SeeBeyond™ eBusiness Integration Suite

e*Way Intelligent Adapter for Apache Web Server User's Guide

Release 4.5.2

Version 20021106175644.

Contents

Chapter 1

Introduction

This document describes how to install and configure the e*Way Intelligent Adapter for Apache Web Server, as well as how to create the Perl CGI scripts required to enable the web server to communicate with e*Gate.

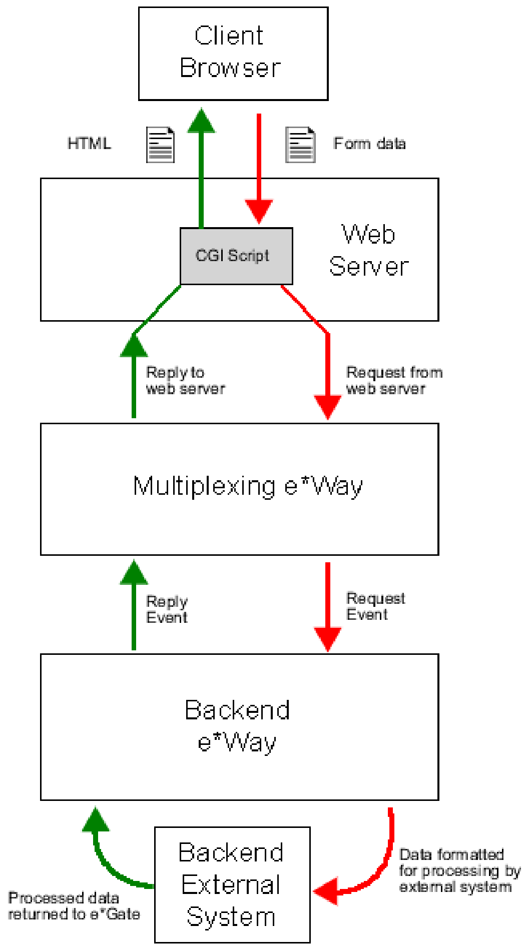
1.1 Overview

The Apache web server is a UNIX-based web server that uses Hypertext Transfer Protocol (HTTP) to deliver World Wide Web documents to clients using internet browsers such as Internet Explorer or Netscape Navigator. The Apache Web Server e*Way extends the functionality of the web server by making external data sources available to calls from within a Perl CGI script. Normally, the web server would be limited to sharing local resources only. By using the Apache Web Server e*Way, the web 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, Oracle, Lotus Notes, Excel, Access, and more. **Figure 1 on page 6** shows an overview of the Apache Web Server e*Way's implementation. For additional details, see **Chapter 5**.

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 operations and administration for the operating system(s) under which the Apache web server and e*Gate systems run; to be thoroughly familiar with CGI scripting using Perl; and to be thoroughly familiar with Windows-style GUI operations.

Figure 1 Overview of the Apache Web Server e*Way implementation



1.1.2 Components

The Apache Web Server e*Way comprises the following:

- Web server components: Perl CGI scripts, extension packages, and supporting library files
- Participating Host components: a multiplexing e*Way and supporting library files

A complete list of installed files appears in **Table 2 on page 12**.

1.2 Supported Operating Systems

The Apache Web Server e*Way is available on the following operating systems:

- Solaris 2.6, 7, and 8
- AIX 4.3.3

- HP-UX 11.0 and HP-UX 11i
- Red Hat Linux 6.2

1.3 System Requirements

To use the Apache Web Server e*Way, you need the following:

- An e*Gate Participating Host, version 4.5.1 or later.
- A TCP/IP network connection.

For the web server that will communicate with the Apache Web Server e*Way, you need a client system with the following:

- Apache Web Server version 1.3 or later.
- Perl: The following Perl libraries are supported:

OS	Perl
AIX 4.3.3	5.005_03 standard
HPUX 11.0, 11.i	5.005_03 HP-UX Developer Resource
Solaris 2.6, 7	download 5.005_03 from Sun Freeware
Solaris 2.8	5.005_03 standard
Linux Red Hat 6.2	5.005_03 standard
Compaq Tru64 5.0a	5.004_04
Windows NT/2000	5.6.1.629

  HPUX clients also require that the Perl executable must be linked against the p-thread library (using the flag **-lpthread**) when it is built.

  The above versions are the only versions that are officially supported and tested.

- Sufficient memory and disk space to support web-server functions. See your Apache Web Server user's guides for more information about server requirements.

Note: *The e*Gate Participating Host may optionally host the web server, but is not required to do so.*

Chapter 2

Installation

This chapter covers the requirements for installing the Apache Web Server e*Way Server and Client and how to configure the web server components needed. A list of the files and directories created by the installation is also provided.

2.1 Supporting Documents

The following documents are designed to work in conjunction with the *e*Way Intelligent Adapter for the Apache Web Server 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 Installing the Apache Web Server e*Way

This section describes the procedure for installing the Apache Web Server e*Way.

2.2.1 Pre-installation

- Exit all programs before running the setup program, including any anti-virus applications.
- You do not require root privileges to install this e*Way. Log in under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privilege to create files in the e*Gate directory tree.

2.2.2 Installation Procedure

If you are installing this e*Way as part of a complete e*Gate installation, please follow the instructions in the *e*Gate Integrator Installation Guide*. The Apache Web Server e*Way is installed as an "Add-on" component in the fourth phase of the installation.

If you are adding the Apache Web Server e*Way to an existing e*Gate installation, follow the instructions below.

To install the Apache Web Server e*Way Client and/or Apache Web Server e*Way Server Components on a UNIX system:

1. Log on to the workstation on which you want to install the e*Way. If you do not wish to log in as root, you must log in as a user with sufficient privilege to install files in the "egate" directory tree.
2. Insert the CD-ROM into the drive.
3. If necessary, mount the CD-ROM drive.
4. At the shell prompt, type

   cd /cdrom/setup
5. Start the installation script by typing:

   ./setup.sh
6. If you are not running as root, you will see a message notifying you that services will not start automatically for non-root users. Press **Enter** to continue.
7. You will see a message confirming that you are running the e*Gate installation script, and reminding you that you can type - (hyphen) to back up a step or **QUIT** (all capitals) to exit the install program. Press **Enter** to continue.
8. You will be prompted to accept the licence agreement. Type **y** and press **Enter**.

   The platform type and a menu of options will display:

```
Installation type (choose one):
  0. Finished with installation.  Quit.
  1. e*Gate Addon Applications
```

9. Type 1 to select the **e*Gate Add-on Applications** and press **Enter**.
10. You will be prompted for the installation path. Press **Enter** to accept the default path, or enter a new path and press **Enter**.
    - If you are logged in as root, the suggested path will be **/opt/egate/client**
    - If you are logged in under any other user name, the suggested path will be **/home/*username*/egate/client**.

    Whether you install e*Gate to a **/home** directory to an application directory such as **/opt**, we strongly recommend that you use the recommended relative path "egate/client" as the destination directory for the add-on-application installation.
11. When prompted, type **U** to update (overwrite) and press **Enter**.

Note: ***U** updates the installation, overriding files as necessary. **M** creates a directory and moves everything in the current directory to **directoryname**.old.*

If you selected "U," you will see a warning regarding shared EXE and DLL files. Read this warning and press **Enter** to continue.

12 Enter the name of the Registry Server supporting these add-on applications. If the installation utility detects a Registry Host running on the current server, it will suggest that host's name.

13 You will be prompted for the "administration login" (an e*Gate user with sufficient privilege to create components within a schema). The default is **Administrator**; unless you have created a different "administrative" user name, press **Enter** to accept the default. The default password is listed in the README.TXT file in the root directory of the installation CD-ROM.

14 Enter and confirm the password for the user specified in the step above.

Note: *e*Gate user names and passwords are case-sensitive.*

15 A menu of add-on options will appear. Type the number corresponding to the add-on package(s) you wish to install **Apache Web Server e*Way Web Server**) and press **Enter**.

16 Follow the on-screen instructions to complete the installation.

17 After the add-on application has been installed, the "Choose add-on packages" menu will appear. Repeat step 15 to install additional packages, or **0** and press **Enter** to continue.

18 When the "installation type" menu appears, the Add-on Applications installation is complete. Do one of the following:

- To exit the setup utility, type **0** and press **Enter.**
- Select another option to continue the installation.

2.2.3 Configuring the Participating Host Components in the Enterprise Manager

Important

From the perspective of the e*Gate GUIs, the Apache Web Server 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 manual discusses procedures within the context of any e*Gate GUI, the term "e*Way" refers only to the Participating Host component of the Apache Web Server e*Way system.

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**.*

For re-installing on AIX, please use the "slibclean" command before the installation procedure to thoroughly unload any old versions of shared libraries that might be cached by the dynamic loader.

2.3 Configuring the Web Server Components

To configure the web server to use the Apache Web Server e*Way client libraries:

Copy the following files onto your Apache web server. These files are available in the e*Gate client directory located on your e*Gate Participating Host.

- Modify the Apache Web server configuration file so that the dynamically loaded library path (LD_LIBRARY_PATH, SHLIB_PATH, LIBPATH, or PATH, depending on the operating system) contains the path of the Apache Web Server e*Way client library files (**stc_common.dll**, **stc_ewipmpclnt.dll**, and **stc_ewipmpclnt.dll.***)

For Apache Web Server on a Solaris operating system, make this modification:

```
setenv LD_LIBRARY_PATH <egate_client>/bin
```

Where <*egate_client*> is installed.

For Apache Web Server on HP-UX operating system, make this modification:

```
setenv LD_PRELOAD stc_ewipmpclntperl.sl:/usr/lib/libpthread.1
```

2.3.1. Configuring Perl

If the above files are copied to a host machine other then the e*Gate Participating Host you need to create a link for Perl and set an environment variable as indicated in Table 1.

Table 1 Link Setup

Operating System	Link Command	Library PATH
hpux11	ln -f -s stc_ewimpmpclntperl.dll stc_ewipmpclntperl.sl	SHLIB_PATH
sparc26	ln -f -s stc_ewimpmpclntperl.dll stc_ewimpmpclntperl.so	LD_LIBRARY_PATH
aix43	ln -f -s stc_ewimpmpclntperl.dll stc_ewimpmpclntperl.so	LIBPATH
linux6x86	ln -f -s stc_ewipmpclntperl.dll stc_ewimpmpclntperl.so	LD_LIBRARY_PATH
ctru64_4	ln -f -s stc_ewipmpclntperl.dll stc_ewipmpclntperl.dll	LD_LIBRARY_PATH

2.4 Files/Directories Created by the Installation

The Apache Web Server 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, files will be committed to the "default" schema on the Registry Host.

Table 2 Apache Web Server e*Way Server-side files installed

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

Table 3 lists the files that must be installed on the system running the Apache web server:

Table 3 Apache Web Server e*Way Client-side files installed

Client Directory	File(s)
\bin	stc_common.dll stc_ewipmpclnt.dll stc_ewipmpclntperl.dll
\perl	stc_ewipmpclntperl.pm

The files **stc_common.dll**, and **stc_ewipmpclnt.dll** install automatically on the system that the participating host is installed. These files **must** be installed on the system supporting the application with which this e*Way communicates, along with any additional files needed. 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.

Chapter 3

Perl Extension-package Subroutines

3.1 Subroutines

The Apache Web Server e*Way Web Server components support the following Perl extension subroutines:

- **Multiplexer_Close** on page 14
- **Multiplexer_Free** on page 15
- **Multiplexer_Init** on page 16
- **Multiplexer_Open** on page 17
- **Multiplexer_Send** on page 18
- **Multiplexer_ToString** on page 19
- **Multiplexer_Wait** on page 20

Multiplexer_Close

Syntax

```
Multiplexer_Close(handle)
```

Description

Multiplexer_Close closes the connection on the specified handle.

Parameters

Name	Type	Description
handle	handle	The handle returned by Multiplexer_Open

Return Values

Returns **true** if the command executed successfully; otherwise, returns **false**.

Examples

```
$result = Multiplexer_Close($handle);
if(!$result)
{
    print "Multiplexer_Close failed.\n";
}
```

Multiplexer_Free

Syntax

```
Multiplexer_Free(handle, message-pointer)
```

Description

Multiplexer_Free frees the memory associated with the message pointer.

Parameters

Name	Type	Description
handle	handle	The handle returned by Multiplexer_Open
message-pointer	pointer	The message pointer

Return Values

Returns **true** if the command executed successfully; otherwise, returns **false**.

Examples

```
$message_ptr = Multiplexer_Wait($handle,
                $message_length,3000, 0, 0);

$result = Multiplexer_Free($handle, $message_ptr);

if(!$result)
{
    print "Multiplexer_Free failed.\n";
}
```

Multiplexer_Init

Syntax

```
Multiplexer_Init(dll-path)
```

Description

Multiplexer_Init specifies the path that contains the e*Way's library files. This function is required in every Perl script that communicates with the Participating Host.

Parameters

Name	Type	Description
dll-path	string	The path that contains the e*Way's **.dll** files. The path can be a relative or absolute path to the script that calls the function.

Return Values

Returns **true** if the command executed successfully; otherwise, returns **false**.

Examples

```
# this is where stc_ewipmpclntperl.dll and stc_ewipmpclntperl.pm are
located.
use lib ("egate/client/bin");
use lib (“egate/client/perl”);

use CGI qw/:standard/;
use stc_ewipmpclntperl.pm;

# call Multiplexer_Init to define the library path
Multiplexer_Init("egate/client/bin");
```

Multiplexer_Open

Syntax

```
Multiplexer_Open(server-name,
     server-port, flags, reserved);
```

Description

Multiplexer_Open opens a connection to the Participating Host that is running the Apache Web Server e*Way.

Parameters

Name	Type	Description
server-name	string	The name of the e*Gate Participating Host
server-port	integer	The port through which to communicate with the multi-plexing e*Way. If this value is zero, the default port **26051** will be used.
flags	integer	Command flags should always be set to “0” (zero) unless directed otherwise by SeeBeyond support personnel.
reserved	integer	Reserved for future SeeBeyond use. always set to “0” (zero) unless directed otherwise by SeeBeyond support personnel.

Return Values

Returns a connection handle.

Examples

```
$handle = Multiplexer_Open(“server.mycompany.com”,26051, 0, 0);
  if(!$handle)
  { print "Multiplexer_Open failed.\n"; }
```

Multiplexer_Send

Syntax

```
Multiplexer_Send(handle, message-length, message,
                         seconds-to-expire, flags, reserved)
```

Description

Multiplexer_Send sends the specified message to the e*Gate Participating Host.

Parameters

Name	Type	Description
handle	handle	The handle returned by Multiplexer_Open
message-length	integer	The length of the message, in bytes
message	string	The data to send to the e*Gate Participating Host
seconds-to-expire	integer	The number of seconds after which the data within the e*Gate system will expire
flags	integer	Command flags. always set to “0” (zero) unless directed otherwise by SeeBeyond support personnel.
reserved	integer	Reserved for future SeeBeyond use. always set to “0” (zero) unless directed otherwise by SeeBeyond support personnel.

Return Values

Returns **true** if the command executed successfully; otherwise, returns **false**.

Examples

```
$result = Multiplexer_Send($handle, $message_length,
                $message, 0, 0, 0);
if(!$result)
{
    print "Multiplexer_Send failed.\n";
}
```

Multiplexer_ToString

Syntax

```
Multiplexer_ToString(message-pointer)
```

Description

Multiplexer_ToString returns the data associated with the specified message pointer as a string.

Parameters

Name	Type	Description
message-pointer	pointer	The pointer returned by Multiplexer_Wait

Return Values

Returns **true** if the command executed successfully; otherwise, returns **false**.

Additional Notes

In the current implementation, the null character ("\0") will terminate a response message, and any information that follows a null character will be discarded when you use **Multiplexer_ToString** to convert the response message to a string.

Examples

```
$message_received = Multiplexer_ToString($message_ptr);
$result = Multiplexer_Free($handle, $message_ptr);
if(!$result)
{
    print "Multiplexer_Free failed.\n";
}
```

See **Multiplexer_Free** on page 15 for more information.

Multiplexer_Wait

Syntax

```
Multiplexer_Wait(handle, message-length,
                           millsecond-timeout, flags, reserved)
```

Description

Multiplexer_Wait causes the application to wait up to the specified number of milliseconds for a message to be received.

Parameters

Name	Type	Description
handle	handle	The handle returned by Multiplexer_Open
message-length	integer	The length of the message received, in bytes (assigned as an output parameter).
millisecond-timeout	integer	The number of milliseconds to wait
flags	integer	Command flags. always set to “0” (zero) unless directed otherwise by SeeBeyond support personnel.
reserved	integer	Reserved for future SeeBeyond use. always set to “0” (zero) unless directed otherwise by SeeBeyond support personnel.

Return Values

Returns a message pointer.

Examples

```
$message_ptr = Multiplexer_Wait($handle,
                           $message_length,3000, 0, 0);
if(!$message_ptr)
{
    print "Multiplexer_Wait failed.\n";
}
```

Chapter 4

Configuration

Important: *From the perspective of the e*Gate GUIs, the Apache Web Server 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 manual discusses procedures within the context of any e*Gate GUI (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 Apache Web Server e*Way system.*

4.1 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.

4.1.1 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 behavior.

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.

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.*

4.2 External Configuration Requirements

To enable the client system to communicate with the Apache 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 12**)
2. Configure the client components as necessary to use the port specified above in **"Push IP Port" on page 22**.

Chapter 5

Implementation

5.1 The Request/Reply Concept

All the applications of the Apache Web Server 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 **response)** to the same thread within the web server that submitted the request.

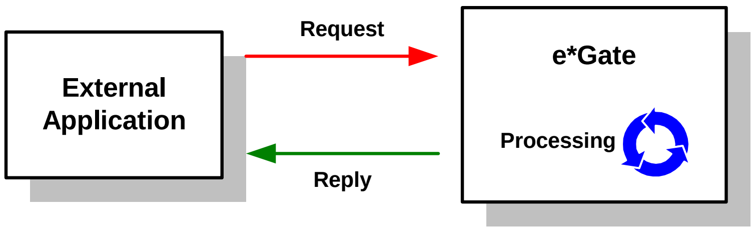


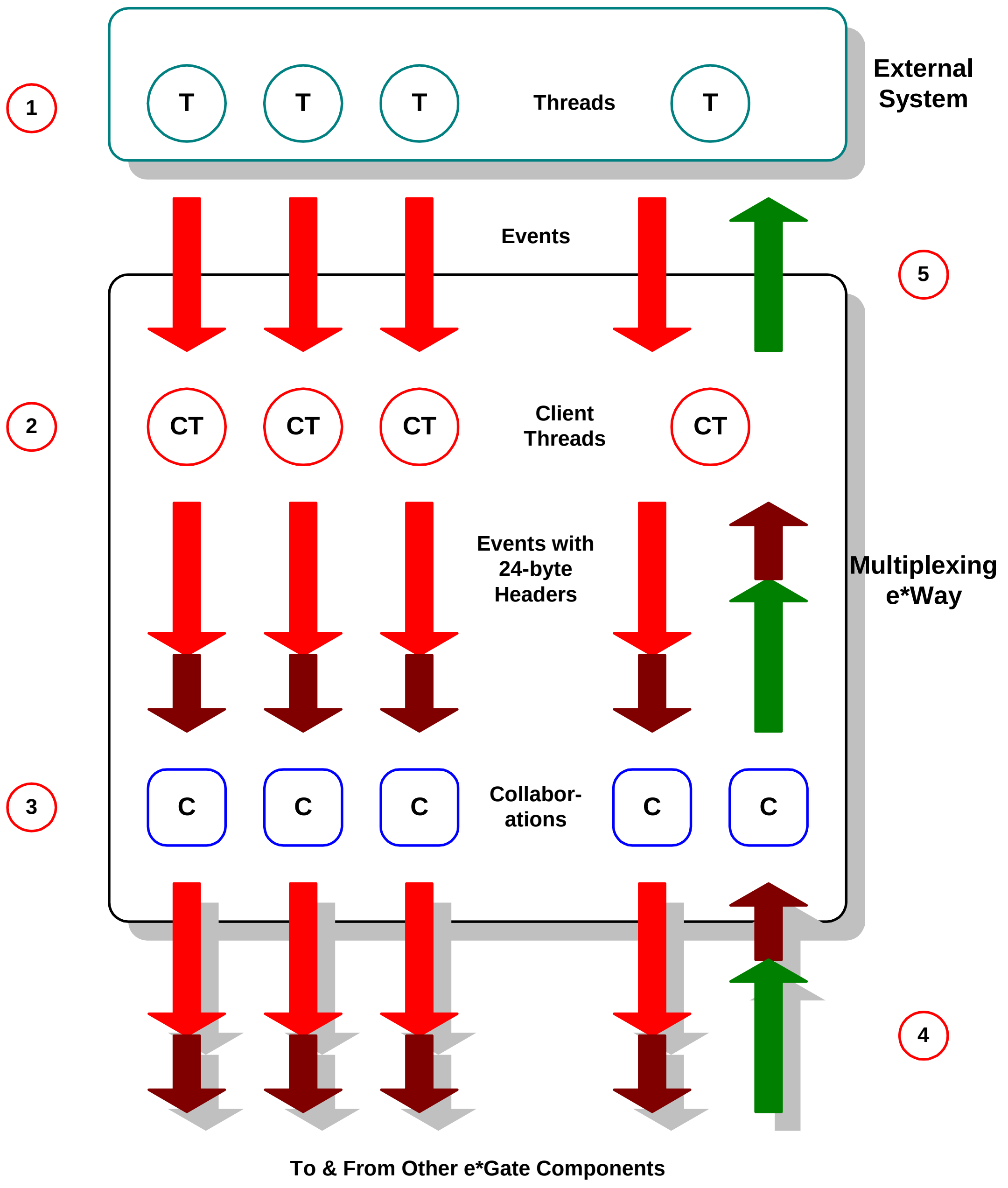
Figure 1 The Request/Reply concept

5.1.1 Request/Reply and the Apache Web Server e*Way Participating Host Components

The Apache Web Server e*Way Participating Host component is a multiplexing e*Way that 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 multiplexing e*Way receives data from an external application and returns processed data to the same application.

Figure 2 Data flow through the multiplexing e*Way



1 The web server uses a Perl script to send the data to the multiplexing 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 Apache Web Server 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 Apache Web Server 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

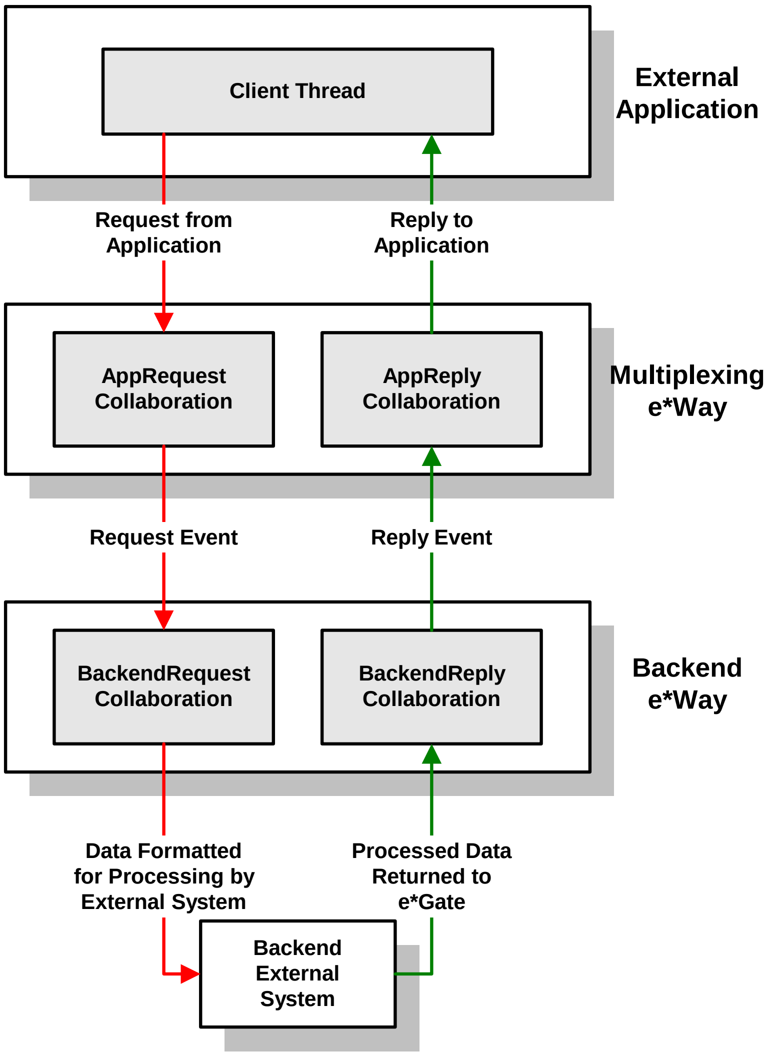
Request/Reply schemas have two classes of components:

1. "Front end" components that handle communications with the web server. 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 Apache Web Server 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 typical Request/Reply schema.

Figure 3 The Request/Reply schema



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

2 The Perl CGI script on the web server packages the data, invokes the e*Gate extension subroutines, and forwards the data to the Participating Host.

3 Data enters the Participating Host through the multiplexing 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 multiplexing e*Way subscribes to the Reply Event.

9 The multiplexing e*Way returns the processed data to the requesting thread in the web server.

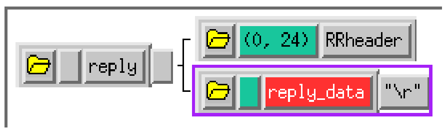
10 The web server completes the HTML page and returns the data to the client browser.

5.1.3 ETDs and Form Data

As discussed in **"Request/Reply and the Apache Web Server e*Way Participating Host Components" on page 25**, the Apache Web Server 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:

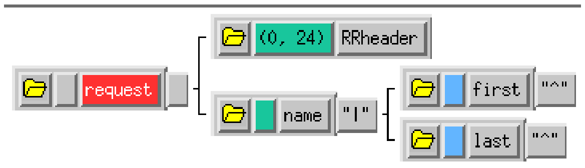
1 Be sure to maintain the 24-byte "header" node unchanged.

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

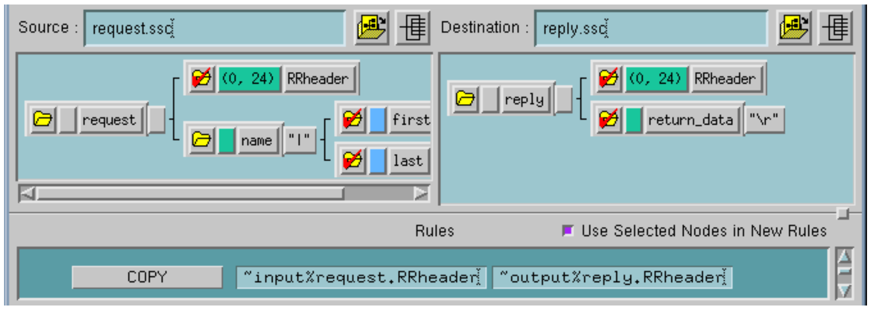


Of course, the more complex the form that collects user data, the more complex the ETD that describes the form data must become. Be sure that the Request ETD meets the requirements for input to the backend system.

Collaboration Rules and the Apache Web Server 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



COPY ~input%request.RRheader ~output%reply.RRheader

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 Apache Web Server e*Way as a single input string. Use periods (.) in your Perl Script to concatenate form data to create the e*Gate input string. For example, to create the "Last name^First name" string for the ETD illustrated in **Figure 5 on page 30**, use Perl code similar to the following:

```
# get form data
$firstname = $query->param('firstname');
$lastname = $query->param('firstname');

# concatenate with delimiter
$egateinput = $lastname . "^" . $firstname;
```

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

5.2 Using the CGI Script

The minimal implementation of the Apache Web Server e*Way via a web server requires two steps:

1. A page that contains a form for accepting user data
2. A Perl CGI script 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 CGI script, using the method POST or GET. For file uploading applications, make sure that the ENCTYPE attribute is set to multipart/form-data

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

The Perl script must do the following:

1. Use the Multiplexer_Init subroutine (see **"Multiplexer_Init" on page 16**) to specify the location of the stc_common and stc_ewipmpclnt **.dll** files
2. Define the host name and TCP/IP port numbers
3. Format the user data as appropriate for processing within e*Gate
4. Use the Multiplexer_Open subroutine (see **"Multiplexer_Open" on page 17**) to open a connection to the e*Gate Participating Host
5. Use the Multiplexer_Send subroutine (see **"Multiplexer_Send" on page 18**) to send data to the e*Gate Participating Host
6. Use the Multiplexer_Wait subroutine (see **"Multiplexer_Wait" on page 20**) to cause the Perl script to pause long enough for e*Gate to process and return the data
7. Use the Multiplexer_ToString subroutine (see **"Multiplexer_ToString" on page 19**) to obtain the returned data and display it within the user's browser
8. Use the Multiplexer_Free subroutine (see **"Multiplexer_Free" on page 15**) to free the memory associated with the returned data
9. Close the connection using the Multiplexer_Close subroutine (see **"Multiplexer_Close" on page 14**)

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

5.3 Sample Implementation

A sample schema for setting up Apache Web Server e*Way server is provided on the e*Gate installation CD-ROM in the directory

/samples/ewwebserver

1. Copy all the files from the e*Gate installation CD-ROM samples directory **/samples/ ewwebserver** to a temporary working directory on the system on which the Apache Web Server e*Way Web Server components are installed.
2. Copy the sample files **sample.pl** and **sample.html** to the Apache “CGI bin” directory. Depending on your installation, this may be **$APACHE_ROOT/cgi-bin** or a specific user’s **public_html** directory.
3. Modify the Perl script **sample.pl** so that path information in the following lines reflect the client installation:

```
# this is where stc_ewipmpclntperl.so is located.
use lib ("egate/client/bin");
# this is where stc_ewipmpclntperl.pm is located.
use lib ("egate/client/perl");
...

# this is where stc_common.dll and stc_ewipmpclnt.dll are located
stc_ewipmpclntperl::Multiplexer_Init("egate/client/bin");
# host name where web server e*way server is running
$ServerName = "localhost";# use default port number
# use default port number
$ServerPort = 0;
```

4. Change to the temporary working directory on the system where the Apache Web Server Participating Host components are installed.

Note: *A user cannot run **stcregutil.exe** on machines that do not have the e*Gate Participating Host installed.*

5. Use **stcregutil.exe** to import the RequestReply schema:

```
stcregutil.exe -rh localhost -rs RequestReply
    -un username -up passwd -i RequestReply.exp
```

6. Again, use **stcregutil.exe** to import the configuration files:

```
stcregutil.exe -rh localhost -rs RequestReply
    -un username -up passwd -fc . -ctl RequestReply.ctl
```

7. Start the Control Broker for this new schema:

```
stccb.exe -ln RequestReply_cb -rh localhost -rs RequestReply
    -un username -up passwd -sm
```

These instructions also appear in “readme” files in the samples directory (**/samples/ ewwebserver**).

Once the client and server are set up, you can test the entire system using a web browser:

1. Use a browser to open the **sample.html** file.
2. Fill out and submit the form.
3. Confirm that the **sample.pl** script returned the form data as expected.

5.3.1 The Sample Application and Null Characters

The sample application sends a file through the Apache Web Server e*Way and returns the file to your browser. However, if your file contains a null character ("\0"), the response message will be terminated by that null character.

In the current implementation, the null character will terminate a response message, and any information that follows that character will be discarded when you use **Multiplexer_ToString** to convert the response message to a string.

Index