

***SeeBeyond™ eBusiness Integration Suite***

# **e\*Way Intelligent Adapter for the Microsoft Internet Information Server User's Guide**

*Release 4.5.3*

*Monk Version*



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

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

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 This work is protected as an unpublished work under the copyright laws.

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

Version 20020419163054.

# Contents

---

## Chapter 1

<b>Introduction</b>	<b>5</b>
<b>Overview</b>	<b>5</b>
Intended Reader	5
Components	6
<b>System Requirements</b>	<b>6</b>

---

## Chapter 2

<b>Installation</b>	<b>7</b>
<b>Installation Procedure</b>	<b>7</b>
<b>Supporting Documents</b>	<b>7</b>
Pre-installation	7
Installation Procedure	7
Web Server Configuration	8
<b>Files/Directories Created by the Installation</b>	<b>9</b>

---

## Chapter 3

<b>Configuration</b>	<b>10</b>
<b>e*Way Configuration Parameters</b>	<b>10</b>
<b>General Settings</b>	<b>10</b>
Request Reply IP Port	11
Push IP Port	11
Rollback if no Clients on Push Port	11
Wait For IQ Ack	11
Send Empty MSG When External Disconnect	12
MUX Instance ID	12
MUX Recovery ID	12
<b>External Configuration Requirements</b>	<b>13</b>

---

Chapter 4

<b>ActiveX APIs</b>	<b>14</b>
<b>Methods</b>	<b>14</b>
Connect	15
Disconnect	16
Send	17
Wait	18
LastErrorCode	19
LastErrorMessage	20
ReplyMessageAsArray	21
ReplyMessageAsString	22
ReplyMessageSize	23
<b>ActiveX Class ID</b>	<b>23</b>

---

Chapter 5

<b>Implementation</b>	<b>24</b>
<b>The Request/Reply Concept</b>	<b>24</b>
Request/Reply and the MS IIS e*Way	24
The Request/Reply Schema	26
ETDs and Form Data	27
Preparing Form Data for Request ETDs	28
<b>Using the ActiveX Control Within ASPs</b>	<b>29</b>
<b>Sample Implementation</b>	<b>30</b>
<b>Index</b>	<b>31</b>

# Introduction

This chapter describes how to install and configure the e\*Way Intelligent Adapter for the Microsoft Internet Information Server (MS IIS e\*Way), as well as how to create the Active Server Page required to enable the web server to communicate with e\*Gate.

---

## 1.1 Overview

The Internet Information Server (IIS) is Microsoft's Web server software. IIS utilizes Hypertext Transfer Protocol (HTTP) to deliver World Wide Web documents to clients using internet browsers such as Internet Explorer or Netscape Navigator. IIS includes support for Common Gateway Interface (CGI) scripts and Active Server Pages (ASPs). Active Server Pages can be used to customize delivery of information to the client regardless of the client platform. When an Active Server Page is requested by a browser, the server carries out any script commands embedded in the page, generates an HTML document, and sends the document back to the browser for display on the requesting (client) computer. Common uses for ASPs are web based queries where a user can look up information such as order status or shipping information.

The MS IIS e\*Way enables the e\*Gate system to exchange data with a MS IIS web server using Active Server Pages (ASPs). The MS IIS e\*Way extends the functionality of the IIS server by making external data sources available. Normally, the IIS server would be limited to sharing local resources only. By using the MS IIS e\*Way the IIS server can access remote data sources. The e\*Way also makes it possible to use a wider range of data sources that would not normally reside on the web server such as SAP, PeopleSoft, Lotus Notes, MS SQL, Oracle, Access, Excel and more.

This document describes how to install and configure the MS IIS e\*Way, and how to create the ASPs required to enable the web server to communicate with e\*Gate.

### 1.1.1 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the e\*Gate system; to have expert-level knowledge of Windows NT operations and administration; to be thoroughly familiar with ASP scripting; and to be thoroughly familiar with Windows-style GUI operations.

## 1.1.2 Components

The MS IIS e\*Way comprises the following:

- **stcewipmp.exe**, the executable component
- Configuration files, which the e\*Way Editor uses to define configuration parameters
- An ActiveX control (DLL) to provide client-side services

A complete list of installed files appears in [Table 1 on page 9](#) and [Table 2 on page 9](#).

---

## 1.2 System Requirements

This e\*Way is supported on the following operating systems:

- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Windows NT 4.0 SP6a

To use the MS IIS e\*Way, you need the following:

- An e\*Gate Participating Host, version 4.5.1 or later
- 4 MB free disk space on both the Participating Host and Registry Host systems

**Note:** *Additional disk space will be required to process and queue the data that this e\*Way processes; the amount necessary will vary based on the type and size of the data being processed, and any external applications performing the processing.*

- A TCP/IP network connection

For the web server that will communicate with the MS IIS e\*Way, you need a client system with the following:

- A Microsoft Internet Information Server with Active Server Pages enabled, and the libraries **stdole32.tlb** and **stdole2.tlb** installed. The Microsoft Personal Web Server (PWS) for Windows NT can also support Active Server Pages, but we recommend that the PWS be used only for testing.
- Sufficient memory and disk space to support web-server functions. See your MS IIS user's guides for more information about server requirements.

# Installation

---

## 2.1 Installation Procedure

This chapter describes the procedures necessary to install the MS IIS e\*Way from the e\*Gate installation CD-ROM.

After the product is installed, you must customize it to execute your site-specific business logic and to interact with your other systems as required. The steps necessary to perform those operations are discussed in the e\*Gate documentation set and online Help systems.

---

## 2.2 Supporting Documents

The following documents are designed to work in conjunction with the *e\*Gate API Kit User's Guide* and to provide additional information that may prove useful to you.

- *e\*Gate Integrator Installation Guide*
- *e\*Gate Integrator System Administration and Operations Guide*.
- *README.txt* file on the e\*Gate installation CD ROM.

### 2.2.1 Pre-installation

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

### 2.2.2 Installation Procedure

To install the MS IIS e\*Way on a Windows system:

- 1 Log in as an Administrator on the workstation on which you want to install the e\*Way.
- 2 Insert the e\*Way installation CD-ROM into the CD-ROM drive.

- 3 If the CD-ROM drive's "Autorun" feature is enabled, the setup application should launch 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 will launch. Follow the on-screen instructions to install the e\*Way.

**Note:** *Be sure to install the e\*Way files in the suggested "client" installation directory. The installation utility detects and suggests the appropriate installation directory. Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested "installation directory" setting.*

- 5 After the installation is complete, exit the install utility and launch the Enterprise Manager.
- 6 In the Component editor, create a new e\*Way.
- 7 Display the new e\*Way's properties.
- 8 On the General tab, under **Executable File**, click **Find**.
- 9 Select the file **stcewipmp.exe**.
- 10 Click OK to close the properties sheet, or continue to configure the e\*Way. Configuration parameters are discussed in [Chapter 3](#). The setup and requirements of schemas required to use this e\*Way are discussed in [Chapter 5](#).

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

*For more information about configuring e\*Ways or how to use the e\*Way Editor, see the [Working with e\\*Ways user guide](#).*

### 2.2.3 Web Server Configuration

The web server with which the MS IIS e\*Way will communicate must have support for Active Server Pages enabled, and the directory containing the ASP scripts must have execute access enabled.

For more information about ASP support and requirements, see your MS IIS user's guides.



## 2.3 Files/Directories Created by the Installation

The MS IIS e\*Way installation process will install the following files 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 Installed on the Participating Host

e*Gate Directory	File(s)
\bin\	stcewimpmp.exe stc_common.dll
\configs\stcewimpmp\	stcewimpmp.def

**Table 2** Files Installed on the External System

File	Purpose
stc_ximpmpclnt.dll stc_common.dll stc_ewimpmpclnt.dll	Supports ActiveX clients

The files **stc\_ximpmpclnt.dll**, **stc\_common.dll**, and **stc\_ewimpmpclnt.dll** install automatically on the system that the participating host is installed. They *must* be installed upon each system that the participating host does not reside. The MS IIS server system with which this e\*Way will communicate will require no additional action if the participating host is installed. The files may be copied from the egate\client folder into any directory on any additional systems, as long as that directory is within the PATH or the calling application’s current working directory..

After the files have been installed, they must be registered in the Windows NT registry on the client system (not the Participating Host) using the **regsvr32** command-line utility. To do this, launch a command shell and type:

```
regsvr32 your_path_location\stc_ximpmpclnt.dll
```

at the command prompt. A separate dialog box will display to confirm that the command was performed successfully; no messages will be displayed in the command window.

For more information about this utility, type **regsvr32** with no arguments at the Windows NT command prompt.

**Note:** *The suggested order for the files is stc\_common.dll, stc\_ewimpmpclnt.dll, stc\_ximpmpclnt.dll and then any additional files as needed.*

# Configuration

This chapter describes the e\*Way configuration parameters and the external configuration requirements for the MSIS e\*Way.

**Important:** *From the perspective of the e\*Gate GUIs, the MSIS e\*Way is not a system of components distributed between the web server and a Participating Host, but a single component that runs an executable file (the multiplexer `stcewimp.exe`). When this manual discusses procedures within the context of any e\*Gate GUI (such as this chapter, which deals in part with the e\*Way Editor), the term “e\*Way” refers only to the Participating Host component of the MSIS e\*Way system.*

---

## 3.1 e\*Way Configuration Parameters

e\*Way configuration parameters are set using the e\*Way Editor.

To change e\*Way configuration parameters:

- 1 In the Enterprise Manager’s Component editor, select the e\*Way you want to configure and display its properties.
- 2 Under **Configuration File**, click **New** to create a new file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file.
- 3 In the **Additional Command Line Arguments** box, type any additional command line arguments that the e\*Way may require, 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 a specific need to do so.

For more information about how to use the e\*Way Editor, see the e\*Way Editor’s online Help or the *e\*Gate Integrator User’s Guide*.

The e\*Way’s configuration parameters are organized into a single section: General Settings.

### 3.1.1 General Settings

The parameters in this section specify the name of the external client system and the IP port through which e\*Gate and the client system communicates.

## Request Reply IP Port

### Description

Specifies the IP port that the e\*Way will listen (bind) for client connections. This parameter is used for Request/Reply protocol.

### Required Values

A valid TCP/IP port number between 1 and 65536. The default is **26051**. Normally, you only need to change the default number if the specified TCP/IP port is in use, or you have other requirements for a specific port number.

## Push IP Port

### Description

Specifies the IP port through which this e\*Way allows an external system to connect and receive unsolicited (without submitting a request) Events.

### Required Values

A valid TCP/IP port number between 0 and 65536. The default is **0**.

### Additional Information

Any Event that this e\*Way receives that has zero values for all fields in the 24 byte MUX header is sent to all callers of the **WaitForUnsolicited**. This parameter is optional. If set to zero, the e\*Way will follow the Request/Reply scenario and not accept unsolicited Events.

Use the netstat program in a DOS window to identify in use ports.

## Rollback if no Clients on Push Port

### Description

Specifies whether the Event will continually roll back if there are no push clients connected.

### Required Values

**Yes** or **No**. If set to **Yes**, the Event will continually roll back if there are no push clients connected.

## Wait For IQ Ack

### Description

Specifies whether the send client function does NOT return until the Event is committed to the IQ.

### Required Values

**Yes** or **No**. If set to **Yes**, the send client function does NOT return until the Event is committed to the IQ.

**Caution:** *This parameter should be set if the data must be committed to the IQ on every transaction before the API returns to the client. Setting this parameter to Yes will significantly impact performance. If normal request/reply type transactions are being sent/received, and the data can be recreated at the client, this parameter should not be set.*

## Send Empty MSG When External Disconnect

### Description

Specifies whether the e\*Way sends an empty incoming message (containing only the multi-plexer header) when an external client disconnects.

### Required Values

**Yes** or **No**. If set to **Yes**, the e\*Way sends an empty incoming message when an external client disconnects.

## MUX Instance ID

### Description

Specifies whether the specified 8 (eight) bytes is prepended to the 24 (twenty-four) byte session ID of the request received from the external connection before sending to e\*Gate.

### Required Values

A string. If this value is other than "0", the 8 bytes are prepended to the 24 byte session ID. The default is 0.

**Note:** *This is a string where "00" and "00000000" are valid MUX Instance IDs, while "0" is to turn this option off. Only the first 8 bytes are used.*

## MUX Recovery ID

### Description

Specifies whether the 8 bytes are prepended to the reply and republish back to e\*Gate provided the value is other than "0" and the multi-plexer finds that the session related to the MUX ID in the return message has been dropped..

### Required Values

A string. If this value is other than "0", the 8 bytes are prepended to the 24 byte session ID. The default is 0.

**Note:** *This is a string where "00" and "00000000" are valid MUX Recovery IDs, while "0" is to turn this option off. Only the first 8 bytes are used.*

---

## 3.2 External Configuration Requirements

To enable the client system to communicate with the MS IIS e\*Way, you must do the following:

- 1 Install the required client files on the external system (see [“Files/Directories Created by the Installation” on page 9](#)).
- 2 Configure the client components as necessary to use the TCP/IP port specified above in [“Request Reply IP Port” on page 11](#) or [“Push IP Port” on page 11](#).
- 3 On the MS IIS web server, enable Active Server Pages, and enable "execute" access for those directories that contain the ASP files. For more information about ASP support and requirements, see your MS IIS user's guides.

# ActiveX APIs

---

## 4.1 Methods

The MS IIS e\*Way ActiveX control supports the following methods:

- **Connect** on page 15
- **Disconnect** on page 16
- **Send** on page 17
- **Wait** on page 18
- **LastErrorCode** on page 19
- **LastErrorText** on page 20
- **ReplyMessageAsArray** on page 21
- **ReplyMessageAsString** on page 22
- **ReplyMessageSize** on page 23

## Connect

### Syntax

*Connect bstrMUXHost, lMUXListenPort*

### Description

**Connect** opens a connection to the specified host using the specified port.

### Parameters

Name	Type	Description
bstrMUXHost	BSTR	The name of a network host.
lMUXListenPort	long	The TCP/IP port number over which to establish the connection.

### Return Values

None.

### Examples

```
const strObjId = "xipmpclnt.MUX"  
const strHost = "localhost"  
const dwPort = 26051  
  
set rr = server.CreateObject("xipmpclnt.MUX")  
  
on error resume next  
  
' Connect to the IP/MP e*Way  
rr.Connect strHost, dwPort
```

## Disconnect

### Syntax

```
Disconnect
```

### Description

**Disconnect** closes an open connection.

### Parameters

None.

### Return Values

None.

### Examples

```
' Close the connection to e*Gate and discard the COM object  
rr.Disconnect()  
set rr = nothing
```



## Send

### Syntax

*Send bstrRequestMessage, cSecondsAlive*

### Description

**Send** sends data into the e\*Gate system.

### Parameters

Name	Type	Description
bstruRequestMessage	BSTR	The message to send into the e*Gate system.
cSecondsAlive	long	The number of <b>seconds</b> this request will “live” within the e*Gate system before being dropped as an “expired” Event.

### Return Values

None.

### Examples

```
' create a string to contain data
dim strSend
strSend = "your data here"
' Send the string to the e*Way, with a 1000 millisecond expiration
rr.Send strSend, 1000
```

## Wait

### Syntax

```
Wait cBlockMilliseconds
```

### Description

**Wait** causes the application to wait the specified number of milliseconds for a message to be received.

### Parameters

Name	Type	Description
cBlockMilliseconds	BSTR	The number of milliseconds to wait to receive a message from the remote host.

### Return Values

None.

### Examples

```
' If an error occurred, show it; otherwise, wait 10,000 milliseconds  
if rr.LastErrorCode() > 0 then  
    ShowError(rr)  
else  
    rr.Wait 10000
```

## LastErrorCode

### Syntax

```
LastErrorCode
```

### Description

**LastErrorCode** returns the last error code or 0 for no error condition.

### Parameters

None.

### Return Values

Returns an error code.

### Examples

```
' If an error occurred, show it; otherwise, display a message  
if rr.LastErrorCode() > 0 then  
    ShowError(rr)  
else  
    Response.Write "Your message here"
```

## LastErrorText

### Syntax

```
LastErrorText
```

### Description

**LastErrorText** returns the text of the last error code.

### Parameters

None.

### Return Values

Returns an error message.

## ReplyMessageAsArray

### Syntax

```
ReplyMessageAsArray
```

### Description

**ReplyMessageAsArray** returns the outbound data as an array.

### Parameters

None.

### Return Values

Returns the outbound data as an array.

## ReplyMessageAsString

### Syntax

```
ReplyMessageAsString
```

### Description

**ReplyMessageAsString** returns the outbound data as a string.

### Parameters

None.

### Return Values

Returns a string.

### Examples

```
if rr.LastErrorCode() > 0 then
    ShowError(rr)
else
    Response.Write(rr.ReplyMessageAsString)
```

## ReplyMessageSize

### Syntax

```
ReplyMessageSize
```

### Description

**ReplyMessageSize** returns the length in bytes of the outbound data.

### Parameters

None.

### Return Values

Returns a long integer.

### Examples

```
if rr.LastErrorCode() > 0 then
    ShowError(rr)
else
    Response.Write(rr.ReplyMessageSize)
```

---

## 4.2 ActiveX Class ID

The ID for the MS IIS e\*Way's ActiveX control is

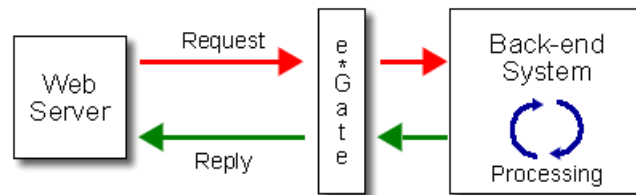
**ximpclnt.MUX**

# Implementation

## 5.1 The Request/Reply Concept

All the applications of the MS IIS e\*Way are based upon the “Request/Reply” concept. At a high-level, this works as follows:

- 1 The web server submits data (a **request**) to the e\*Gate system.
- 2 The e\*Gate system processes the data as required, communicating with other external (“backend”) systems as necessary.
- 3 The e\*Gate system returns data (a **reply**) to the same thread within the web server that submitted the request.



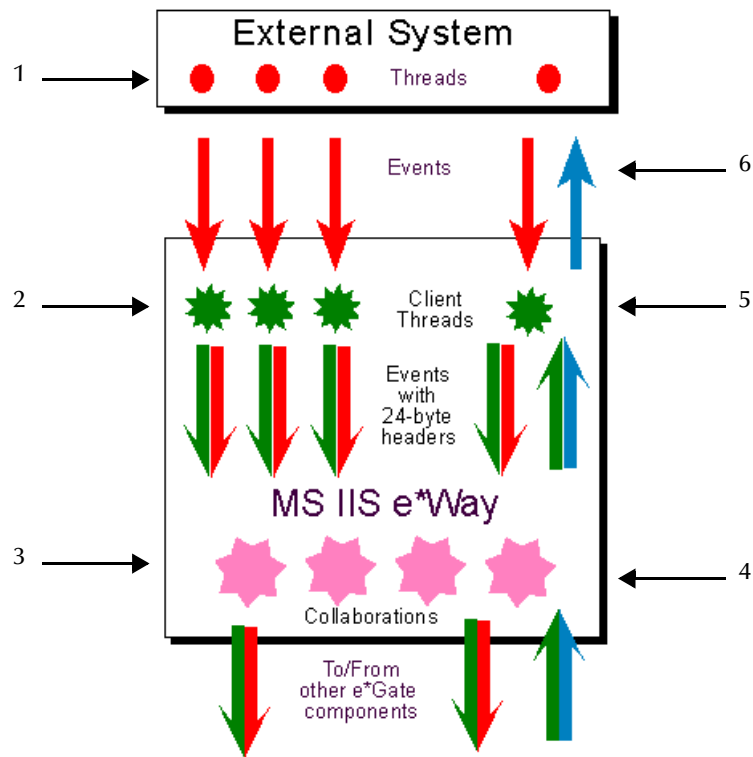
**Figure 1** The Request/Reply concept

### 5.1.1 Request/Reply and the MS IIS e\*Way

The MS IIS e\*Way uses a proprietary IP-based protocol to multi-thread Event exchange between the e\*Way and external systems or other e\*Gate components.

Figure 2 illustrates how the MS IIS e\*Way receives data from an external application and returns processed data to the same application.





**Figure 2** Data flow through the MS IIS e\*Way

- 1 The web server uses SeeBeyond's ActiveX controls to send the data to the MS IIS e\*Way using an SeeBeyond-proprietary IP-based protocol.
- 2 Client threads within the e\*Way package the data as e\*Gate Events, adding a 24-byte header. Among other functions, this header provides "return address" information that can optionally be used to return data to the client thread that originated it.

Each e\*Way can handle up to 1,000 client threads at once. If your requirements demand more processing power, you can define more MS IIS e\*Ways.

- 3 Collaborations within the e\*Way perform any appropriate processing that may be required, and route the processed Events to other destinations (such as an external system for additional data retrieval or processing).

**Note:** The 24-byte header *must* be preserved as the Events are processed through the e\*Gate system.

- 4 Processed data, still containing the original 24-byte header, is returned to the MS IIS e\*Way.
- 5 The e\*Way uses the 24-byte "return address" to identify the destination of the data to be returned to the external system.
- 6 The e\*Way returns the data, minus the 24-byte header, to the client thread within the web server.

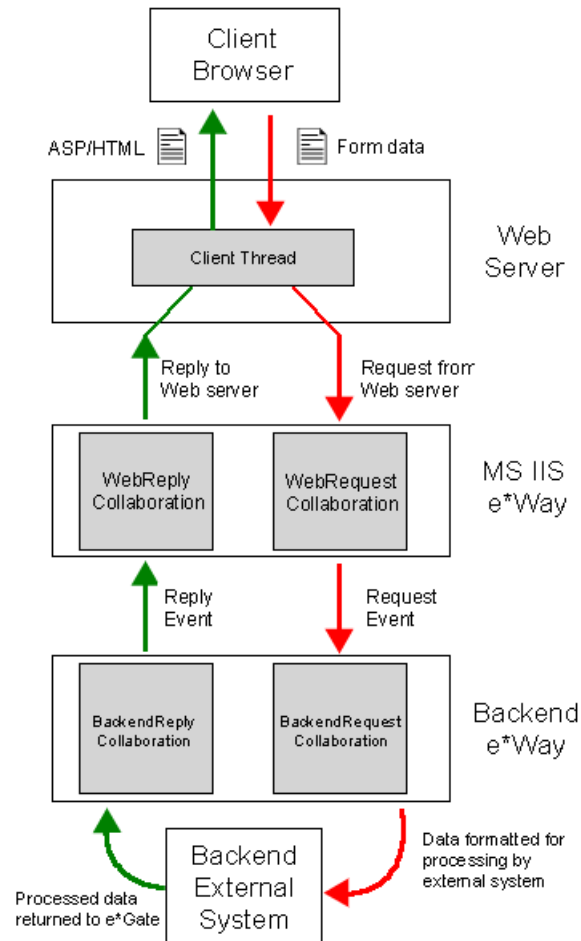
## 5.1.2 The Request/Reply Schema

Request/Reply schemas have two classes of components:

- 1 “Front end” components that handle communications with the external application. These components receive requests and route replies to the correct destination.
- 2 “Back end” components that process the requests and compose the replies. These components also provide the bridge between the e\*Gate system and your existing systems.

The MS IIS e\*Way and its related Collaborations comprise the front-end components. A second e\*Way and its related Collaborations comprise the back-end components (more e\*Ways may be added to communicate with more external systems as required). The backend e\*Way(s) can be of any type required to communicate with the external system(s).

Figure 3 below illustrates a Request/Reply schema.



**Figure 3** The Request/Reply schema

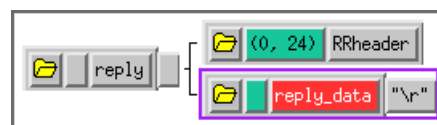
- 1 The user submits data to the web server via a browser form.

- 2 The ASP page on the web server packages the data and forwards it to e\*Gate.
- 3 Data enters the e\*Gate system through the MS IIS e\*Way's WebRequest Collaboration.
- 4 The WebRequest Collaboration publishes the Request Event.
- 5 The BackendRequest Collaboration within the back-end e\*Way subscribes to the Request Event, and processes the data as appropriate, and routes it to the external backend system.
- 6 The external backend system processes the data and returns it to e\*Gate via the Backend e\*Way.
- 7 The Backend e\*Way's BackendReply Collaboration publishes the data as the Reply Event.
- 8 The WebReply Collaboration within the MS IIS e\*Way subscribes to the Reply Event.
- 9 The MS IIS e\*Way returns the processed data to the requesting thread in the web server.
- 10 The web server completes the ASP page and returns the data as HTML to the client browser.

### 5.1.3 ETDs and Form Data

As discussed in [“Request/Reply and the MS IIS e\\*Way” on page 24](#), the MS IIS e\*Way maintains “return address” information in a 24-byte header that must be preserved as the data flows through the e\*Gate system.

The simplest Event Type Definition (ETD) that can be used within a Request/Reply schema has two nodes: one for the header, the second for the remainder of the Event data.



**Figure 4** The simplest Request/Reply ETD

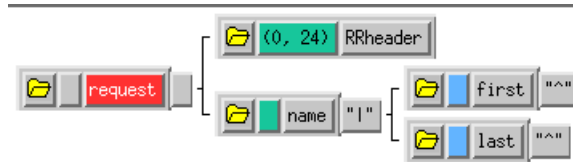
This ETD is sufficient if you wish to send data through the e\*Gate system simply as a blob. For example, you can compose the reply to the web server as a block of completely formatted HTML code, then return that reply using the above ETD.

Although the simple ETD in Figure 4 can be sufficient for reply data, request (input) data will probably have a more complex structure. To accommodate this, you must add the additional structure to the basic Request/Reply ETD:

- 1 Be sure to maintain the 24-byte “header” node unchanged.
- 2 Add one subnode beneath the “data” node for each element of input data.

- 3 Modify those subnodes as necessary (for example, to use appropriate delimiters or record lengths).

Figure 5 below illustrates an ETD that describes delimited data (for example, as in the data “First name^Last name”).

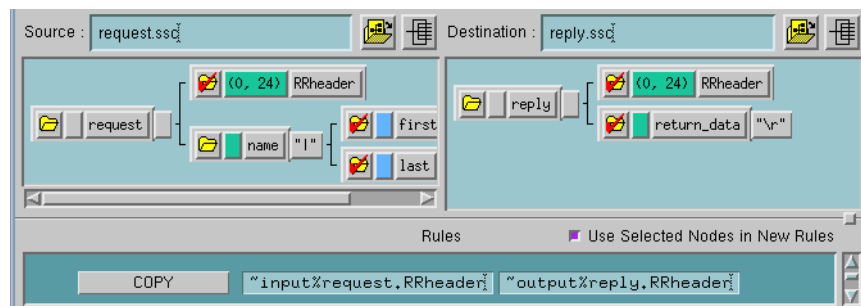


**Figure 5** A Request/Reply ETD for delimited data

Of course, the more complex the form that collects user data, the more complex the ETD that describes the form data must become. Be sure that the Request ETD meets the requirements for input to the backend system.

#### Collaboration Rules and the MS IIS Header

Collaboration Rules that manipulate data between ETDs must preserve the Request/Reply header (in the figures above, “RRheader”). Be sure that each Collaboration Rule that manipulates Request/Reply data copies the contents of the Request/Reply header from the source ETD to the target ETD (as shown in Figure 6 below).



**Figure 6** Copying the Request/Reply header

### 5.1.4 Preparing Form Data for Request ETDs

Form data to be used as input for e\*Gate must be formatted appropriately for the Request (input) ETD and sent to the MS IIS e\*Way as a single input string. Create the e\*Gate input string simply by concatenating the form data with appropriate delimiters. For example, to create the “Last name^First name” string for the ETD illustrated in [Figure 5 on page 28](#), use the following ASP code:

```

dim strFName
dim strLName
dim strSend

strFName = Request.form("txtFirstname")
strLName = Request.form("txtLastname")

```

```
`assemble the string to be sent
strSend = strLName & "^" & strFName & "|"
```

You can also use other coding schemes to delimit form data, such as the XML code in the example below:

```
strSend = "<FirstName>" & strFName & "</FirstName>"
strSend = strSend & "<LastName>" & strLName & "</LastName>"
```

Once the ASP code has assembled the input string, it can send the string to the e\*Way using the "Send" method. See the next sections for more information about using ASP code to send data into the e\*Gate system.

---

## 5.2 Using the ActiveX Control Within ASPs

The minimal implementation of the MS IIS e\*Way via a web server requires two pages:

- 1 A page that contains a form for accepting user data
- 2 An ASP file that both sends the user data to the e\*Way and posts the results to the user's browser.

The form requires only basic HTML functions to post the data to the server. The ACTION attribute of the <FORM> tag must take the name of the ASP file, using the method POST.

It is important that *none* of the form fields being processed by the ASP file be blank. You may validate the form input on either the client side (with JavaScript or VBScript) or on the server side within the ASP file, but you must validate the form fields to ensure none are blank before the ASP file processes the fields.

The ASP file must do the following:

- 1 Create an instance of the ActiveX control using **server.CreateObject**
- 2 Define the host name and TCP/IP port numbers
- 3 Use the Connect method (see **"Connect" on page 15**) to open a connection to the MS IIS e\*Way
- 4 Format the user data as appropriate for processing within e\*Gate
- 5 Use the Send method (see **"Send" on page 17**) to send data to the e\*Gate system
- 6 Use the Wait method (see **"Wait" on page 18**) to cause the web-server thread to pause long enough for e\*Gate to process and return the data
- 7 Use one of the "ReplyMessageAs" methods (such as **"ReplyMessageAsString" on page 22**) to display the returned data within the user's browser.
- 8 Handle errors using one of the "LastError" methods (such as **"LastErrorCode" on page 19**)
- 9 Close the connection using the Disconnect method (see **"Disconnect" on page 16**)

Additional information can be found in commented sample files (see the next section for more information).

---

## 5.3 Sample Implementation

A sample implementation is available in the **samples** directory of the e\*Gate CD-ROM. Navigate to the directory

```
samples/ewmux
```

and follow the directions in the README file in that directory.

In the demonstration schema, the back end is provided by a TCP/IP e\*Way that applies data-manipulation Collaboration Rules and a Loopback e\*Way that sends the TCP/IP e\*Way's output back into the e\*Gate system.

If you use the Enterprise Manager to examine the sample schema, you will note that the Loopback e\*Way has no Collaborations. This is correct; the Loopback e\*Way requires none to perform its "loopback" function.

***Note:** The TCP/IP e\*Way used in the demonstration schema was developed specifically for this use. A general-purpose TCP/IP e\*Way is also available for other uses; contact SeeBeyond for more information.*

# Index

## A

ActiveX control  
 Class ID 23  
 in ASPs 29  
 ASPs 29

## C

Class ID, ActiveX control 23  
 Configuration parameters  
 Push IP Port 11  
 Request Reply IP Port 11  
 Connect method 15

## D

delimited data, handling in ETDs 27  
 Disconnect method 16

## E

ETDs, sample 28  
 Event Type Definitions, sample 28

## F

files created by installation procedure 9

## H

header, in Collaboration Rules 28

## I

installation  
 files/directories created 9

## L

LastErrorCode method 19  
 LastErrorText method 20

## M

maximum client threads per e\*Way 25  
 methods  
 Connect 15  
 Disconnect 16  
 LastErrorCode 19  
 LastErrorText 20  
 ReplyMessageAsArray 21  
 ReplyMessageAsString 22  
 ReplyMessageSize 23  
 Send 17  
 Wait 18  
 MUX Instance ID 12  
 MUX Recovery ID 12

## P

Push IP Port 11

## R

ReplyMessageAsArray method 21  
 ReplyMessageAsString method 22  
 ReplyMessageSize method 23  
 Request Reply IP Port 11  
 request/reply  
 header, in Collaboration Rules 28  
 overview 24  
 schema 26  
 Rollback if no Clients on Push Port 11

## S

schema for request/reply configuration 26  
 Send Empty MSG When External Disconnect 12  
 Send method 17  
 Supporting Documents 7  
 system requirements 6

## W

Wait For IQ Ack 11  
 Wait method 18