SeeBeyond ICAN Suite

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

Release 5.0.5 for Schema Run-time Environment (SRE)

Monk Version

Version 20050406012149.

Contents

Chapter 1

Introduction

This chapter provides an overview of SeeBeyond™ Technology Corporation's (SeeBeyond™) e*Way™ Intelligent Adapter for the Microsoft Internet Information Server (MS IIS e*Way) and an introduction to this guide. It also explains how to create the Active Server Page required to enable the Web server to communicate with the e*Gate Integrator system.

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 high-level knowledge of Windows 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 is made up of the following components:

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

A complete list of installed files appears in **Table 1 on page 8** and **Table 2 on page 9**.

1.2 Supported Operating Systems

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

- Windows 2000, Windows XP, and Windows Server 2003
- Japanese Windows 2000, Windows XP, and Windows Server 2003

1.3 System Requirements

To use the MS IIS e*Way, you need to meet the following requirements:

- An e*Gate Participating Host
- 4 MB free disk space on both the Participating Host and Registry Host systems

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

- A TCP/IP network connection

1.4 External System Requirements

For the Web server that communicates with the MS IIS e*Way, you need a client system with the following requirements:

- 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 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.

Chapter 2

Installation

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.

2.1 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 Installation Procedure

To install the MS IIS e*Way on Windows systems

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 launches. 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 Schema Designer.

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 **e*Gate Integrator User's Guide**.*

2.3 Web Server Configuration

The Web server with which the MS IIS e*Way communicates 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.

2.4 Files/Directories Created by the Installation

The MS IIS e*Way installation process installs the files shown in Table 1 within the e*Gate directory tree. Files are installed within the **eGate\client** tree on the Participating Host and committed to the "default" schema on the Registry Host.

Table 1 Files Installed on Participating Host

Directories	Files
bin\	stcewipmp.exe stc_common.dll
configs\stcewipmp\	stcewipmp.def

Table 2 shows the files installed on the external system.

Table 2 Files Installed on External System

File	Purpose
stc_xipmpclnt.dll stc_common.dll stc_ewipmpclnt.dll	Supports ActiveX clients

The files **stc_xipmpclnt.dll**, **stc_common.dll**, and **stc_ewipmpclnt.dll** install automatically on the system where the Participating Host is installed. These files *must* be installed on each system where the Participating Host does not reside. The MS IIS server system with which this e*Way communicates require no additional action if the Participating Host is installed.

You can copy the files from the **eGate\client** folder into any directory on any additional system, as long as that directory is within the PATH statement or the calling application's current working directory.

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

regsvr32 your_path_location\stc_xipmpclnt.dll

Enter this line at the command prompt. A separate dialog box displays to confirm that the command was performed successfully. No messages are displayed in the command window.

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

Note: *The suggested order for the files is* ***stc_common.dll***, ***stc_ewipmpclnt.dll***, ***stc_xipmpclnt.dll****, then any additional files as needed.*

Chapter 3

Configuration

This chapter describes the e*Way configuration parameters and the external configuration requirements for the MS IIS e*Way.

3.1 e*Way Configuration Parameters

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

Important: *From the perspective of the e*Gate Schema Designer, the MS IIS 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 **stcewipmp.exe**). When this guide discusses procedures within the context of any Schema Designer feature (such this chapter, which deals in part with the e*Way Editor), the term "e*Way" refers only to the Participating Host component of the MS IIS e*Way system.*

To change e*Way configuration parameters

1. In the Schema Designer'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 where the e*Way listens (binds) 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 follows 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 continually rolls back if there are no push clients connected.

Required Values

Yes or **No**. If set to **Yes**, the Event continually rolls 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 significantly impacts 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 operations:

1. Install the required client files on the external system (see **"Files/Directories Created by the Installation" on page 8**).
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.

Chapter 4

ActiveX API Methods

This chapter explains the ActiveX Application Programming Interface (API) methods.

4.1 Methods

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

The rest of this section explains these methods in detail.

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

```
Disonnect
```

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
bstrRequestMessage	BSTR	The message to send into the e*Gate system.
cSecondsAlive	long	The number of **seconds** this request can "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

Chapter 5

Implementation

This chapter explains how to implement the e*Way Intelligent Adapter for the Microsoft Internet Information Server in a production environment.

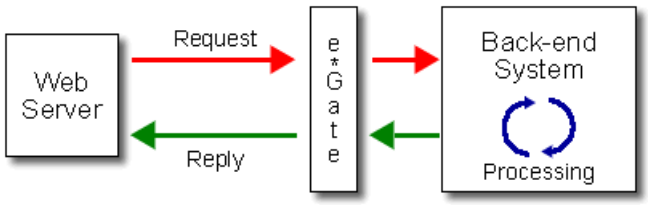
5.1 The Request/reply Model

All the applications of the MS IIS e*Way are based upon the request/reply model. At a high level, this model 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.

For an illustration of how this model operates, see Figure 1.

Figure 1 The Request/reply Model

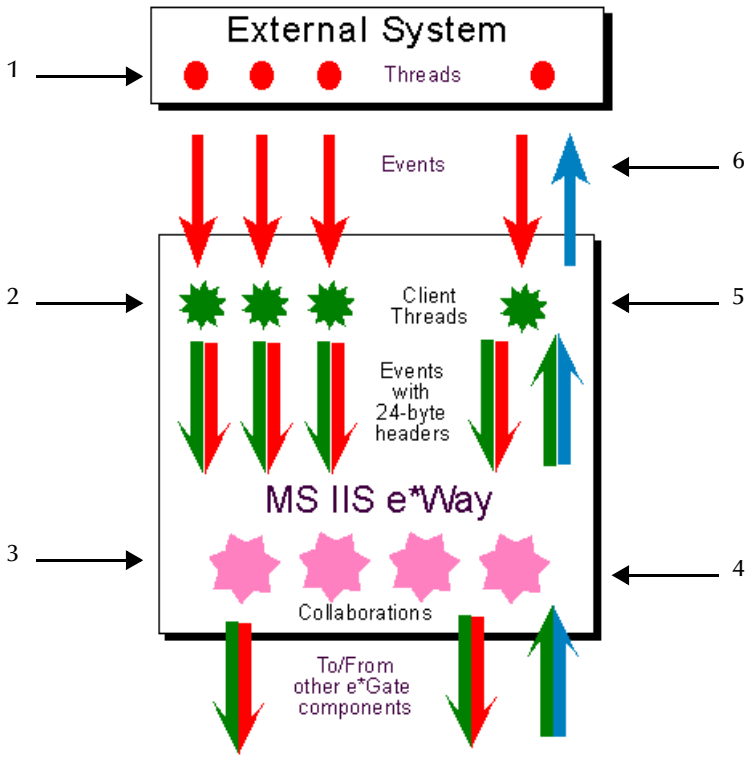


5.1.1 Request/reply and the MS IIS e*Way

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

Figure 2 on page 20 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 while 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

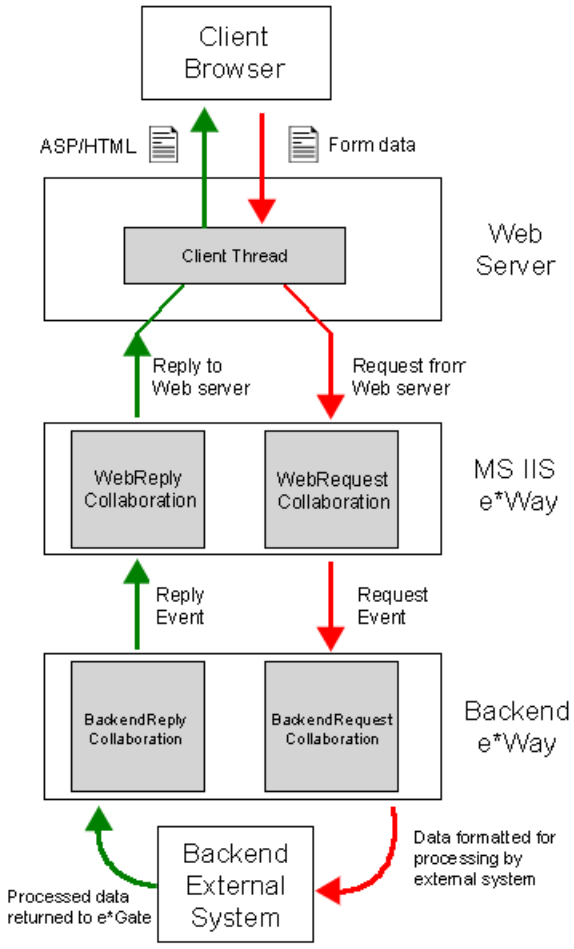
Request/reply schemas have two classes of components:

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 illustrates a request/reply-based schema.

Figure 3 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.

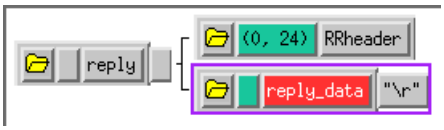
10 The Web server completes the ASP page and returns the data as HTML to the client browser.

5.1.3 ETDs and Form Data

As discussed in **"Request/reply and the MS IIS e*Way" on page 19**, 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 (see Figure 4).

Figure 4 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 is likely to have a more complex structure. To accommodate this fact, you must add the additional structure to the basic Request/reply ETD as follows:

- Be sure to maintain the 24-byte header node unchanged.
- Add one sub-node beneath the data node for each element of input data.
- Modify those sub-nodes as necessary (for example, to use appropriate delimiters or record lengths).

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

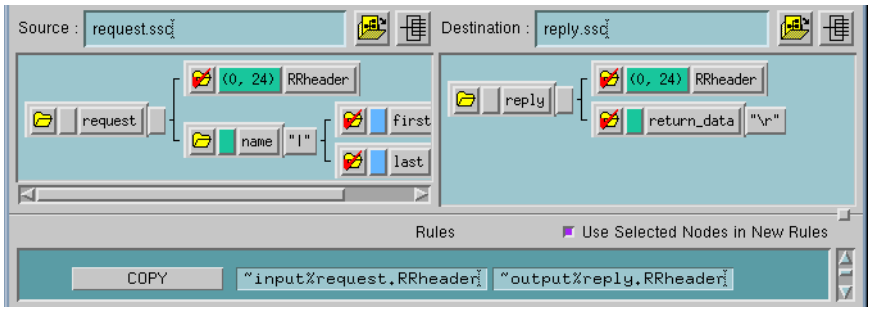
Figure 5 Request/reply ETD for Delimited Data

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 back-end system.

5.1.4 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.

Figure 6 Copying the Request/reply Header



5.1.5 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 23**, 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 following example:

```
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 14**) 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 15**) to send data to the e*Gate system
6. Use the Wait method (see **"Wait" on page 16**) 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 17**) to display the returned data within the user's browser.
8. Handle errors using one of the "LastError" methods (such as **"LastErrorCode" on page 16**)
9. Close the connection using the Disconnect method (see **"Disconnect" on page 15**)

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 following directory:

samples/ewmux

Follow the directions in the **Readme.txt** 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 Schema Designer to examine the sample schema, you can see 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