF
####################################################################
```

As the file's text indicates, you can choose one of the following Parallel Library TCP/IP options:

- **Section 2.1.1**: These lines (the default) allow you to use conventional TCP/IP.
- **Section 2.1.2**: These lines allow you to use Parallel Library TCP/IP in the same way as conventional TCP/IP, that is, without TCP/IP port sharing.

  To use this feature, comment out the following line:

  ```
  add_define =TCPIP^PROCESS^NAME class=map file=\$ZTC0
  ```

  Then uncomment this line:

  ```
  add_define =TCPIP^PROCESS^NAME class=map file=\$TSAM2
  ```

Keep in mind that other logical host TCP/IP ports that you may not want to be shared still can be shared. In your overall eGate deployment, be sure to set only the port number properties for the TCP/IP eWays that you want to be shareable. At the same time, be sure to set all the logical hosts' nonsharing TCP/IP ports to have different port numbers that you do *not* want to be shared.

Note: *See the SeeBeyond ICAN Suite Installation Guide for more information on this file.*

4.5 Using the TCP/IP OTDs

You can use the TCP/IP OTDs via the eGate Enterprise Designer to set up the eWay as desired. The Collaboration Definition Editors and OTD Editor allow you to access OTD features via Windows drag-and-drop operations.

4.5.1 OTD Overview

An OTD contains a set of rules that define an object. The object encodes data as it travels through eGate. OTDs are used as the basis for creating Collaboration Definitions for a Project.

Each OTD acts as a template with a unique set of eWay features. The TCP/IP eWay OTD template is not customizable and cannot be edited.

The four parts of an OTD are:

- **Element**: This is the highest level in the OTD tree. The element is the basic container that holds the other parts of the OTD. The element can contain fields and methods.
- **Field**: Fields are used to represent data. A field can contain data in any of the following formats: string, boolean, int, double, or float.
- **Method**: Method nodes represent actual Java methods.
- **Parameter**: Parameter nodes represent the Java methods' parameters.

4.5.2 OTD Components

Each of the OTDs is made up of the following components:

- **OTD Operation**: The OTD is used in a Collaboration Definition to operate with eWays.
- **Definition and eGate Enterprise Designer**: An eWay Properties window provides a central location where you can define the eWay's properties. The Enterprise Designer also allows you to access this window.
- **eWay**: An eWay provides access to the information necessary to interface with a specified external application.

All OTDs must be set up and administered using the Enterprise Designer.

Note: *For complete information on how to use the Enterprise Designer and the eWay Properties window, see the* ***eGate Integrator User's Guide****.*

4.5.3 Basic OTD Operation

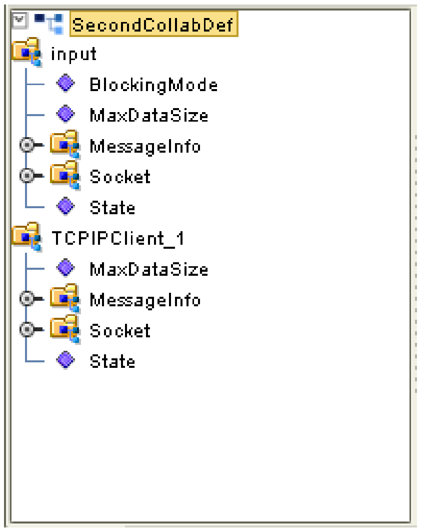
The TCP/IP eWay employs the following OTDs:

- **Server**: For TCP/IP *inbound* communication
- **Client**: For the TCP/IP *outbound* communication

These OTDs can perform inbound and outbound communication in a Collaboration either one direction at a time or simultaneously. Each OTD represents a deployable RA that includes TCP/IP (socket) functions plus an enveloped message model. The eWay's TCP/IP OTDs provide a basic TCP/IP client-and-server model that can handle all of the eWay's Java 2 Enterprise Edition (J2EE)-related connections.

Figure 9 shows the two OTDs' basic structure as it appears in the Collaboration Definition (Java) Editor window. The OTD with the root node labeled **input** is the server OTD, and the one with the **TCPIPClient_1** root node is the client. You can expand the nodes **MessageInfo** and **Socket** to reveal additional sub-nodes.

Figure 9 Collaboration Definition (Java): TCP/IP OTDs



4.5.4 TCP/IP OTD Features

The eWay's TCP/IP server and client OTDs provide the following features:

- **Java Socket Functions**: Allows you to expose a Java socket object via J2EE connection management, using whatever the **java.net.Socket** interface provides.
- **Predefined Message Envelope Types**: Several types of enveloped messages (over a TCP/IP connection) are provided with the eWay. See **"Envelope Type" on page 15** for details.
- **Extensibility for Enveloped Messages**: If the predefined message envelope types (over a TCP/IP connection) cannot meet your demands, you can provide your own customized enveloping. The eWay has corresponding defined interfaces for this purpose. See **"Customized Enveloping" on page 32** for details. Also, a sample customized envelope is provided.
- **Socket Property Options**: You can set these properties (for example, time-out and buffer size) statically using the eWay **Properties** dialog box or dynamically using methods in an OTD.

- **Message Property Options**: You can set these properties (for example, envelope type and beginning marker) statically using the eWay **Properties** dialog box or dynamically via methods in an OTD.
- **Single-port Connections**: The TCP/IP server OTD can handle multiple, virtually unlimited connections asynchronously, on a single port. Each work is a separate thread.
- **Dedicated Session Mode**: This feature, when enabled in a server, allows the current client's request to exclusively hold the server port that it connects to. See **"Dedicated Session Mode" on page 20** for details.
- **Server Endpoint Property Options**: You can set the eWay's server socket and message properties as desired.

Chapter 5

Using eWay Java Methods

This chapter provides an overview of the Java classes/interfaces and methods contained in the TCP/IP eWay. These methods are used to extend the functionality of the eWay.

Chapter Topics

- **"TCP/IP eWay Methods and Classes: Overview" on page 50**
- **"eWay Java Classes/Interfaces" on page 51**

5.1 TCP/IP eWay Methods and Classes: Overview

The TCP/IP eWay exposes various Java methods to add extra functionality to the eWay. These methods make it easier to set information in the TCP/IP eWay Object Type Definitions (OTDs), as well as get information from them.

5.1.1 Relation to eWay Configuration

The nature of this data transfer depends on the properties you set for the eWay in the eGate Enterprise Designer's eWay **Properties** dialog box. For more information on the eWay's properties settings, see **Chapter 3**.

5.1.2 TCP/IP eWay Javadoc

For a complete list of the Java methods within the classes listed in this chapter, refer to the **Javadoc**. You can download the Javadoc while you are installing the eWay. For complete instructions, see the *SeeBeyond ICAN Suite Installation Guide*.

5.2 eWay Java Classes/Interfaces

The TCP/IP eWay provides the following Java classes/interfaces:

- **TCPIPEXTClientApplication**: Extends **TCPIPClientApplication** with the message envelope interface for TCP/IP.
- **TCPIPEXTServerApplication**: Extends **TCPIPServerApplication** with the message envelope interface for TCP/IP.
- **TCPIPClientApplication**: Represents the TCP/IP client application interface.
- **TCPIPServerApplication**: Represents the TCP/IP server application interface.

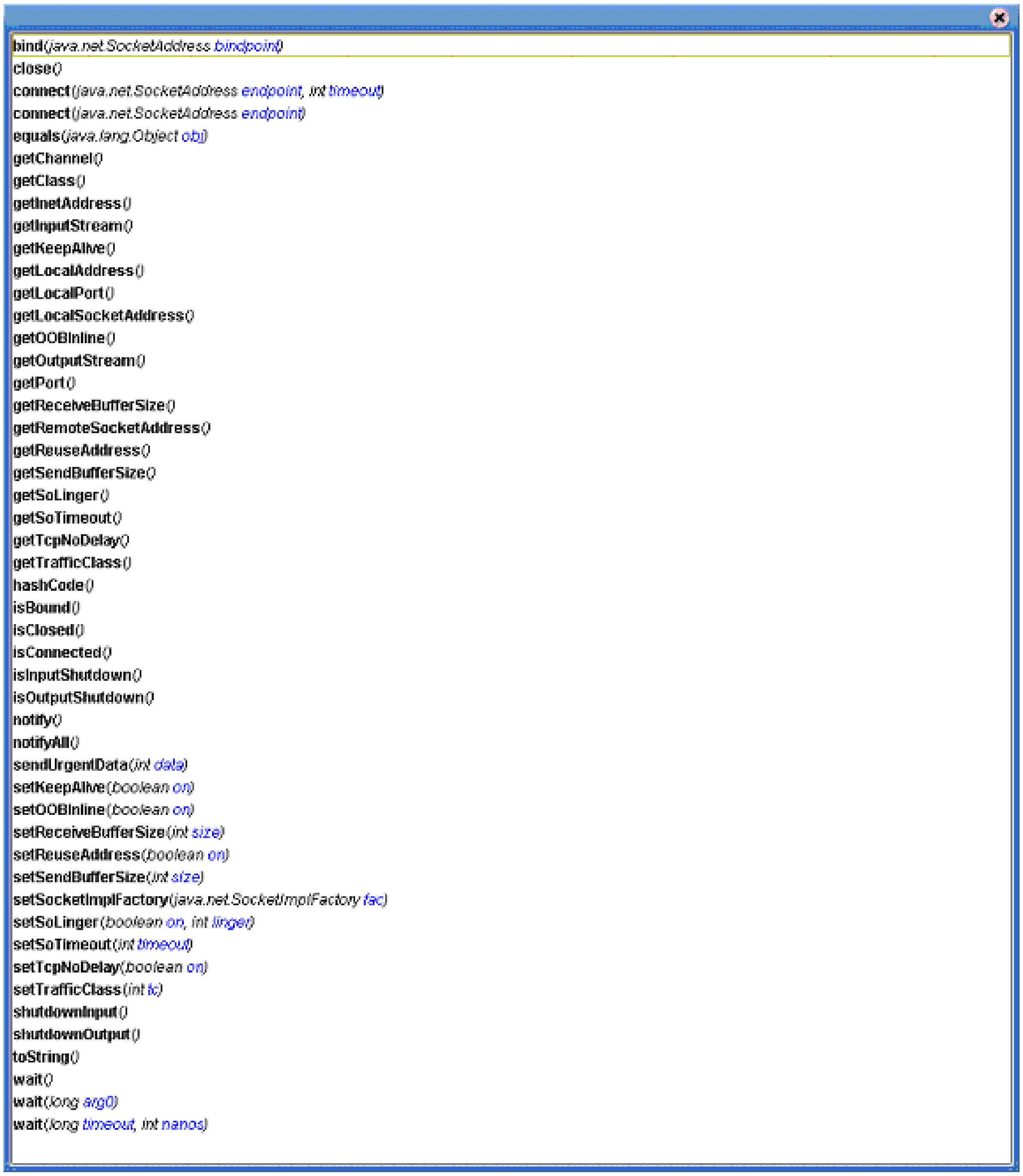
Appendix A

Methods Not Used in the eWay

You may see some method nodes exposed in the TCP/IP Object Type Definitions (OTDs), which cannot be used in Collaboration (Java).

Figure 10 on page 53 shows the methods available for **Channel** node that cannot be used in Collaboration (Java).

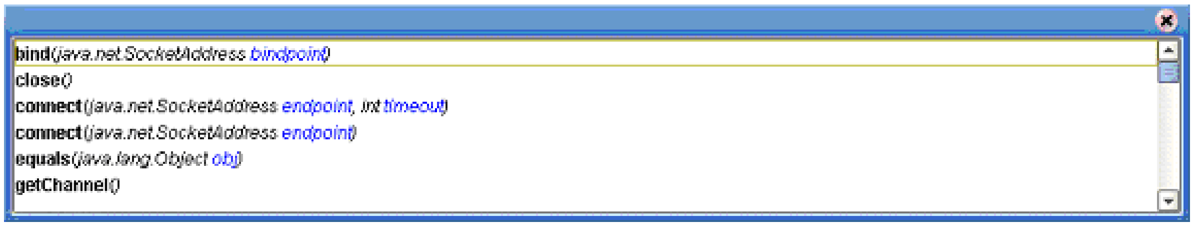
Figure 10 Unused Channel Methods



bind(java.net.SocketAddress bindpoint)
close()
connect(java.net.SocketAddress endpoint, int timeout)
connect(java.net.SocketAddress endpoint)
equals(java.lang.Object obj)
getChannel()
getClass()
getInetAddress()
getInputStream()
getKeepAlive()
getLocalAddress()
getLocalPort()
getLocalSocketAddress()
getOOBInline()
getOutputStream()
getPort()
getReceiveBufferSize()
getRemoteSocketAddress()
getReuseAddress()
getSendBufferSize()
getSoLinger()
getSoTimeout()
getTcpNoDelay()
getTrafficClass()
hashCode()
isBound()
isClosed()
isConnected()
isInputShutdown()
isOutputShutdown()
notify()
notifyAll()
sendUrgentData(int data)
setKeepAlive(boolean on)
setOOBInline(boolean on)
setReceiveBufferSize(int size)
setReuseAddress(boolean on)
setSendBufferSize(int size)
setSocketImplFactory(java.net.SocketImplFactory fac)
setSoLinger(boolean on, int linger)
setSoTimeout(int timeout)
setTcpNoDelay(boolean on)
setTrafficClass(int tc)
shutdownInput()
shutdownOutput()
toString()
wait()
wait(long arg0)
wait(long timeout, int nanos)

Figure 11 shows the methods exposed in the Socket node that cannot be used by Collaboration (Java).

Figure 11 Unused Socket Methods



```
bind(java.net.SocketAddress bindpoint)
close()
connect(java.net.SocketAddress endpoint, int timeout)
connect(java.net.SocketAddress endpoint)
equals(java.lang.Object obj)
getChannel()
```

Index

B

basic features **6**
Bytes to Read **15**

C

Collaboration Editor (Java) **31**
Connectivity Map Canvas **31**
Custom Enveloped Class Name **15**
Customer Defined Property **15**

D

Dedicated Session Mode **20**

E

Envelope Message (Client) **26**
Envelope Message (Server) **15**
Envelope Type **16**
  Begin-End Marked **16**
  Custom **18**
  End Marked **16**
  Fixed Length **16**
  Length Prefixed **17**
  Marked and Fixed **17**
  Per Active Connection **18**
eWay components, basic **32**
eWay operation, general **6**
eWay Properties sheet, using **12**
eWay properties, overview **7**
eWay with Java Collaborations, overview **39**
external system requirements **7**

G

General Client Settings **24**
general operation, eWay **6**
General Server Settings **19**

H

Host **28**
HP NonStop server requirements **8**, **44**

I

Ignore Until Char Value **19**
import sample Project **31**
installation **9**
  sample projects **10**
  sar files **10**

J

Java methods and classes
  overview **50**
Javadoc **10**, **50**
Javadoc, obtaining **50**

K

Keep Alive (client) **24**
Keep Alive (server) **21**

L

Logical Host requirements **7**

M

Max Data Size (client) **24**
Max Data Size (server) **20**
methods not used by eWay **52**

N

Numeric Representation **19**

O

operating systems
  supported **7**
OTD Editor **31**
OTDs, eWay
  components **47**

P

Parallel TCP/IP With Port Sharing **44**
Port **29**
Project
  canvas **30**
  deploying **43**
  External Environment **40**
Project sample
  components **37**
  importing **31**

operation **38**
overview **36**, **37**
Project sample, Collaboration (Java)
overview **39**
properties
Bytes to Read **15**
Custom Enveloped Class Name **15**
Customer Defined Property **15**
Dedicated Session Mode **20**
Envelope Type **16**
Host **28**
Ignore Until Char Value **19**
Keep Alive (client) **24**
Keep Alive (server) **21**
Max Data Size (client) **24**
Max Data Size (server) **20**
Numeric Representation **19**
Port **29**
Receive Buffer Size (client) **24**
Receive Buffer Size (server) **21**
Send Buffer Size (client) **25**
Send Buffer Size (server) **21**
Server Socket Factory Implementation Class **21**
ServerPort **27**
ServerSoTimeout **22**
Socket Factory Implementation Class **25**
SoLinger (client) **25**
SoLinger (server) **22**
SoLinger Timeout (client) **25**
SoLinger Timeout (server) **22**
SoTimeout (client) **26**
SoTimeout (server) **22**
Store Until Char Value **19**
TcpNoDelay (client) **26**
TcpNoDelay (server) **22**
Width of Length **19**
properties template **11**

R

Receive Buffer Size (client) **24**
Receive Buffer Size (server) **21**

S

sample scenarios
client mode **36**
server mode **37**
Send Buffer Size (client) **25**
Send Buffer Size (server) **21**
Server Socket Factory Implementation Class **21**
ServerPort **27**
ServerSoTimeout **22**
setting eWay properties
client mode properties **23**
Environment Explorer **12**
overview **11**
Project Explorer **11**
server mode properties **14**, **26**, **28**
server mode properties,Project Explorer **13**
values
Envelope Message (Client) **26**
Envelope Message (Server) **15**
General Client Settings **24**
General Server Settings **19**
TCP/IP Base Settings **24**
values, Environment Explorer **26**
Socket Factory Implementation Class **25**
SoLinger (client) **25**
SoLinger (server) **22**
SoLinger Timeout (client) **25**
SoLinger Timeout (server) **22**
SoTimeout (client) **26**
SoTimeout (server) **22**
Store Until Char Value **19**
supported operating systems **7**
System Requirements **7**
system requirements **7**
external **7**

T

TCP/IP Base Settings **24**
TCP/IP OTD
basic operation **47**
features **48**
TcpNoDelay (client) **26**
TcpNoDelay (server) **22**

W

Width of Length **19**