SeeBeyond™ eBusiness Integration Suite

TCP/IP e*Way Intelligent Adapter User's Guide

Release 4.5.2

SEEBEYOND™

Version 20030204164518.

Contents

Chapter 1

Introduction

This chapter introduces you to SeeBeyond™ Technology Corporation's (SeeBeyond™) TCP/IP e*Way™ Intelligent Adapter.

1.1 Overview

The TCP/IP e*Way provides real-time, reliable data transfer for systems that support TCP/IP. This e*Way is enabled by the Monk programming language.

1.1.1 Intended Reader

The reader of this guide is presumed:

- To be a developer or system administrator with the responsibility for maintaining the e*Gate system
- To have a high-level knowledge of Windows NT and UNIX operations and administration
- To be thoroughly familiar with Windows-style GUI operations
- To be thoroughly familiar with the *Transmission Control Protocol/Internet Protocol* (TCP/IP)

1.1.2 Components

The following components comprise the TCP/IP e*Way:

- **stcewgenericmonk.exe**, the executable component
- Configuration files, which the e*Way Configuration Editor uses to define configuration parameters
- Monk function scripts, explained under **"TCP/IP e*Way Functions" on page 49**.

A complete list of installed files appears in **Table 1 on page 12**.

1.2 Supported Operating Systems

The Monk-enabled TCP/IP e*Way is available on the following operating systems:

- Windows XP
- Windows 2000, Windows 2000 SP1, Windows 2000 SP2, and Windows 2000 SP3
- Windows NT 4.0 SP6a
- Solaris 2.6, 7, and 8
- AIX 4.3.3 and AIX 5.1
- HP-UX 11.0 and HP-UX 11i
- Compaq Tru64 V4.0F, V5.0A, and V5.1A
- Red Hat Linux 6.2
- OS/390 V2, R10
- z/OS 1.2, 1.3, and 1.4
- Japanese Windows 2000, Windows 2000 SP1, Windows 2000 SP2, and Windows 2000 SP3
- Japanese Windows NT 4.0 SP6a
- Japanese Solaris 2.6, 7, and 8
- Japanese HP-UX 11.0
- Korean Solaris 8

1.3 System Requirements

To use the TCP/IP e*Way, you need to meet the following requirements:

- An e*Gate Participating Host, version 4.5.1 or later; for Windows XP operating systems, you need an e*Gate Participating Host, version 4.5.3 or later
- A TCP/IP network connection

Note: *To use this e*Way with Compaq Tru64, you need the Compaq Tru64 UNIX patches. See the **e*Gate Integrator Installation Guide** for details.*

The e*Way must be configured and administered using the e*Gate Enterprise Manager.

Note: *Additional disk space can be 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 doing the processing.*

External System Requirements

To enable the e*Way to communicate properly with the TCP/IP system, you need:

- Host on which the server is running
- Port location on which the server is listening

Chapter 2

Installation

This chapter describes how to install the TCP/IP e*Way.

2.1 Installation on Windows Systems

2.1.1 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.

2.1.2 Installation Procedure

To install the TCP/IP 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 is launched. Follow the on-screen instructions to install the e*Way.

Be sure to install the e*Way files in the suggested "client" installation directory. The installation utility detects and suggests the appropriate installation directory.

Caution: *Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested "installation directory" setting.*

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 Configuration Editor, see the **e*Gate Integrator User's Guide**.

2.2 UNIX Installation

2.2.1 Pre-installation

- You do not require root privileges to install this e*Way. Log in under the user name that you wish to own the e*Way files. Be sure that this user has sufficient privilege to create files in the e*Gate directory tree.

2.2.2 Installation Procedure

To install the TCP/IP e*Way on a UNIX system

1. Log in on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.
2. If necessary, mount the CD-ROM drive.
3. At the shell prompt, type

   cd /cdrom

4. Start the installation script by typing:

   setup.sh

5. A menu of options appears. Select the "install e*Way" option. Then, follow any additional on-screen directions.

Be sure to install the e*Way files in the suggested "client" installation directory. The installation utility detects and suggests the appropriate installation directory.

Caution: *Unless you are directed to do so by SeeBeyond support personnel, do not change the suggested "installation directory" setting.*

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 Configuration Editor, see the *e*Gate Integrator User's Guide*.

2.3 Installing on OS/390 V2R10

Please see the *e*Gate Integrator Installation Guide* for procedures on how to install this e*Way on the OS/390 operating system.

2.4 Files/Directories Created by the Installation

The TCP/IP 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 Created by Installation

Directories	Files
bin\	stcewgenericmonk.exe
bin\	stc_monkfilesys.dll stc_monktcpip.dll
configs\stcewgenericmonk	stcewtcpipext.def
monk_library\ewtcpipext	tcpip-ack.monk tcpip-exchange.monk tcpip-extconnect.monk tcpip-init.monk tcpip-nack.monk tcpip-notify.monk tcpip-outgoing.monk tcpip-shutdown.monk tcpip-startup.monk tcpip-verify.monk tcpip-server-verify.monk tcpip-server-startup.monk tcpip-server-shutdown.monk tcpip-server-outgoing.monk tcpip-server-notify.monk tcpip-server-nack.monk tcpip-server-init.monk tcpip-server-extconnect.monk tcpip-server-exchange.monk tcpip-server-ack.monk

Chapter 3

Configuration

This chapter describes how to configure the TCP/IP e*Way.

3.1 e*Way Configuration Parameters

e*Way configuration parameters are set using the e*Way Configuration 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. Select the executable file **stcewgenericmonk.exe**.
3. 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.
4. 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 Configuration Editor, see the e*Way Configuration Editor's online Help or the *e*Gate Integrator User's Guide*.

Note: *All parameter values are mandatory, except for the Monk Positive Acknowledgment and Negative Acknowledgment functions, for both the TCP/IP client and server. See* **"Positive Acknowledgment Function" on page 30** *and* **"Negative Acknowledgment Function" on page 30**.

The e*Way's configuration parameters are organized into the following sections:

- General Settings
- Communication Setup
- Monk Configuration
- TCP/IP Configuration
- TCP/IP Server Configuration

3.1.1 General Settings

The General Settings control basic operational parameters.

Journal File Name

Description

Specifies the name of the journal file.

Required Values

A valid file name, optionally including an absolute path (for example, **c:\temp\filename.txt**). If an absolute path is not specified, the file is stored in the e*Gate **SystemData** directory. See the *e*Gate Integrator System Administration and Operations Guide* for more information about file locations.

Additional Information

An Event is journaled for the following conditions:

- When the number of resends is exceeded (see **Max Resends Per Message** in the next section)
- When its receipt is due to an external error, but **Forward External Errors** is set to **No**. (See **"Forward External Errors" on page 14** for more information.)

Max Resends Per Message

Description

Specifies the number of times the e*Way attempts to resend a message (Event) to the external system after receiving an error.

Required Values

An integer between 1 and 1,024. The default is 5.

Max Failed Messages

Description

Specifies the maximum number of failed messages (Events) that the e*Way allows. When the specified number of failed messages is reached, the e*Way shuts down and exits.

Required Values

An integer between 1 and 1,024. The default is 3.

Forward External Errors

Description

Selects whether error messages that begin with the string "DATAERR" that are received from the external system are queued to the e*Way's configured queue. See **"Exchange Data with External Function" on page 27** for more information.

Required Values

Yes or **No**. The default value, **No**, specifies that error messages are not forwarded.

See **“Schedule-driven data exchange functions” on page 22** for information about how the e*Way uses this function.

3.1.2 Communication Setup

The Communication Setup parameters control the schedule by which the e*Way obtains data from the external system.

Note: *The schedule you set using the e*Way’s properties in the Enterprise Manager controls when the e*Way executable runs. The schedule you set within the parameters discussed in this section (using the e*Way Configuration Editor) determines when data is exchanged. Be sure you set the "exchange data" schedule to fall within the "run the executable" schedule.*

Start Exchange Data Schedule

Description

Establishes the schedule to invoke the e*Way’s **Exchange Data with External** function.

Required Values

One of the following:

- One or more specific dates/times
- A single repeating interval (such as yearly, weekly, monthly, daily, or every *n* seconds).

Also required: If you set a schedule using this parameter, you must also define all three of the following:

- Exchange Data With External Function
- Positive Acknowledgment Function
- Negative Acknowledgment Function

If you do not do so, the e*Way terminates execution when the schedule attempts to start.

Additional Information

When the schedule starts, the e*Way determines whether it is waiting to send an ACK or NAK to the external system (using the Positive and Negative Acknowledgment functions) and whether the connection to the external system is active. If no ACK/NAK is pending and the connection is active, the e*Way immediately executes the **Exchange Data with External** function. Thereafter, the **Exchange Data with External** function is called according to the **Exchange Data Interval** parameter until the **Stop Exchange Data Schedule** time is reached.

See **“Exchange Data with External Function” on page 27**, **“Exchange Data Interval” on page 16**, and **“Stop Exchange Data Schedule” on page 16** for more information.

Stop Exchange Data Schedule

Description

Establishes the schedule to stop data exchange.

Required Values

One of the following values:

- One or more specific dates/times
- A single repeating interval (such as yearly, weekly, monthly, daily, or every *n* seconds).

Exchange Data Interval

Description

Specifies the number of seconds the e*Way waits between calls to the **Exchange Data with External** function during scheduled data exchanges.

Required Values

An integer, 0 to 86,400. The default is 120.

Additional Information

If **Zero Wait Between Successful Exchanges** (see **"Zero Wait Between Successful Exchanges" on page 17**) is set to **Yes** and the **Exchange Data with External Function** returns data, the **Exchange Data Interval** setting is ignored, and the e*Way invokes the **Exchange Data with External Function** immediately.

If this parameter is set to zero, there is no exchange data schedule set and the **Exchange Data with External Function** is never called.

See **"Down Timeout" on page 16** and **"Stop Exchange Data Schedule" on page 16** for more information about the data exchange schedule.

Down Timeout

Description

Specifies the number of seconds that the e*Way waits between calls to the **External Connection Establishment** function. See **"External Connection Establishment Function" on page 28** for more information.

Required Values

An integer between 1 and 86,400. The default is 15.

Up Timeout

Description

Specifies the number of seconds the e*Way waits between calls to the **External Connection Verification** function. See **"External Connection Verification Function" on page 29** for more information.

Required Values

An integer between 1 and 86,400. The default is 15.

Resend Timeout

Description

Specifies the number of seconds the e*Way waits between attempts to resend a message (Event) to the external system, after receiving an error message from the external system.

Required Values

An integer between 1 and 86,400. The default is 10.

Zero Wait Between Successful Exchanges

Description

Selects whether to initiate data exchange after the **Exchange Data Interval** or immediately after a successful previous exchange.

Required Values

Yes or **No**. If this parameter is set to **Yes**, the e*Way immediately invokes the **Exchange Data with External** function if the previous exchange function returned data. If this parameter is set to **No**, the e*Way always waits the number of seconds specified by **Exchange Data Interval** between invocations of the **Exchange Data with External** function. The default is **No**.

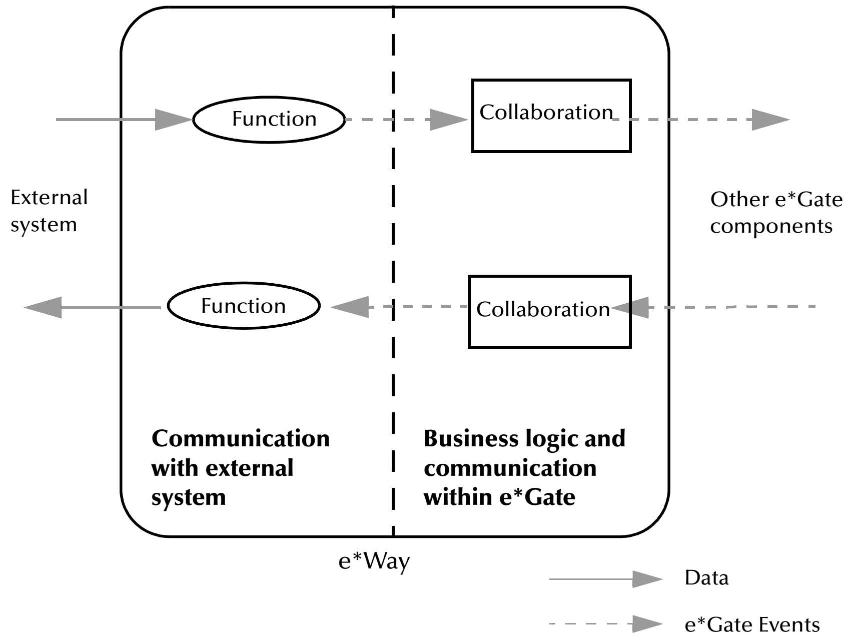
See **"Exchange Data with External Function" on page 27** for more information.

3.1.3 Monk Configuration

The parameters in this section help you set up the information required by the e*Way to utilize Monk for communication with the external system.

Conceptually, an e*Way is divided into two halves. One half of the e*Way (shown on the left in Figure 1) handles communication with the external system; the other half manages the Collaborations that process data and subscribe or publish to other e*Gate components.

Figure 1 e*Way internal architecture



The “communications half” of the e*Way uses Monk functions to start and stop scheduled operations, exchange data with the external system, package data as e*Gate “Events” and send those Events to Collaborations, and manage the connection between the e*Way and the external system. The **Monk Configuration** options discussed in this section control the Monk environment and define the Monk functions used to perform these basic e*Way operations. You can create and modify these functions using the Collaboration Rules Editor or a text editor (such as **Notepad**, or UNIX **vi**).

The “communications half” of the e*Way is single-threaded. Functions run serially, and only one function can be executed at a time. The “business logic” side of the e*Way is multi-threaded, with one executable thread for each Collaboration. Each thread maintains its own Monk environment; therefore, information such as variables, functions, path information, and so on cannot be shared between threads.

Operational Details

The Monk functions in the "communications half" of the e*Way fall into the following groups:

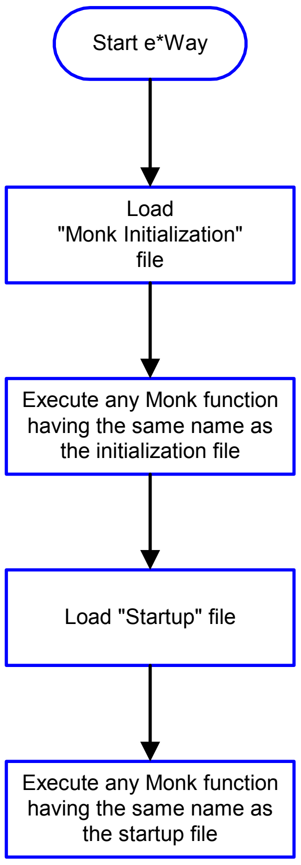
Type of Operation	Name
Initialization	**Startup Function** on page 26 (also see **Monk Environment Initialization File** on page 26)
Connection	**External Connection Establishment Function** on page 28 **External Connection Verification Function** on page 29 **External Connection Shutdown Function** on page 29
Schedule-driven data exchange	**Exchange Data with External Function** on page 27 **Positive Acknowledgment Function** on page 30 **Negative Acknowledgment Function** on page 30
Shutdown	**Shutdown Command Notification Function** on page 31
Event-driven data exchange	**Process Outgoing Message Function** on page 27

A series of figures on the next several pages illustrates the interaction and operation of these functions.

Initialization Functions

Figure 2 on page 20 illustrates how the e*Way executes its initialization functions.

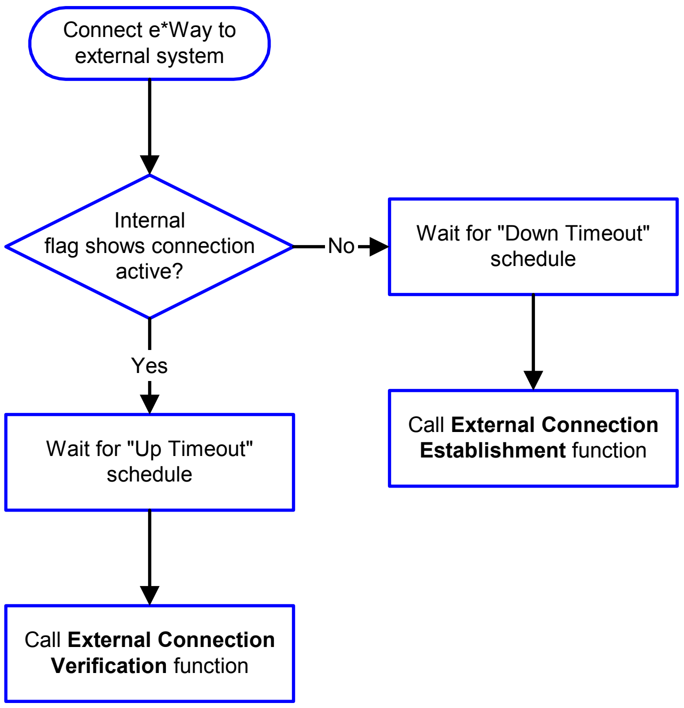
Figure 2 Initialization Functions


Start e*Way
Load "Monk Initialization" file
Execute any Monk function having the same name as the initialization file
Load "Startup" file
Execute any Monk function having the same name as the startup file

Connection Functions

Figure 3 on page 21 illustrates how the e*Way executes the connection establishment and verification functions.

Figure 3 Connection establishment and verification functions

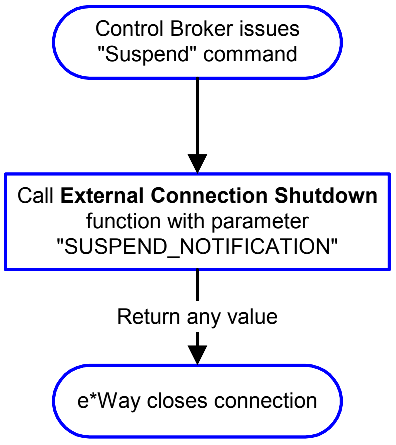

Connect e*Way to external system
Internal flag shows connection active?
No
Wait for "Down Timeout" schedule
Call **External Connection Establishment** function
Yes
Wait for "Up Timeout" schedule
Call **External Connection Verification** function

Note: *The e*Way selects the connection function based on an internal "up/down" flag rather than a poll to the external system. See* **Figure 5 on page 22** *and* **Figure 7 on page 24** *for examples of how different functions use this flag.*

User functions can manually set this flag using Monk functions. See **send-external-up** *on page 51 and* **send-external-down** *on page 50 for more information.*

Figure 4 illustrates how the e*Way executes its "connection shutdown" function.

Figure 4 Connection shutdown function


Control Broker issues "Suspend" command
Call **External Connection Shutdown** function with parameter "SUSPEND_NOTIFICATION"
Return any value
e*Way closes connection

Schedule-driven Data Exchange Functions

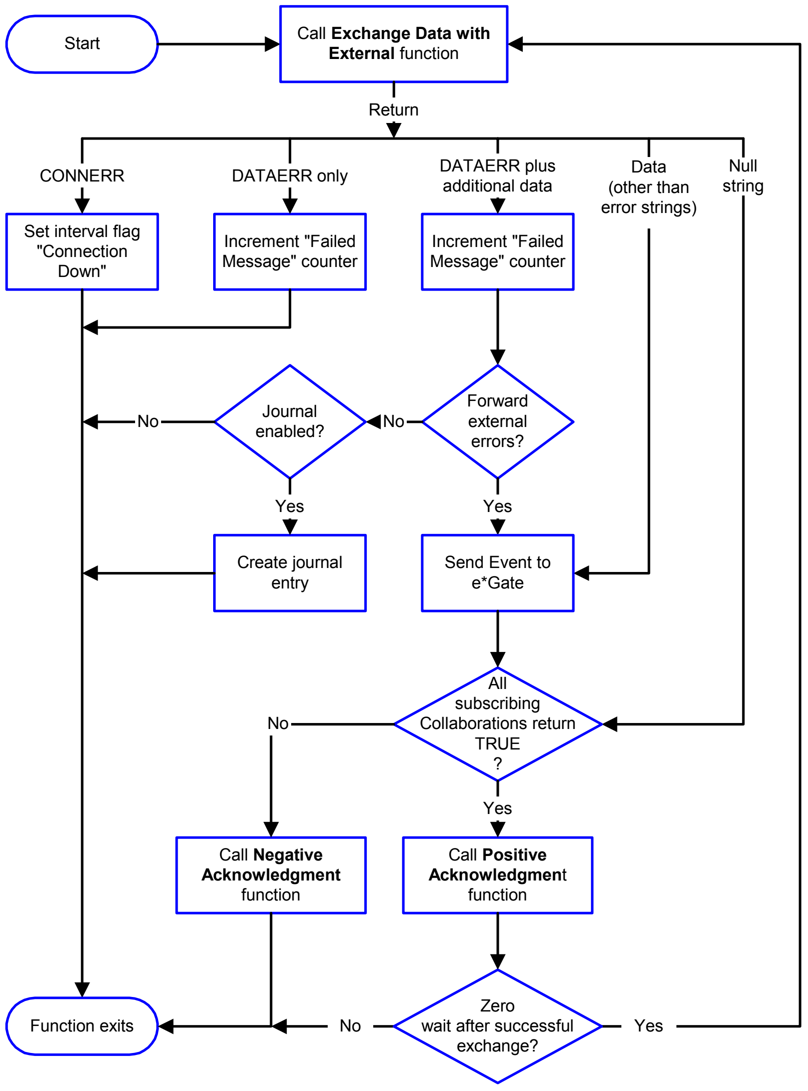
Figure 5 illustrates how the e*Way performs schedule-driven data exchange using the **Exchange Data with External Function**. The **Positive Acknowledgment Function** and **Negative Acknowledgment Function** are also called during this process.

“Start” can occur in any of the following ways:

- The “Start Data Exchange” time occurs
- Periodically during data-exchange schedule (after “Start Data Exchange” time, but before “Stop Data Exchange” time), as set by the Exchange Data Interval
- The **start-schedule** Monk function is called

After the function exits, the e*Way waits for the next “start schedule” time or command.

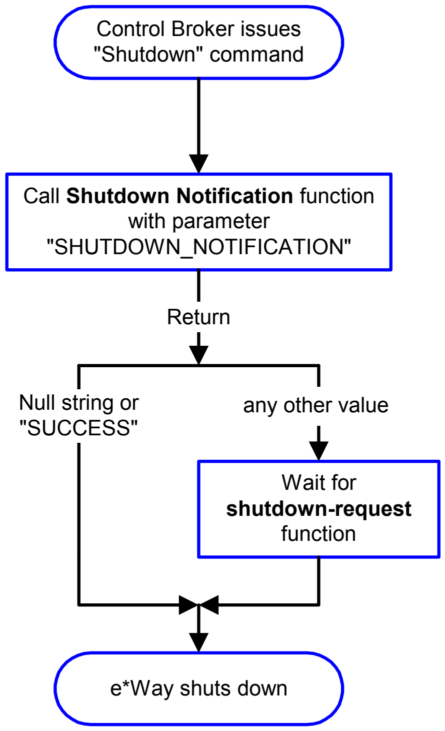
Figure 5 Schedule-driven data exchange functions


Start
Call Exchange Data with External function
Return
CONNERR
DATAERR only
DATAERR plus additional data
Data (other than error strings)
Null string
Set interval flag "Connection Down"
Increment "Failed Message" counter
Increment "Failed Message" counter
Journal enabled?
Forward external errors?
No
No
Yes
Yes
Create journal entry
Send Event to e*Gate
All subscribing Collaborations return TRUE ?
No
Yes
Call Negative Acknowledgment function
Call Positive Acknowledgment function
Zero wait after successful exchange?
No
Yes
Function exits

Shutdown Functions

Figure 6 illustrates how the e*Way implements the shutdown request function.

Figure 6 Shutdown functions


Control Broker issues "Shutdown" command
Call **Shutdown Notification** function with parameter "SHUTDOWN_NOTIFICATION"
Return
Null string or "SUCCESS"
any other value
Wait for **shutdown-request** function
e*Way shuts down

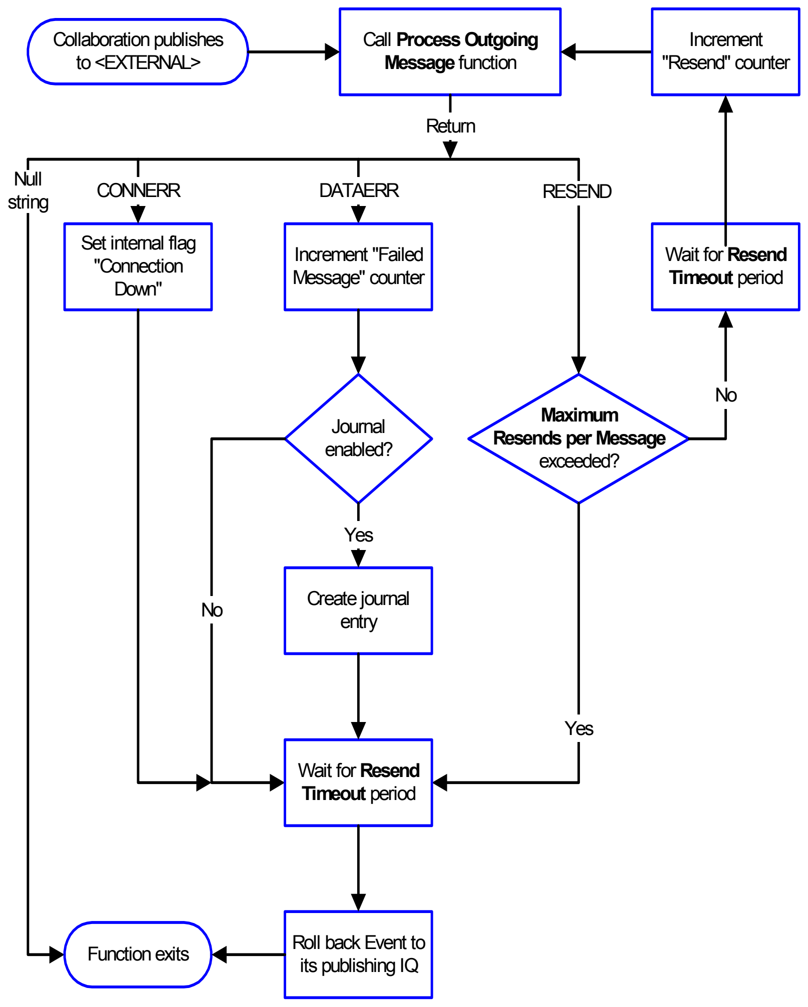
Event-driven Data Exchange Functions

Figure 7 on page 24 illustrates event-driven data-exchange using the **Process Outgoing Message Function**.

Every two minutes, the e*Way checks the “Failed Message” counter against the value specified by the **Max Failed Messages** parameter. When the “Failed Message” counter exceeds the specified maximum value, the e*Way logs an error and shuts down.

After the function exits, the e*Way waits for the next outgoing Event.

Figure 7 Event-driven data-exchange functions


Collaboration publishes to <EXTERNAL>
Call **Process Outgoing Message** function
Increment "Resend" counter
Return
Null string
CONNERR
DATAERR
RESEND
Set internal flag "Connection Down"
Increment "Failed Message" counter
Wait for **Resend Timeout** period
Journal enabled?
Maximum Resends per Message exceeded?
No
Yes
Create journal entry
No
Yes
Wait for **Resend Timeout** period
Function exits
Roll back Event to its publishing IQ

How to Specify Function Names or File Names

Parameters that require the name of a Monk function accept either a function name or a file name. If you specify a file name, be sure that the file has one of the following extensions:

- **.monk**
- **.tsc**
- **.dsc**

Note: *When the e*Way is configured as a server, the default functions must be changed to the server functions. For example,* ***tcpip-init*** *would be changed to* ***tcpip-server-init****.*

Additional Path

Description

Specifies a path to be appended to the "load path," the path Monk uses to locate files and data (set internally within Monk). The directory specified in **Additional Path** is searched after the default load paths.

Required Values

A pathname, or a series of paths separated by semicolons. This parameter is optional and may be left blank.

Additional information

The default load paths are determined by the "bin" and "Shared Data" settings in the .egate.store file. See the *e*Gate Integrator System Administration and Operations Guide* for more information about this file.

To specify multiple directories, manually enter the directory names rather than selecting them with the "file selection" button. Directory names must be separated with semicolons, and you can mix absolute paths with relative e*Gate paths. For example:

```
monk_scripts\my_dir;c:\my_directory
```

The internal e*Way function that loads this path information is called only once, when the e*Way first starts up.

Auxiliary Library Directories

Description

Specifies a path to auxiliary library directories. Any **.monk** files found within those directories are automatically loaded into the e*Way's Monk environment. This parameter is optional and may be left blank.

Required Values

A pathname, or a series of paths separated by semicolons. The default is **monk_library/ewtcpipext**.

Additional information

To specify multiple directories, manually enter the directory names rather than selecting them with the "file selection" button. Directory names must be separated with semicolons, and you can mix absolute paths with relative e*Gate paths. For example:

```
monk_scripts\my_dir;c:\my_directory
```

The internal e*Way function that loads this path information is called only once, when the e*Way first starts up.

This parameter is optional and may be left blank.

Monk Environment Initialization File

Specifies a file that contains environment initialization functions, which are loaded after the auxiliary library directories are loaded. Use this feature to initialize the e*Way's Monk environment (for example, to define Monk variables that are used by the e*Way's function scripts).

Required Values

A filename within the "load path", or filename plus path information (relative or absolute). If path information is specified, that path is appended to the "load path." See **"Additional Path" on page 25** for more information about the "load path." The default is **tcpip-init.monk**. (See **tcpip-init** on page 55 or **tcpip-server-init** on page 59 for more information.)

Additional information

Any environment-initialization functions called by this file accept no input, and must return a string. The e*Way loads this file and tries to invoke a function of the same base name as the file name (for example, for a file named **my-init.monk**, the e*Way would attempt to execute the function **my-init**).

Typically, it is a good practice to initialize any global Monk variables that may be used by any other Monk Extension scripts.

The internal function that loads this file is called once when the e*Way first starts up (see **Figure 2 on page 20**).

Note: *When the e*Way is configured as a server, the default functions must be changed to the server functions. For example, **tcpip-init** would be changed to **tcpip-server-init**.*

Startup Function

Description

Specifies a Monk function that the e*Way loads and invoke upon startup or whenever the e*Way's configuration is reloaded. This function should be used to initialize the external system before data exchange starts.

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. This parameter is optional and may be left blank. The default is **tcpip-startup**. (See **tcpip-startup** on page 64 or **tcpip-server-startup** on page 63 for more information.)

Additional information

The function accepts no input, and must return a string.

The string "FAILURE" indicates that the function failed; any other string (including a null string) indicates success.

This function is called after the e*Way loads the specified "Monk Environment Initialization file" and any files within the specified **Auxiliary Directories**.

The e*Way loads this file and tries to invoke a function of the same base name as the file name (see **Figure 2 on page 20**). For example, for a file named **my-startup.monk**, the e*Way would attempt to execute the function **my-startup**.

Process Outgoing Message Function

Description

Specifies the Monk function responsible for sending outgoing messages (Events) from the e*Way to the external system. This function is event-driven (unlike the Exchange Data with External function, which is schedule-driven).

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. ***You may not leave this field blank.*** The default is **tcpip-outgoing**. (See **tcpip-outgoing** on page 57 or **tcpip-server-outgoing** on page 61 for more information.)

Additional Information

The function requires a non-null string as input (the outgoing Event to be sent) and must return a string.

The e*Way invokes this function when one of its Collaborations publishes an Event to an <EXTERNAL> destination (as specified within the Enterprise Manager). The function returns one of the following (see **Figure 7 on page 24** for more details):

- Null string: Indicates that the Event was published successfully to the external system.
- “RESEND”: Indicates that the Event should be resent.
- “CONNERR”: Indicates that there is a problem communicating with the external system.
- “DATAERR”: Indicates that there is a problem with the message (Event) data itself.

If a string other any in the previous list is returned, the e*Way creates an entry in the log file indicating that an attempt has been made to access an unsupported function.

Note: *If you wish to use **event-send-to-egate** to enqueue failed Events in a separate IQ, the e*Way must have an inbound Collaboration (with appropriate IQs) configured to process those Events. See **event-send-to-egate** on page 49 for more information.*

Exchange Data with External Function

Description

Specifies a Monk function that initiates the transmission of data from the external system to the e*Gate system and forwards that data as an inbound Event to one or more e*Gate Collaborations. This function is called according to a schedule (unlike the **Process Outgoing Message Function**, which is event-driven).

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. This parameter is optional and may be left blank. The default is **tcpip-exchange**. (See **tcpip-exchange** on page 54 or **tcpip-server-exchange** on page 58 for more information.)

Additional Information

The function accepts no input and must return a string (see **Figure 5 on page 22** for more details):

- Null string: Indicates that the data exchange was completed successfully. No information is sent into the e*Gate system.
- "CONNERR": Indicates that a problem with the connection to the external system has occurred.
- "DATAERR": Indicates that a problem with the data itself has occurred. The e*Way handles the string "DATAERR" and "DATAERR" plus additional data differently; see **Figure 5 on page 22** for more details.
- Any other string: The contents of the string are packaged as an inbound Event. The e*Way must have at least one Collaboration configured suitably to process the inbound Event, as well as any required IQs.

This function is initially triggered by the **Start Data Exchange** schedule or manually by the Monk function **start-schedule**. After the function has returned true and the data received by this function has been ACKed or NAKed (by the **Positive Acknowledgment Function** or **Negative Acknowledgment Function**, respectively), the e*Way checks the **Zero Wait Between Successful Exchanges** parameter. If this parameter is set to **Yes**, the e*Way immediately calls the **Exchange Data with External** function again; otherwise, the e*Way does not call the function until the next scheduled "start exchange" time or the schedule is manually invoked using the Monk function **start-schedule** (see **start-schedule** on page 52 for more information).

External Connection Establishment Function

Description

Specifies a Monk function that the e*Way calls when it has determined that the connection to the external system is down.

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. ***This field cannot be left blank.*** The default is **tcpip-extconnect**. (See **tcpip-extconnect** on page 54 or **tcpip-server-extconnect** on page 59 for more information.)

Additional Information

The function accepts no input and must return a string:

- "SUCCESS" or "UP": Indicates that the connection was established successfully.
- Any other string (including the null string): Indicates that the attempt to establish the connection failed.

This function is executed according to the interval specified within the **Down Timeout** parameter, and is *only* called according to this schedule.

The **External Connection Verification** function (see below) is called when the e*Way has determined that its connection to the external system is up.

External Connection Verification Function

Description

Specifies a Monk function that the e*Way calls when its internal variables show that the connection to the external system is up.

Required Values

The name of a Monk function. This function is optional; if no **External Connection Verification** function is specified, the e*Way executes the **External Connection Establishment** function in its place. The default is **tcpip-verify**. (See **tcpip-verify** on page 65 or **tcpip-server-verify** on page 63 for more information.)

Additional Information

The function accepts no input and must return a string:

- "SUCCESS" or "UP": Indicates that the connection was established successfully.
- Any other string (including the null string): Indicates that the attempt to establish the connection failed.

This function is executed according to the interval specified within the **Up Timeout** parameter, and is *only* called according to this schedule.

The **External Connection Establishment** function (see above) is called when the e*Way has determined that its connection to the external system is down.

External Connection Shutdown Function

Description

Specifies a Monk function that the e*Way calls to shut down the connection to the external system.

Required Values

The name of a Monk function. The default is **tcpip-shutdown**. (See **tcpip-shutdown** on page 64 or **tcpip-server-shutdown** on page 62 for more information.)

Additional Information

This function requires a string as input, and may return a string.

This function is only invoked when the e*Way receives a "suspend" command from a Control Broker. When the "suspend" command is received, the e*Way invokes this function, passing the string "SUSPEND_NOTIFICATION" as an argument.

Any return value indicates that the "suspend" command can proceed and that the connection to the external system can be broken immediately.

Positive Acknowledgment Function

Description

Specifies a Monk function that the e*Way calls when ***all*** the Collaborations to which the e*Way sent data have processed and enqueued that data successfully.

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. This parameter is required if the **Exchange Data with External** function is defined. The default is **tcpip-ack**. (See **tcpip-ack** on page 53 or **tcpip-server-ack** on page 58 for more information.)

Additional Information

The function requires a non-null string as input (the Event to be sent to the external system) and must return a string:

- "CONNERR": Indicates a problem with the connection to the external system. When the connection is re-established, the Positive Acknowledgment function is called again, with the same input data.
- Null string: The function completed execution successfully.

After the **Exchange Data with External** function returns a string that is transformed into an inbound Event, the Event is handed off to one or more Collaborations for further processing. If the Event's processing is completed successfully by *all* the Collaborations to which it was sent, the e*Way executes the Positive Acknowledgment function (otherwise, the e*Way executes the Negative Acknowledgment function).

Negative Acknowledgment Function

Description

Specifies a Monk function that the e*Way calls when the e*Way fails to process and queue Events from the external system.

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. This parameter is required if the **Exchange Data with External** function is defined. The default is **tcpip-nack**. (See **tcpip-nack** on page 55 or **tcpip-server-nack** on page 60 for more information.)

Additional Information

The function requires a non-null string as input (the Event to be sent to the external system) and must return a string:

- "CONNERR": Indicates a problem with the connection to the external system. When the connection is re-established, the function is called again.
- Null string: The function completed execution successfully.

This function is only called during the processing of inbound Events. After the **Exchange Data with External** function returns a string that is transformed into an inbound Event, the Event is handed off to one or more Collaborations for further processing. If the Event's processing is not completed successfully by *all* the Collaborations to which it was sent, the e*Way executes the Negative Acknowledgment function (otherwise, the e*Way executes the Positive Acknowledgment function).

Shutdown Command Notification Function

Description

Specifies a Monk function that is called when the e*Way receives a "shut down" command from the Control Broker. This parameter is optional.

Required Values

The name of a Monk function. (See **tcpip-notify** on page 56 or **tcpip-server-notify** on page 60 for more information.)

Additional Information

When the Control Broker issues a shutdown command to the e*Way, the e*Way calls this function with the string "SHUTDOWN_NOTIFICATION" passed as a parameter.

The function accepts a string as input and must return a string:

- A null string or "SUCCESS": Indicates that the shutdown can occur immediately.
- Any other string: Indicates that shutdown must be postponed. Once postponed, shutdown does not proceed until the Monk function **shutdown-request** is executed (see **shutdown-request** on page 51).

Note: *If you postpone a shutdown using this function, be sure to use the (**shutdown-request)** function to complete the process in a timely manner.*

3.1.4 TCP/IP Configuration

This section defines the TCP/IP parameters used in when the e*Way is acting as a client.

Host

Description

Specifies the Host on which the server is running. This parameter is ***mandatory***.

Required Values

A string containing a valid hostname.

Port

Description

Specifies the port on which the server is listening for connection requests. This parameter is ***mandatory***.

Required Values

An integer between **1** and **864000**. The default is **8888**.

PacketSize

Description

Specifies the number of bytes per packet of data. This number also determines the size of the buffers. This parameter is ***mandatory***.

Required Values

An integer between **1** and **864000**. The default is **4096**.

Timeout

Description

Specifies the amount of time, in milliseconds, the e*Way awaits a response when making requests to the server.

Required Values

An integer between **1** and **864000**. The default is **50000**.

NoDelay

Description

Specifies whether the system can delay connections or requests. Generally, **NoDelay/True** is necessary for high-volume and/or critical transactions. In cases of low-volume and/or noncritical transactions, you can use **NoDelay/False**. This parameter is ***mandatory***.

Required Values

True or **False**. The default is **True**.

ACKValue

Description

Specifies the positive acknowledgment return value.

Required Values

A string.

NACKValue

Description

Specifies the negative acknowledgment return value.

Required Values

A string.

3.1.5 TCP/IP Server Configuration

This section defines the TCP/IP parameters used when the e*Way is acting as a server.

Host

Description

Specifies the host on which the server is running. This parameter is ***mandatory***.

Required Values

A string containing a valid hostname.

Port

Description

Specifies the port on which the server is listening for connection requests. This parameter is ***mandatory***.

Required Values

An integer between **1** and **864000**. The default is **8888**.

PacketSize

Description

Specifies the number of bytes per packet of data. This number also determines the size of the buffers.

Required Values

An integer between **1** and **864000**. The default is **4096**. This parameter is ***mandatory***.

MaxConnections

Descriptions

Specifies the maximum number of client connections that the server can accommodate.

Required Values

An integer between **1** and **1024**. The default is **5**.

WaitForClientTimeout

Description

Specifies the amount of time, in milliseconds, used when waiting for a new client connection.

Required Values

An integer between **1000** and **300000**. The default is **1000**.

Process WaitTime

Description

Specifies the amount of time, in milliseconds, to wait before checking for new client connections.

Required Values

An integer between **1000** and **300000**. The default is **1000**.

Timeout

Description

Specifies the amount of time in milliseconds used when receiving client messages (Events) and sending messages (Events) to clients.

Required Values

An integer between **1** and **864000**. The default is **50000**.

NoDelay

Description

Specifies whether the system can delay connections or requests. Generally, **NoDelay/True** is necessary for high-volume and/or critical transactions. In cases of low-volume and/or noncritical transactions, you can use **NoDelay/False**. This parameter is ***mandatory***.

Required Values

True or **False**. The default is **True**.

ACKValue

Description

Specifies the positive acknowledgment return value.

Required Values

A string.

NACKValue

Description

Specifies the negative acknowledgment return value.

Required Values

A string.

3.2 External Configuration Requirements

There are no configuration changes required in the external system. All necessary configuration changes can be made within e*Gate.

Chapter 4

Implementation

This chapter explains how to implement a sample schema with the TCP/IP e*Way.

4.1 Implementation Process: Overview

To implement the sample schema that uses the TCP/IP e*Way, do the following steps:

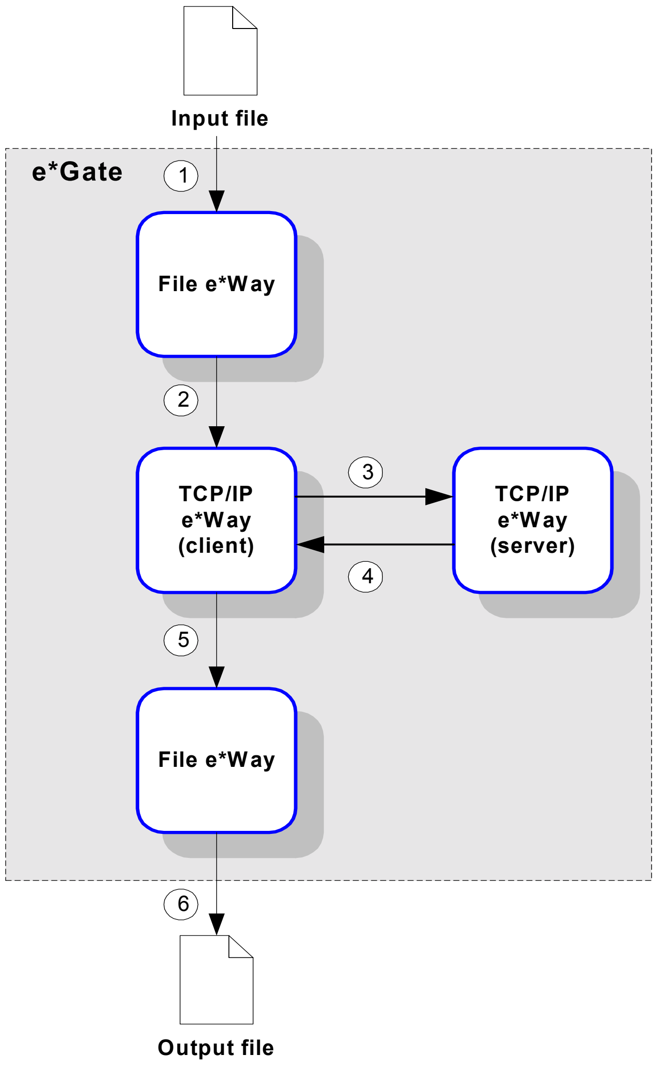
- Create a new schema.
- In the e*Gate Integrator Enterprise Manager, do the following:
  - Define Event Type Definitions (ETDs) to package the data being exchanged with the external system
  - Define Collaboration Rules to process Event data.
  - Define any IQs to which Event data is published prior to sending it to the external system.
  - Define the e*Way components (this procedure is discussed in **Chapter 2**).
  - Within each e*Way component, configure Collaborations to apply the required Collaboration Rules.
- Use the e*Way Configuration Editor to set the e*Way's configuration parameters.
- Be sure that any other e*Gate components are configured as necessary to complete the schema.
- Test the schema and make any necessary corrections.

Note: *For more information about creating or modifying any component within the e*Gate Enterprise Manager, see the e*Gate Integrator User's Guide or the Enterprise Manager's online Help.*

The sample schema consists of four e*Ways that operate as follows: one reads an Event from an external text file and forwards the Event to an e*Way configured as a TCP/IP client. The TCP/IP client, in turn, sends the Event to an e*Way configured as a TCP/IP server, which appends the Event with additional data and sends it back to the client. The client sends the appended Event to the fourth e*Way, which stores the Event in an output file.

Figure 8 illustrates the flow of data in the sample schema.

Figure 8 Data Flow in the Sample Schema



Generally, the sample schema operates as follows:

1 A file e*Way reads in data from a text file.

2 The file e*Way sends the data to a TCP/IP e*Way configured as a client.

3 The client e*Way sends the data to a TCP/IP e*Way configured as a server.

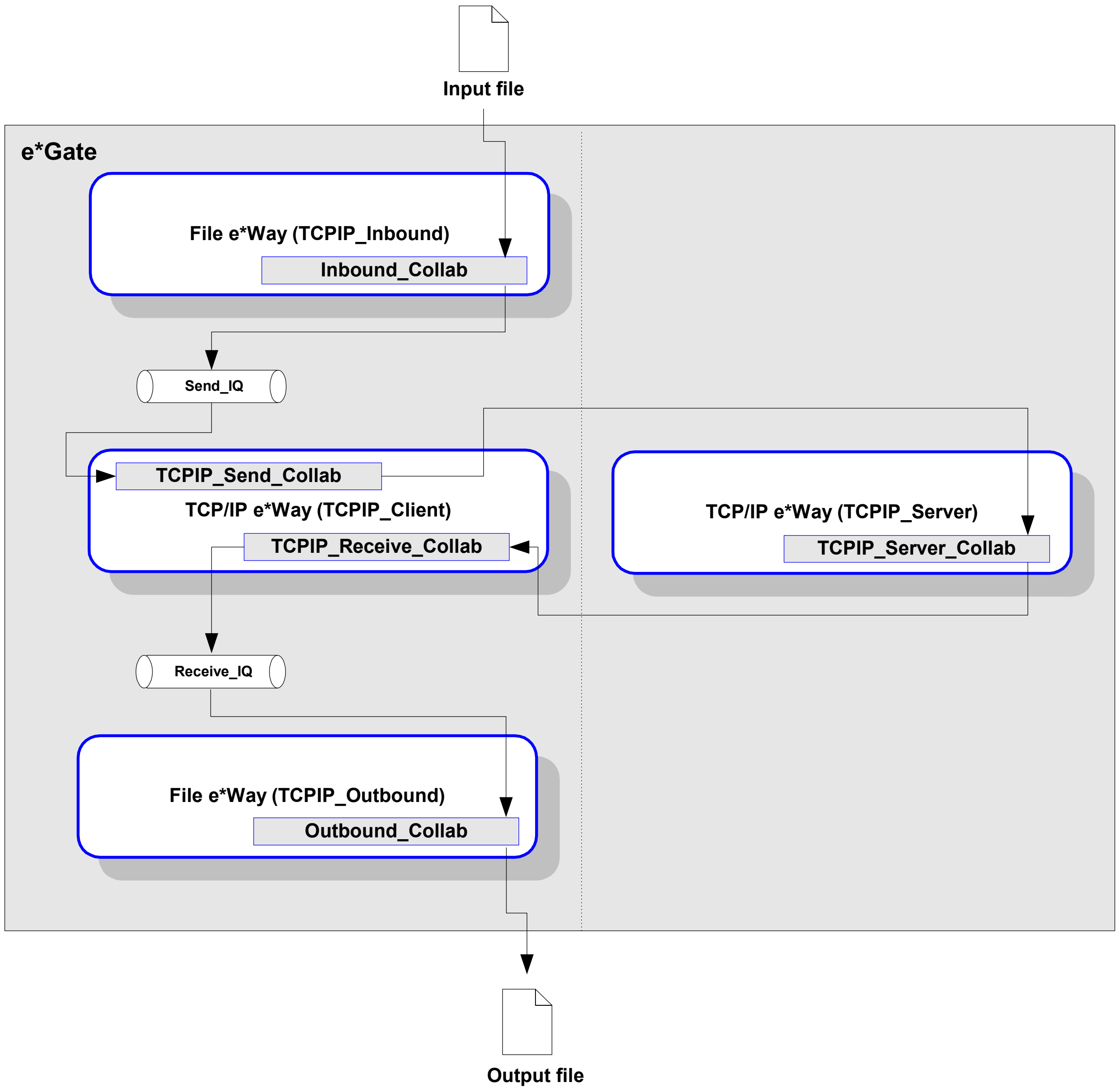
4 The server e*Way appends a note to the data and sends the data back to the client e*Way.

5 The client e*Way forwards the data to another file e*Way.

6 The data is stored in an output file on the local system.

The schema incorporates a number of different components, as shown in Figure 9.

Figure 9 Components in the Sample Schema



In the example, the e*Way named **TCPIP_Server** acts as an external TCP/IP server that receives messages from the e*Way named **TCPIP_Client**. If you have an existing TCP/IP server available and can configure it to bounce messages back across the same port that they are received, you do not need to configure the **TCPIP_Server** component in the schema.

4.2 Creating the Sample Schema

The first task in deploying the sample implementation is to create a new schema name. While it is possible to use the default schema for the sample implementation, it is recommended that you create a separate schema for testing purposes. After you install the e*Gate Integrator, do the following steps:

1. Start the e*Gate Enterprise Manager GUI.
2. When the Enterprise Manager prompts you to log in, select the host that you specified during installation, and enter your password.
3. You are then prompted to select a schema. Click on **New**.
4. Enter a name for the new schema; In this case, enter *TCPIP_Test*, or similar name as desired.

The e*Gate Enterprise Manager opens under your new schema. You now need to create the components required to use this sample schema. To do so, complete the following:

1. Identify the Event Type.
2. Define Collaboration Rules to process Event data.
3. Define the IQs to which Event data is published prior to sending it to the external system.
4. Define and configure each e*Way and associated Collaborations to apply the required Collaboration Rules.
5. Run the schema and make any necessary corrections.

The rest of this chapter describes each of these steps in detail.

Note: *For more information about creating or modifying any schema component within the e*Gate Enterprise Manager, see the Enterprise Manager's online Help system.*

4.2.1 Identify the Event Type

The sample schema uses a single Event Type, **GenericInEvent**. This Event Type transfers the data as a packet and uses an Event Type Definition with a single root node. **GenericInEvent** comes pre-packaged with e*Gate, you do not need to create it.

To identify GenericInEvent

1. In the Navigator, select the **Event Types** folder.
2. Verify that **GenericInEvent** is listed in the Editor and uses the Event Type Definition **GenericInEvent.ssc**.

4.2.2 Define Collaboration Rules

The next step is to define the Collaboration Rules the schema uses to process the data. The sample schema uses a single set of Collaboration Rules, called **PassThru**.

To create PassThru

1 In the Navigator, select the **Collaboration Rules** folder.

2 Create a new Collaboration Rules component called **PassThru**.

3 Edit the Properties of **PassThru** as follows:

Service	Pass Through
Subscription	**GenericInEvent** (the Event Type identified in Step 1)
Publication	**GenericInEvent** (the Event Type identified in Step 1)

4.2.3 Define IQs

The sample schema utilizes two IQs: One to store incoming data and another to store data after it has been processed by the TCP/IP server.

To define IQs

1 In the Navigator, select the **Participating Hosts** folder, then drill down to select the IQ Manager.

2 Create the following IQs:

- **Send_IQ**
- **Receive_IQ**

3 In the Properties for each IQ, verify that it uses the STC Standard IQ Service.

4 Display the Properties for the IQ Manager, and configure it to start automatically.

4.2.4 Define e*Ways and Collaborations

The sample schema utilizes four e*Way components: Two file-based e*Ways to read the incoming data from a file and send the results to an output file; one TCP/IP e*Way configured as a client; and one TCP/IP e*Way configured as a server.

Note: *The sample schema includes the TCP/IP e*Way configured as a server to simulate an external connection to a TCP/IP server. If you have an actual TCP/IP server available, you can use that as the external server and omit creating and configuring the* ***TCPIP_Server*** *e*way component in the schema.*

Important: *You must create and configure the e*Ways and Collaborations in the order listed below. Otherwise, publishers and subscribers may not be available to other components.*

TCPIP_Inbound

To create the TCPIP_Inbound e*Way

1. In the Navigator, select the **Participating Hosts** folder, then drill down to select the Control Broker.
2. Create an e*Way named **TCPIP_Inbound**.
3. Display the e*Way's Properties.
4. On the **General** tab, under **Executable file**, click **Find** to assign the file **stcewfile.exe**.
5. Select the **Start Up** tab. Configure the e*Way to start automatically.

To configure the e*Way's parameters

1. With the e*Way's Properties dialog box still displayed, select the **General** tab.
2. Under **Configuration file**, click **New**.

   This launches the e*Way Configuration Editor.
3. Configure the parameters in the e*Way Configuration Editor as follows:

Section	Parameter: Setting
General Settings	AllowIncoming: **Yes**
	AllowOutgoing: **No**
Poller (inbound) settings	PollDirectory: **C:\TEMP** (or another temporary directory)
	RemoveEOL: **No**

   Parameters not listed in the table should retain their default values.
4. Save the settings and promote the file to run time.
5. In the e*Way's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

To create the Collaboration for TCPIP_Inbound

1. In the Navigator, select the e*Way **TCPIP_Inbound**.
2. Create a Collaboration named **Inbound_Collab**.
3. Display the Collaboration's Properties and edit them as follows:

Collaboration Rules	**PassThru**
Subscriptions	Event Type: **GenericInEvent**
	Source: **<External>**
Publications	Event Type: **GenericInEvent**
	Destination: **Send_IQ**

4 In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

TCPIP_Client

To create the TCPIP_Client e*Way

1 In the Navigator, select the **Participating Hosts** folder, then select the Control Broker.

2 Create an e*Way named **TCPIP_Client**.

3 Display the e*Way's Properties.

4 On the **General** tab, under **Executable file**, click **Find** to assign the file **stcewgenericmonk.exe**.

5 Select the **Start Up** tab. Configure the e*Way to start automatically.

To configure the e*Way's parameters

1 With the e*Way's Properties dialog box still displayed, select the **General** tab.

2 Under **Configuration file**, click **New**.

3 In the **e*Way Template Selection** dialog box, select **stcewtcpipext**.

This launches the e*Way Configuration Editor.

4 Configure the parameters in the e*Way Configuration Editor as follows:

Section	Parameter: Setting
Communication Setup	Exchange Data Interval: **30**
TCPIP Configuration	Host: the name of the Participating Host (for example, **localhost**)

Parameters not listed in the table should retain their default values.

5 Save the settings and promote the file to run time.

6 In the e*Way's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

To create the first Collaboration for TCPIP_Client

1 In the Navigator, select the e*Way **TCPIP_Client**.

2 Create a Collaboration named **TCPIP_Send_Collab**.

3 Display the Collaboration's Properties and edit them as follows:

Collaboration Rules	**PassThru**
Subscriptions	Event Type: **GenericInEvent**
	Source: **Inbound_Collab**
Publications	Event Type: **GenericInEvent**
	Destination: **<External>**

4 In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

To create the second Collaboration for TCPIP_Client

1 In the Navigator, select the e*Way **TCPIP_Client**.

2 Create a Collaboration named **TCPIP_Receive_Collab**.

3 Display the Collaboration's Properties and edit them as follows:

Collaboration Rules	**PassThru**
Subscriptions	Event Type: **GenericInEvent**
	Source: **<External>**
Publications	Event Type: **GenericInEvent**
	Destination: **Receive_IQ**

4 In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

TCPIP_Server

To create the TCPIP_Server e*Way

1 In the Navigator, select the **Participating Hosts** folder, then select the Control Broker.

2 Create an e*Way named **TCPIP_Server**.

3 Display the e*Way's Properties.

4 On the **General** tab, under **Executable file**, click **Find** to assign the file **stcewgenericmonk.exe**.

5 Select the **Start Up** tab. Configure the e*Way to start automatically.

To configure the e*Way's parameters

1 With the e*Way's Properties dialog box still displayed, select the **General** tab.

2 Under **Configuration file**, click **New**.

3 In the **e*Way Template Selection** dialog box, select **stcewtcpipext**.

This launches the e*Way Configuration Editor.

4 Configure the parameters in the e*Way Configuration Editor as follows:

Section	Parameter: Setting
Communication Setup	Exchange Data Interval: **50**
Monk Configuration	Monk Environment Initialization File: **monk_library/ewtcpipext/tcpip-server-init.monk**
	Startup Function: **tcpip-server-startup**
	Process Outgoing Message Function: **tcpip-server-outgoing**
	Exchange Data With External Function: **tcpip-server-exchange**
	External Connection Establishment Function: **tcpip-server-extconnect**
	External Connection Verification Function: **tcpip-server-verify**
	External Connection Shutdown Function: **tcpip-server-shutdown**
	Positive Acknowledgment Function: **tcpip-server-ack**
	Negative Acknowledgment Function: **tcpip-server-nack**
TCPIP Server Configuration	Host: the name of the Participating Host (for example, **localhost**)

Parameters not listed in the table should retain their default values.

5 Save the settings and promote the file to run time.

6 In the e*Way's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

To create the Collaboration for TCPIP_Server

1 In the Navigator, select the e*Way **TCPIP_Server**.

2 Create a Collaboration named **TCPIP_Server_Collab**.

3 Display the Collaboration's Properties and edit them as follows:

Collaboration Rules	**PassThru**
Subscriptions	Event Type: **GenericInEvent**
	Source: **<External>**
Publications	Event Type: **GenericInEvent**
	Destination: **<External>**

4 In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

TCPIP_Outbound

To create the TCPIP_Outbound e*Way

1 In the Navigator, select the **Participating Hosts** folder, then select the Control Broker.

2 Create an e*Way named **TCPIP_Outbound**.

3 Display the e*Way's Properties.

4 On the **General** tab, under **Executable file**, click **Find** to assign the file **stcewfile.exe**.

5 Select the **Start Up** tab. Configure the e*Way to start automatically.

To configure the e*Way's parameters

1 With the e*Way's Properties dialog box still displayed, select the **General** tab.

2 Under **Configuration file**, click **New**.

This launches the e*Way Configuration Editor.

3 Configure the parameters in the e*Way Configuration Editor as follows:

Section	Parameter: Setting
General Settings	AllowIncoming: **No**
	AllowOutgoing: **Yes**
Outbound (send) settings	OutputDirectory: **C:\TEMP** (or another temporary directory)

Parameters not listed in the table should retain their default values.

4 Save the settings and promote the file to run time.

5 In the e*Way's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

To create the Collaboration for TCPIP_Outbound

1 In the Navigator, select the e*Way **TCPIP_Outbound**.

2 Create a Collaboration named **Outbound_Collab**.

3 Display the Collaboration's Properties and edit them as follows:

Collaboration Rules	**PassThru**
Subscriptions	Event Type: **GenericInEvent**
	Source: **TCPIP_Receive_Collab**
Publications	Event Type: **GenericInEvent**
	Destination: **<External>**

4 In the Collaboration's Properties dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

This operation completes schema configuration in the e*Gate Enterprise Manager.

4.2.5 Run the Schema

Before running the schema, create a sample input text file. Unless you specified something different in the e*Way Configuration Editor, make sure you save the file in the **C:\TEMP** directory with a **.fin** extension. To execute the *TCPIP_test* schema, do the following:

1 Go to the command prompt, and enter the following:

```
stccb -rh hostname -rs TCPIP_Test -un username -up user password
    -ln hostname_cb
```

Substitute *hostname*, *username* and user *password* as appropriate.

2 Exit from the command prompt, and start the e*Gate Monitor GUI.

3 When prompted, specify the hostname which contains the Control Broker you started in Step 1 above.

4 Select the *TCPIP_Test* schema.

5 After you verify that the Control Broker is connected (the message in the **Control** tab of the console indicates the command "succeeded" and the status as "up"), highlight the IQ Manager, *hostname*_igmgr, then click the right button of the mouse, and select **Start**.

6 Highlight each of the e*Ways, right click the mouse, and select **Start**.

Expected Results

e*Gate returns an output file (**output*n*.dat**, where *n* is a number) in the **C:\TEMP** directory (or whatever directory you specified) and changes the extension of the input file to **.~in**.

4.3 Importing the Sample Schema

Most e*Ways include sample schemas that you can use for testing purposes. These samples are automatically installed during the installation of e*Gate. The sample schema created from scratch in the previous sections is identical to the sample schema included with the e*Gate installation.

This section explains how to import the sample schema without having to create the individual components.

To import the schema

1 From the e*Gate Enterprise Manager, choose **New** on the **File** menu.

The **New Schema** dialog box appears.

2 Click **Create from export** and enter the schema name **TCPIP_Test**.

3 Click **Find** and use the **Import from File** dialog box to select the following schema export file:

samples\ewtcpipext\TcpipExt_Sample.zip

The dialog box closes.

4 From the **New Schema** dialog box, click **Open**.

The Enterprise Manager displays the imported schema.

When the e*Gate Enterprise Manager GUI opens, you can see that most of the components (including e*Ways, Event Types, Collaborations, Collaboration Rules and IQs) are already defined and configured. Only the TCPIP_Server e*Way needs to be configured.

To configure TCPIP_Server e*Way

1 Select the TCPIP_Server e*Way from the **Components** tab of the e*Gate Enterprise Manager.

2 Right click with your mouse and select **Properties**.

3 On the **General** tab of the **Properties** window, click on the **Edit** button under the **Configuration file** field.

4 When the **Settings** window opens, refer to the following table to configure the parameters for this configuration file:

Section	Parameter: Setting
Communication Setup	Exchange Data Interval: **50**
Monk Configuration	Monk Environment Initialization File: **monk_library/ewtcpipext/tcpip-server-init.monk**
	Startup Function: **tcpip-server-startup**
	Process Outgoing Message Function: **tcpip-server-outgoing**
	Exchange Data With External Function: **tcpip-server-exchange**
	External Connection Establishment Function: **tcpip-server-extconnect**

Section (Continued)	**Parameter: Setting**
Monk Configuration (cont.)	External Connection Verification Function: **tcpip-server-verify**
	External Connection Shutdown Function: **tcpip-server-shutdown**
	Positive Acknowledgment Function: **tcpip-server-ack**
	Negative Acknowledgment Function: **tcpip-server-nack**
TCPIP Server Configuration	Host: the name of the Participating Host (for example, **localhost**)

Parameters not listed in the table can retain their default values.

5 Save the settings and promote the file to run time.

6 In the e*Way's **Properties** dialog box, click **OK** to save all changes and return to the e*Gate Enterprise Manager's main window.

To run this schema, see **"Run the Schema" on page 46**.

Chapter 5

TCP/IP e*Way Functions

The TCP/IP e*Way functions fall into the following categories:

5.1 Basic Functions

The functions in this category control the e*Way's most basic operations.

Note: *The functions described in this section can only be used by the functions defined within the e*Way's configuration file. None of the functions are available to Collaboration Rules scripts executed by the e*Way.*

The basic functions are:

event-send-to-egate

Syntax

```
(event-send-to-egate string)
```

Description

event-send-to-egate sends data that the e*Way has already received from the external system into the e*Gate system as an Event.

Parameters

Name	Type	Description
string	string	The data to be sent to the e*Gate system.

Return Values

Boolean
Returns **#t** (true) if the data is sent successfully, otherwise, returns **#f** (false).

Throws

None.

Additional information

This function can be called by any e*Way function when it is necessary to send data to the e*Gate system in a blocking fashion.

get-logical-name

Syntax

```
(get-logical-name)
```

Description

get-logical-name returns the logical name of the e*Way.

Parameters

None.

Return Values

string
Returns the name of the e*Way (as defined by the Enterprise Manager).

Throws

None.

send-external-down

Syntax

```
(send-external-down)
```

Description

send-external down instructs the e*Way that the connection to the external system is down.

Parameters

None.

Return Values

None.

Throws

None.

send-external-up

Syntax

```
(send-external-up)
```

Description

send-external-up instructs the e*Way that the connection to the external system is up.

Parameters

None.

Return Values

None.

Throws

None.

shutdown-request

Syntax

```
(shutdown-request)
```

Description

shutdown-request completes the e*Gate shutdown procedure that was initiated by the Control Broker but was interrupted by returning a non-null value within the Shutdown Command Notification Function (see **"Shutdown Command Notification Function" on page 31**). Once this function is called, shutdown proceeds immediately.

Once interrupted, the e*Way's shutdown cannot proceed until this Monk function is called. If you do interrupt an e*Way shutdown, we recommend that you complete the process in a timely fashion.

Parameters

None.

Return Values

None.

Throws

None.

start-schedule

Syntax

```
(start-schedule)
```

Description

start-schedule requests that the e*Way execute the “Exchange Data with External” function specified within the e*Way’s configuration file. Does not affect any defined schedules.

Parameters

None.

Return Values

None.

Throws

None.

stop-schedule

Syntax

```
(stop-schedule)
```

Description

stop-schedule requests that the e*Way halt execution of the “Exchange Data with External” function specified within the e*Way’s configuration file. Execution is stopped when the e*Way concludes any open transaction. Does not affect any defined schedules, and does not halt the e*Way process itself.

Parameters

None.

Return Values

None.

Throws

None.

5.2 TCP/IP e*Way Standard Functions

The functions described in this section can only be used by the functions defined within the e*Way's configuration file. None of the functions are available to Collaboration Rules scripts executed by the e*Way.

The current suite of TCP/IP e*Way standard functions that control the e*Way's communications center are divided into two groups:

TCP/IP Client Configuration Functions

tcpip-ack on page 53	**tcpip-notify** on page 56
tcpip-exchange on page 54	**tcpip-outgoing** on page 57
tcpip-extconnect on page 54	**tcpip-shutdown** on page 64
tcpip-init on page 55	**tcpip-startup** on page 64
tcpip-nack on page 55	**tcpip-verify** on page 65

TCP/IP Server Configuration Functions

tcpip-server-ack on page 58	**tcpip-server-notify** on page 60
tcpip-server-exchange on page 58	**tcpip-server-outgoing** on page 61
tcpip-server-extconnect on page 59	**tcpip-server-shutdown** on page 62
tcpip-server-init on page 59	**tcpip-server-startup** on page 63
tcpip-server-nack on page 60	**tcpip-server-verify** on page 63

tcpip-ack

Syntax

```
(tcpip-ack message-string)
```

Description

tcpip-ack sends a positive acknowledgment to the external system after all Collaborations to which the e*Way sent data have processed and enqueued that data successfully.

Parameters

Name	Type	Description
message-string	string	The Event for which an acknowledgment is sent.

Return Values

string

An empty string indicates a successful operation. The e*Way is then able to proceed with the next request.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, the function is called again.

Additional Information

See **"Positive Acknowledgment Function" on page 30** for more information.

tcpip-exchange

Syntax

```
(tcpip-exchange)
```

Description

tcpip-exchange sends a received Event from the external system to e*Gate. The function expects no input.

Parameters

None.

Return Values

string

An empty string indicates a successful operation. Nothing is sent to e*Gate.

A string, containing Event data, indicates successful operation, and the returned Event is sent to e*Gate.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, this function is re-executed with the same input Event.

Throws

None.

Additional Information

See **"Exchange Data with External Function" on page 27** for more information.

tcpip-extconnect

Syntax

```
(tcpip-extconnect)
```

Description

tcpip-extconnect establishes a connection to the external system.

Parameters

None.

Return Values

string
UP indicates the connection is established. Anything else indicates no connection.

Throws

None.

Additional Information

See **“External Connection Establishment Function” on page 28** for more information.

tcpip-init

Syntax

```
(tcpip-init)
```

Description

tcpip-init begins the initialization process for the e*Way. This function loads the **stc_monktcpip.dll** file and the initialization file, thereby making the function scripts available for future use.

Parameters

None.

Return Values

string
If a FAILURE string is returned, the e*Way is shut down. Any other return indicates success.

Throws

None.

Additional Information

Within this function, any necessary global variables to be used by the function scripts could be defined. The internal function that loads this file is called once when the e*Way first starts up.

See **“Monk Environment Initialization File” on page 26** for more information.

tcpip-nack

Syntax

```
(tcpip-nack message-string)
```

Description

tcpip-nack sends a negative acknowledgment to the external system when the e*Way fails to process and queue Events from the external system.

Parameters

Name	Type	Description
message-string	string	The Event for which a negative acknowledgment is sent.

Return Values

string

An empty string indicates a successful operation. The e*Way is then able to proceed with the next request.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, the function is called again.

Throws

None.

Additional Information

See **"Negative Acknowledgment Function" on page 30** for more information.

tcpip-notify

Syntax

```
(tcpip-notify command)
```

Description

tcpip-notify notifies the external system that the e*Way is shutting down.

Parameters

Name	Type	Description
command	string	When the e*Way calls this function, it passes the string "SHUTDOWN_NOTIFICATION" as the parameter.

Return Values

string

If a FAILURE string is returned, the e*Way is shut down. Any other return indicates success.

Throws

None.

Additional Information

See **“Shutdown Command Notification Function” on page 31** for more information.

tcpip-outgoing

Syntax

```
(tcpip-outgoing event-string)
```

Description

tcpip-outgoing is used for sending a received message from e*Gate to the external system.

Parameters

Name	Type	Description
event-string	string	The Event to be processed.

Return Values

string

An empty string indicates a successful operation.

RESEND causes the Event to be immediately resent.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established this function is re-executed with the same input Event.

DATAERR indicates that there is a problem with the message (Event) data itself. First, the e*Way pauses the number of seconds specified by the **Resend Timeout** parameter. Then, the e*Way increments its “failed message (Event)” counter and rolls back the message (Event) to the IQ from which it was obtained. If the e*Way’s journal is enabled the message (Event) is journaled.

See **event-send-to-egate** on page 49 for more information.

Throws

None.

Additional Information

See **Process Outgoing Message Function** on page 27 for more information.

tcpip-server-ack

Syntax

```
(tcpip-server-ack message-string)
```

Description

tcpip-server-ack is used to send a positive acknowledgment to the external system, and for post processing after successfully sending data to e*Gate.

Parameters

Name	Type	Description
message-string	string	The Event for which an acknowledgment is sent.

Return Values

string

An empty string indicates a successful operation. The e*Way is then able to proceed with the next request.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, the function is called again.

Additional Information

See **"Positive Acknowledgment Function" on page 30** for more information.

tcpip-server-exchange

Syntax

```
(tcpip-server-exchange)
```

Description

tcpip-server-exchange is used for sending a received Event from the external system to e*Gate. The function expects no input.

Parameters

None.

Return Values

string

An empty string indicates a successful operation. Nothing is sent to e*Gate.

A string, containing Event data, indicates successful operation, and the returned Event is sent to e*Gate.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, this function is re-executed with the same input Event.

Throws

None.

Additional Information

See **"Exchange Data with External Function" on page 27** for more information.

tcpip-server-extconnect

Syntax

```
(tcpip-server-extconnect)
```

Description

tcpip-server-extconnect is used to establish external system connection.

Parameters

None.

Return Values

string
UP indicates the connection is established. Anything else indicates no connection.

Throws

None.

Additional Information

See **"External Connection Establishment Function" on page 28** for more information.

tcpip-server-init

Syntax

```
(tcpip-server-init)
```

Description

tcpip-server-init begins the initialization process for the e*Way. This function loads the **stc_monktcpip.dll** file.

Parameters

None.

Return Values

string
If a FAILURE string is returned, the e*Way is shut down. Any other return indicates success.

Throws

None.

Additional Information

Within this function, any necessary global variables to be used by the function scripts could be defined. The internal function that loads this file is called once when the e*Way first starts up.

See **"Monk Environment Initialization File" on page 26** for more information.

tcpip-server-nack

Syntax

```
(tcpip-server-nack message-string)
```

Description

tcpip-server-nack is used to send a negative acknowledgment to the external system, and for post processing after failing to send data to e*Gate.

Parameters

Name	Type	Description
message-string	string	The Event for which a negative acknowledgment is sent.

Return Values

string
An empty string indicates a successful operation. The e*Way is then able to proceed with the next request.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, the function is called again.

Throws

None.

Additional Information

See **"Negative Acknowledgment Function" on page 30** for more information.

tcpip-server-notify

Syntax

```
(tcpip-server-notify command)
```

Description

tcpip-server-notify notifies the external system that the e*Way is shutting down.

Parameters

Name	Type	Description
command	string	When the e*Way calls this function, it passes the string “SUSPEND_NOTIFICATION” as the parameter.

Return Values

string
If a FAILURE string is returned, the e*Way is shut down. Any other return indicates success.

Throws

None.

Additional Information

See **“Shutdown Command Notification Function” on page 31** for more information.

tcpip-server-outgoing

Syntax

```
(tcpip-server-outgoing event-string)
```

Description

tcpip-server-outgoing is used for sending a received Event from e*Gate to the external system.

Parameters

Name	Type	Description
event-string	string	The Event to be processed.

Return Values

string
An empty string indicates a successful operation.

RESEND causes the Event to be immediately resent.

CONNERR indicates a problem with the connection to the external system. When the connection is re-established, this function is re-executed with the same input Event.

DATAERR indicates that there is a problem with the message (Event) data itself. First, the e*Way pauses the number of seconds specified by the **Resend Timeout** parameter. Then, the e*Way increments its "failed message (Event)" counter and rolls back the message (Event) to the IQ from which it was obtained. If the e*Way's journal is enabled the message (Event) is journaled.

See **event-send-to-egate** on page 49 for more information.

Throws

None.

Additional Information

See **"Process Outgoing Message Function" on page 27** for more information.

tcpip-server-shutdown

Syntax

```
(tcpip-server-shutdown shutdown)
```

Description

tcpip-server-shutdown is called by the system to request that the external shut down. A return value of SUCCESS indicates that the shutdown can occur immediately. Any other return value indicates that the shutdown Event must be delayed. The user is then required to execute a (shutdown-request) call from within a Monk function to allow the requested shutdown process to continue.

Parameters

Name	Type	Description
shutdown	string	When the e*Way calls this function, it passes the string "SHUTDOWN_NOTIFICATION" as the parameter.

Return Values

string
SUCCESS allows an immediate shutdown to occur. Anything else delays shutdown until the **shutdown-request** is executed successfully.

Throws

None.

Additional Information

See **"External Connection Shutdown Function" on page 29** for more information.

tcpip-server-startup

Syntax

```
(tcpip-server-startup)
```

Description

tcpip-server-startup is used for instance specific function loads and invokes setup.

Parameters

None.

Return Values

string
FAILURE causes shutdown of the e*Way. Any other return indicates success.

Throws

None.

Additional Information

This function should be used to initialize the external system before data exchange starts. Any additional variables may be defined here.

See **“Startup Function” on page 26** for more information.

tcpip-server-verify

Syntax

```
(tcpip-server-verify)
```

Description

tcpip-server-verify is used to verify whether the external system connection is established.

Parameters

None.

Return Values

string
UP if connection established. Any other value indicates the connection is not established.

Throws

None.

Additional Information

See **“External Connection Verification Function” on page 29** for more information.

tcpip-shutdown

Syntax

```
(tcpip-shutdown shutdown)
```

Description

tcpip-shutdown is called by the system to request that the external shut down. A return value of **SUCCESS** indicates that the shutdown can occur immediately. Any other return value indicates that the shutdown Event must be delayed. The user is then required to execute a (shutdown-request) call from within a Monk function to allow the requested shutdown to process to continue.

Parameters

Name	Type	Description
shutdown	string	When the e*Way calls this function, it passes the string “SUSPEND_NOTIFICATION” as the parameter.

Return Values

string
SUCCESS allows an immediate shutdown to occur. Anything else delays shutdown until the **shutdown-request** is executed successfully.

Throws

None.

Additional Information

See **“External Connection Shutdown Function” on page 29** for more information.

tcpip-startup

Syntax

```
(tcpip-startup)
```

Description

tcpip-startup is used for instance specific function loads and invokes setup.

Parameters

None.

Return Values

string
FAILURE causes shutdown of the e*Way. Any other return indicates success.

Throws

None.

Additional Information

This function should be used to initialize the external system before data exchange starts. Any additional variables may be defined here.

See **“Startup Function” on page 26** for more information.

tcpip-verify

Syntax

```
(tcpip-verify)
```

Description

tcpip-verify is used to verify whether the external system connection is established.

Parameters

None.

Return Values

string
UP if connection established. Any other value indicates the connection is not established.

Throws

None.

Additional Information

See **“External Connection Verification Function” on page 29** for more information.

5.3 TCP/IP e*Way Client Functions

The current suite of TCP/IP client functions that facilitate connection to the external client system are:

tcpip-close on page 66

tcpip-connect on page 66

tcpip-isconnected on page 67

tcpip-recv on page 68

tcpip-send on page 68

tcpip-waiting on page 69

tcpip-close

Syntax

```
(tcpip-close hCon)
```

Description

tcpip-close de-allocates the TCP/IP session handle obtained by **tcpip-connect**.

Parameters

Name	Type	Description
hCon	opaque handle	The handle associated with the TCP/IP session.

Return Values

Boolean
Returns **#t** (true) if the data is sent successfully; otherwise, returns **#f** (false).

Throws

None.

Example

```
(if (tcpip-close hCon)
    (display “Connection closed successfully.”)
    (display “Connection close failed.”)
)
```

tcpip-connect

Syntax

```
(tcpip-connect pszHostName dwPort cPacketSize fNoDelay)
```

Description

tcpip-connect locates and establishes external system connection.

Parameters

Name	Type	Description
pszHostName	string	A zero-delimited string specifying the hostname for connection.
dwPort	integer	The port number for the connection.

Name	Type	Description
cPacketSize	integer	The packet size.
fNoDelay	Boolean (#t, #f)	A flag specifying that the e*Way is awaiting connection to the external system.

Return Values

handle
The handle associated with a TCP/IP session.

Throws

None.

Examples

```
(define hCon (tcpip-connect “myhost” 8888 4096 #f))
```

tcpip-isconnected

Syntax

```
(tcpip-isconnected hCon)
```

Description

tcpip-isconnected verifies whether the external system connection is established.

Parameters

Name	Type	Description
hCon	opaque handle	The handle associated with the TCP/IP session.

Return Values

Boolean
Returns **#t** (true) if the data is sent successfully; otherwise, returns **#f** (false).

Throws

None.

Examples

```
(if (tcpip-isconnected hCon)
    (display “Connection is opened.”)
    (display “Connection is closed.”)
)
```

tcpip-recv

Syntax

```
(tcpip-recv hCon cbMax [cmsTimeout])
```

Description

tcpip-recv retrieves the message from the read buffer.

Parameters

Name	Type	Description
hCon	opaque handle	The handle associated with the TCP/IP session.
cbMax	integer	The maximum number of bytes in the expected string.
cmsTimeout	integer	The amount of time between attempts in milliseconds. The default is 2000. This parameter is optional.

Return Values

string
Returns the string retrieved from the buffer.

Throws

None.

Examples

```
(define retStr (tcpip-recv hCon 20))
(display retStr)
```

tcpip-send

Syntax

```
(tcpip-send hCon pszMessage)
```

Description

tcpip-send verifies whether the message was sent successfully.

Parameters

Name	Type	Description
hCon	opaque handle	The handle associated with the TCP/IP session.
pszMessage	string	The string to send.

Return Values

Boolean
Returns **#t** (true) if the data is sent successfully, or returns **#f** (false); can also return MONK_UNSPECIFIED.

Throws

None.

Example

```
(if (tcpip-send hCon “Message string”)
    (display “Message sent successfully.”)
    (display “Message being sent failed.”)
)
```

tcpip-waiting

Syntax

```
(tcpip-waiting hCon [cmsTimeout=2000])
```

Description

tcpip-waiting returns the number of bytes ready to be read in the buffer.

Parameters

Name	Type	Description
hCon	opaque handle	The handle associated with the TCP/IP session.
cmsTimeout	An Integer	An optional parameter, the default is set at 2000, specifying the amount of time between attempts in milliseconds.

Return Values

integer
Returns the number of bytes ready to be read in the read buffer.

Throws

None.

Examples

```
(define retStr “”)
(define cbWaiting (tcpip-waiting hCon 2000))

(if (> cbWaiting 0)
    (set! retStr (tcpip-recv hCon cbWaiting 2000))
    (display “No Data\n”)
)
(display retStr)
```

5.4 TCP/IP Server Functions

The current suite of TCP/IP server functions that facilitate connection to the TCP/IP Server system are:

tcpip-server-client-count on page 70

tcpip-server-client-isconnected on page 71

tcpip-server-clients-waiting on page 71

tcpip-server-close on page 72

tcpip-server-close-client on page 72

tcpip-server-connect on page 73

tcpip-server-end-service-client on page 74

tcpip-server-isconnected on page 75

tcpip-server-recv on page 75

tcpip-server-send on page 76

tcpip-server-service-next-client on page 76

tcpip-server-waiting on page 77

tcpip-server-client-count

Syntax

```
(tcpip-server-client-count hServer)
```

Description

tcpip-server-client-count obtains the number of open clients on the server.

Parameters

Name	Type	Description
hServer	opaque handle	The handle to the server, returned by **tcpip-server-connect**.

Return Values

integer
Returns the number of open clients.

Throws

None.

tcpip-server-client-isconnected

Syntax

```
(tcpip-server-client-isconnected hClient)
```

Description

tcpip-server-client-isconnected checks whether the currently serviced client's connection is opened or its receive buffer has data.

Parameters

Name	Type	Description
hClient	opaque handle	The handle to the client, returned by **tcpip-service-next-client**.

Return Values

Boolean
Returns **#t** (true) to indicate that the client connection is opened or the client's receive buffer has data; otherwise, returns **#f** (false).

Additional Information

Call this API to ensure that the client connection can be closed via **tcpip-server-end-service-client**, since **tcip-server-client-isconnected** does not close the client's connection.

tcpip-server-clients-waiting

Syntax

```
(tcpip-server-clients-waiting hServer)
```

Description

tcpip-server-clients-waiting checks whether the server has any client connections.

Parameters

Name	Type	Description
hServer	opaque handle	The handle to the server, returned by **tcpip-server-connect**.

Return Values

Boolean
Returns **#t** (true) to indicate that there are open connections to clients; otherwise, returns **#f** (false).

Throws

None.

tcpip-server-close

Syntax

```
(tcpip-server-close hServer)
```

Description

tcpip-server-close shuts down the server and closes the connection to the host.

Parameters

Name	Type	Description
hServer	opaque handle	The handle to the server, returned by **tcpip-server-connect**.

Return Values

Boolean
Returns **#t** (true) to indicate that the connection closed successfully; otherwise, returns **#f** (false).

tcpip-server-close-client

Syntax

```
(tcpip-server-close-client hServer hClient)
```

Description

tcpip-server-close-client closes the client connection.

Parameters

Name	Type	Description
hServer	opaque handle	The handle to the server, returned by **tcpip-server-connect**.
hClient	opaque handle	The handle to the client, returned by **tcpip-service-next-client**.

Return Values

Boolean

Returns **#t** (true) to indicate that the connection closed without any errors; otherwise, returns **#f** (false).

Additional Information

To insure that the client connection is not closed abruptly, and the data in the client's receive buffer does not get lost, call **tcpip-server-client-isconnected** before calling this API.

tcpip-server-connect

Syntax

```
(tcpip-server-connect pszHostName dwPort cPacketSize cMaxConnections
cmsWaitForClientTimeout cmsProcessWait fNoDelay)
```

Description

tcpip-server-connect establishes a connection to the host and starts the server.

Parameters

Name	Type	Description
pszHostName	string	A zero delimited string specifying the host name to which to connect.
dwPort	integer	The port number on which the server listens for client connections.
cPacketSize	integer	The number specifying the size of each packet.
cMaxConnections	integer	The number specifying the maximum number of clients that can connect.

Name	Type	Description
cmsWaitForClientTimeout	integer	The integer specifying the timeout in milliseconds, for a client connection.
cmsProcessWait	integer	The integer specifying the time, in milliseconds, between checks for client connections.
fNoDelay	Boolean (#t, #f)	A flag specifying that the e*Way is awaiting connection to the external system.

Return Values

handle
Returns the handle to the server.

Throws

None.

tcpip-server-end-service-client

Syntax

```
(tcpip-server-end-service-client hClient)
```

Description

tcpip-server-end-service-client returns the currently serviced client back to the waiting queue.

Parameters

Name	Type	Description
hClient	opaque handle	The handle to the client, returned by **tcpip-service-next-client**.

Return Values

Boolean
Returns **#t** (true) if successful; otherwise, returns **#f** (false).

Additional Information

This API **does not** close the client connection. This API manages the client handle for memory management. Call this API before making subsequent calls to **tcpip-server-service-next-client**.

tcpip-server-isconnected

Syntax

```
(tcpip-server-isconnected hServer)
```

Description

tcpip-server-isconnected checks whether the server is up and the connection opened.

Parameters

Name	Type	Description
hServer	opaque handle	The handle to the server, returned by **tcpip-server-connect**.

Return Values

Boolean
Returns **#t** (true) to indicate that the connection is open; otherwise, returns **#f** (false).

tcpip-server-recv

Syntax

```
(tcpip-server-recv hCon cbMax [cmsTimeout=2000])
```

Description

tcpip-server-recv retrieves the message from the read buffer. All parameters are mandatory.

Parameters

Name	Type	Description
hCon	Opaque handle	The handle associated with the TCP/IP session.
cbMax	Integer	The maximum number of bytes in the expected string.
cmsTimeout	Integer	The amount of time between attempts, in milliseconds. The default is 2000.

Return Values

string
Returns the string retrieved from the buffer.

Throws

None.

Examples

```
(define retStr (tcpip-server-recv hCon 20))
(display retStr)
```

tcpip-server-send

Syntax

(tcpip-server-send *hClient pszMessage*)

Description

tcpip-server-send sends a message to the client.

Parameters

Name	Type	Description
hClient	opaque handle	The handle to the client, returned by **tcpip-service-next-client**.
pszMessage	string	The message to send to the client.

Return Values

Boolean
Returns **#t** (true) if successful; otherwise, returns **#f** (false).

tcpip-server-service-next-client

Syntax

(tcpip-server-service-next-client *hServer*)

Description

tcpip-server-service-next-client retrieves the next client in the queue for servicing the client's request.

Parameters

Name	Type	Description
hServer	opaque handle	The handle to the server, returned by **tcpip-server-connect**.

Return Values

handle

Returns the handle to the next client waiting to be serviced.

tcpip-server-waiting

Syntax

```
(tcpip-server-waiting hClient cmsTimeout)
```

Description

tcpip-server-waiting retrieves the number of bytes of data waiting in the client's receive buffer.

Parameters

Name	Type	Description
hClient	opaque handle	The handle to the client, returned by **tcpip-service-next-client**.
cmsTimeout	integer	An integer specifying the amount of milliseconds between attempts to retrieve data.

Return Values

integer

Returns the number of bytes waiting in the client's receive buffer.

Additional Information

If **cbMax** is greater than or equal to the number of bytes in the client's receive buffer, all data in the buffer is returned, otherwise, the **cbMax** number of bytes from the buffer is returned.

Index