SeeBeyond ICAN Suite

WAP e*Way Intelligent Adapter User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Version 20050405202654.

Contents

Chapter 1

Introduction

This document describes how to install and configure the WAP e*Way Intelligent Adapter.

1.1 Overview

The WAP e*Way Intelligent Adapter enables the e*Gate system to exchange data with mobile devices that are WAP 1.1 enabled via Phone.com's UP.browser.

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 be thoroughly familiar with ***WAP 1.1 protocol***; Phone.com UP.browser; and to be thoroughly familiar with Windows-style GUI operations.

1.1.2 Components

The WAP e*Way comprises the following:

- **stcewgenericmonk.exe**, the executable component
- Configuration files, which the e*Way Editor uses to define configuration parameters
- Monk function scripts, discussed in **Chapter 4**.
- Library files.

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

1.2 Supported Operating Systems

The WAP e*Way is supported on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003

1.3 System Requirements

To use the WAP e*Way, you need the following:

- An e*Gate Participating Host.
- A TCP/IP network connection.
- Java 2 SDK Version 4.1.
- 8 MB free disk space on both the Participating and Registry Hosts, for e*Way executable, configuration, library, and script files.

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

- Access to a Web Server to host the data to be requested and retrieved by the mobile device.

1.3.1 External System Requirements

To use the WAP e*Way, you need the following:

- Active registration/subscription to a WAP 1.1 enabled via Phone.com's UP.browser.
- ODBC driver.

Chapter 2

Installation

This chapter describes how to install the WAP e*Way.

2.1 Windows

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 WAP e*Way on a Windows system:

1. Log in as an Administrator on the workstation on which you want to install the e*Way.
2. Insert the e*Way installation CD-ROM into the CD-ROM drive.
3. If the CD-ROM drive's "Autorun" feature is enabled, the setup application should launch automatically; skip ahead to step 4. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.
4. The InstallShield setup application will launch. Follow the on-screen instructions to install the e*Way.

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

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.

2.2 Files/Directories Created by the Installation

The WAP e*Way installation process will install the following files within the e*Gate directory tree. Files will be installed within the "egate\client" tree on the Participating Host and committed to the "default" schema on the Registry Host.

Table 1 Files created by the installation

e*Gate Directory	File(s)
bin\	stcewgenericmonk.exe
server\bin\	stc_monkwap.dll
configs\stcewgenericmonk\	stcewwap.def
monk_library\	ewwap.gui
monk_library\ewwap\	wap-ack.monk wap-exchange.monk wap-connect.monk wap-init.monk wap-nack.monk wap-notifiy.monk wap-outgoing.monk wap-shutdown.monk wap-startup.monk wap-verify.monk

Chapter 3

Configuration

This chapter describes how to configure the WAP e*Way.

3.1 e*Way Configuration Parameters

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

To change e*Way configuration parameters:

1. In the Schema Designer's Component editor, select the e*Way you want to configure and display its properties.
2. Under **Configuration File**, do one of three things:
   - Click **New** to create a new file. Then, from the **e*Way Template Selection** list, select **WAP e*Way** and click **OK**.
   - Click **Find** to select an existing configuration file.
   - Click **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.

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

- General Settings
- Communication Setup
- Monk 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 filename, optionally including an absolute path (for example, **c:\temp\filename.txt**). If an absolute path is not specified, the file will be 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 will be journaled for the following conditions:

- When the number of resends is exceeded (see Max Resends Per Message below)
- When its receipt is due to an external error, but Forward External Errors is set to **No**. (See **"Forward External Errors" on page 10** for more information.)

Max Resends Per Message

Description

Specifies the number of times the e*Way will attempt to resend a message (Event) to the external system after receiving an error. When this maximum is reached, the message is considered "Failed" and is written to the journal file.

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 will allow. When the specified number of failed messages is reached, the e*Way will shut down and exit.

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 will be queued to the e*Way's configured queue. See **"Exchange Data with External Function" on page 24** for more information.

Required Values

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

See **"Schedule-driven Data Exchange Functions" on page 18** 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 Schema Designer controls when the e*Way executable will run. The schedule you set within the parameters discussed in this section (using the e*Way Editor) determines when data will be 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 will terminate 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 will be 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 24**, **"Exchange Data Interval" on page 12**, and **"Stop Exchange Data Schedule" on page 11** for more information.

Stop Exchange Data Schedule

Description

Establishes the schedule to stop data exchange.

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).

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 between 0 and 86,400. The default is 120.

Additional Information

If **Zero Wait Between Successful Exchanges** is set to **Yes** and the **Exchange Data with External Function** returns data, The **Exchange Data Interval** setting will be ignored and the e*Way will invoke the **Exchange Data with External Function** immediately.

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

See **"Down Timeout" on page 12** and **"Stop Exchange Data Schedule" on page 11** for more information about the data-exchange schedule.

Down Timeout

Description

Specifies the number of seconds that the e*Way will wait between calls to the **External Connection Establishment** function. See **"External Connection Establishment Function" on page 25** 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 will wait between calls to the **External Connection Verification** function. See **"External Connection Verification Function" on page 25** 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 will wait 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 will immediately invoke the **Exchange Data with External** function if the previous exchange function returned data. If this parameter is set to **No**, the e*Way will always wait the number of seconds specified by **Exchange Data Interval** between invocations of the **Exchange Data with External** function. The default is **No**.

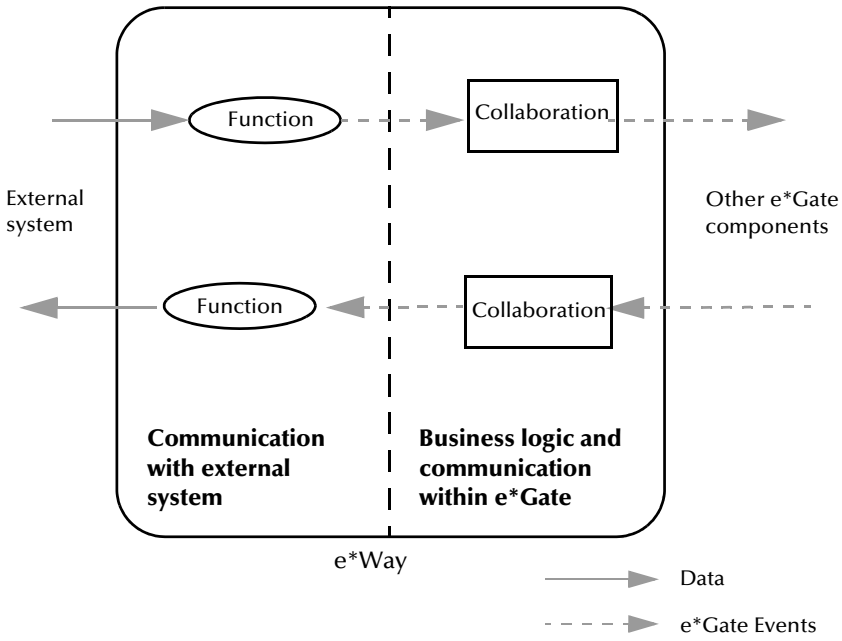
See **"Exchange Data with External Function" on page 24** for more information.

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 below) 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:

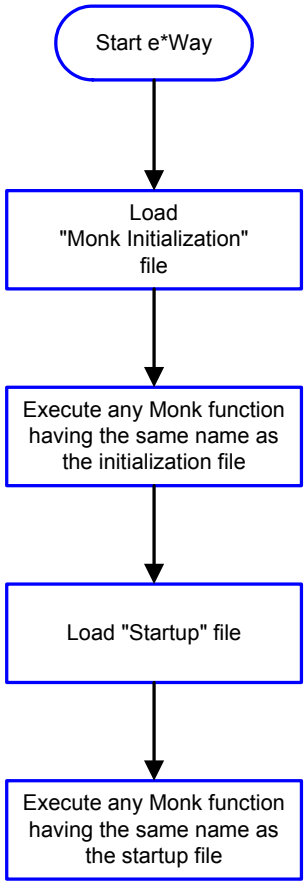
Type of Operation	Name
Initialization	**Startup Function** on page 23 (also see **Monk Environment Initialization File** on page 22)
Connection	**External Connection Establishment Function** on page 25 **External Connection Verification Function** on page 25 **External Connection Shutdown Function** on page 26
Schedule-driven data exchange	**Exchange Data with External Function** on page 24 **Positive Acknowledgment Function** on page 26 **Negative Acknowledgment Function** on page 27
Shutdown	**Shutdown Command Notification Function** on page 27
Event-driven data exchange	**Process Outgoing Message Function** on page 23

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

Initialization Functions

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

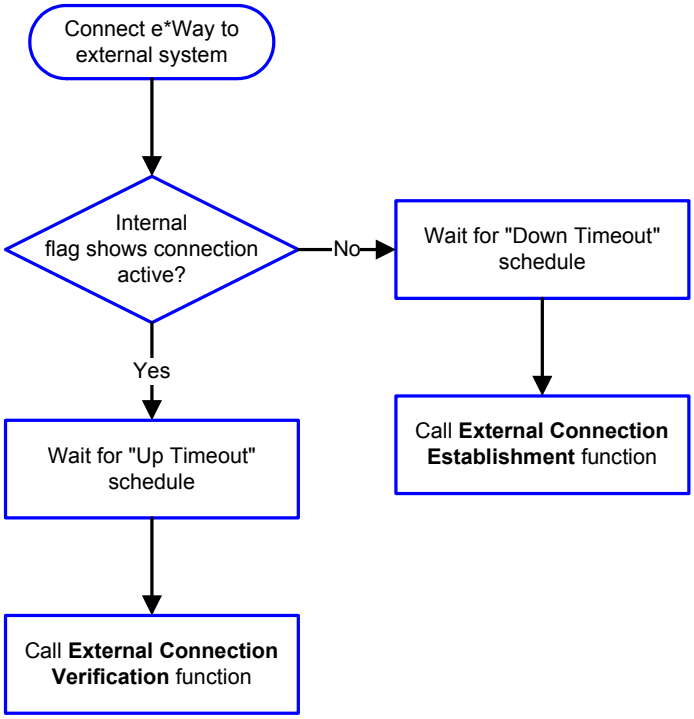
Figure 2 Initialization Functions



Connection Functions

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

Figure 3 Connection establishment and verification functions

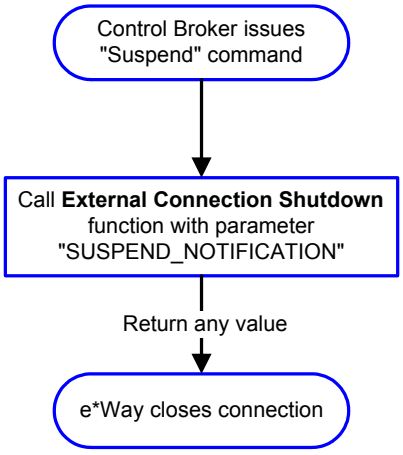


Note: *The e*Way selects the connection function based on an internal "up/down" flag rather than a poll to the external system. See* **Figure 5 on page 19** *and* **Figure 7 on page 21** *for examples of how different functions use this flag.*

User functions can manually set this flag using Monk functions. See **send-external-up** *on page 31 and* **send-external-down** *on page 30 for more information.*

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

Figure 4 Connection shutdown function



Schedule-driven Data Exchange Functions

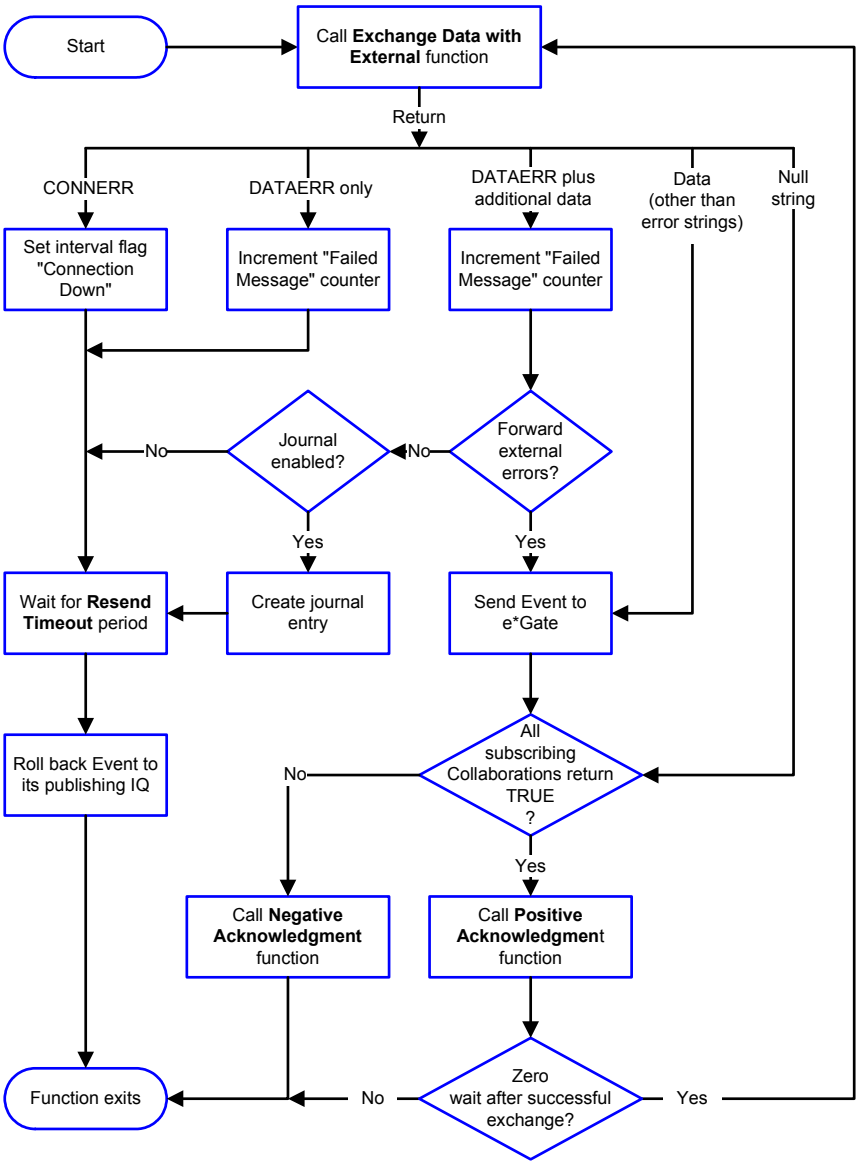
Figure 5 (on the next page) illustrates how the e*Way performs schedule-driven data exchange using the **Exchange Data with External Function**. The **Positive Acknowledgment Function** and **Negative Acknowledgment Function** are also called during this process.

"Start" can occur in any of the following ways:

- The "Start Data Exchange" time occurs
- Periodically during the 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.

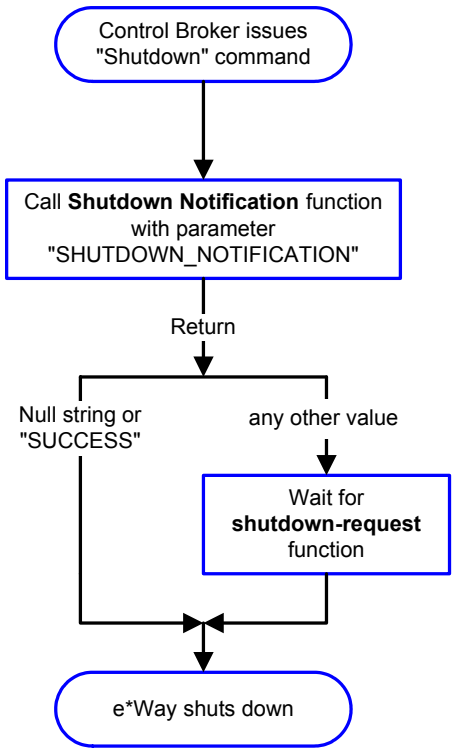
Figure 5 Schedule-driven data exchange functions



Shutdown Functions

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

Figure 6 Shutdown functions



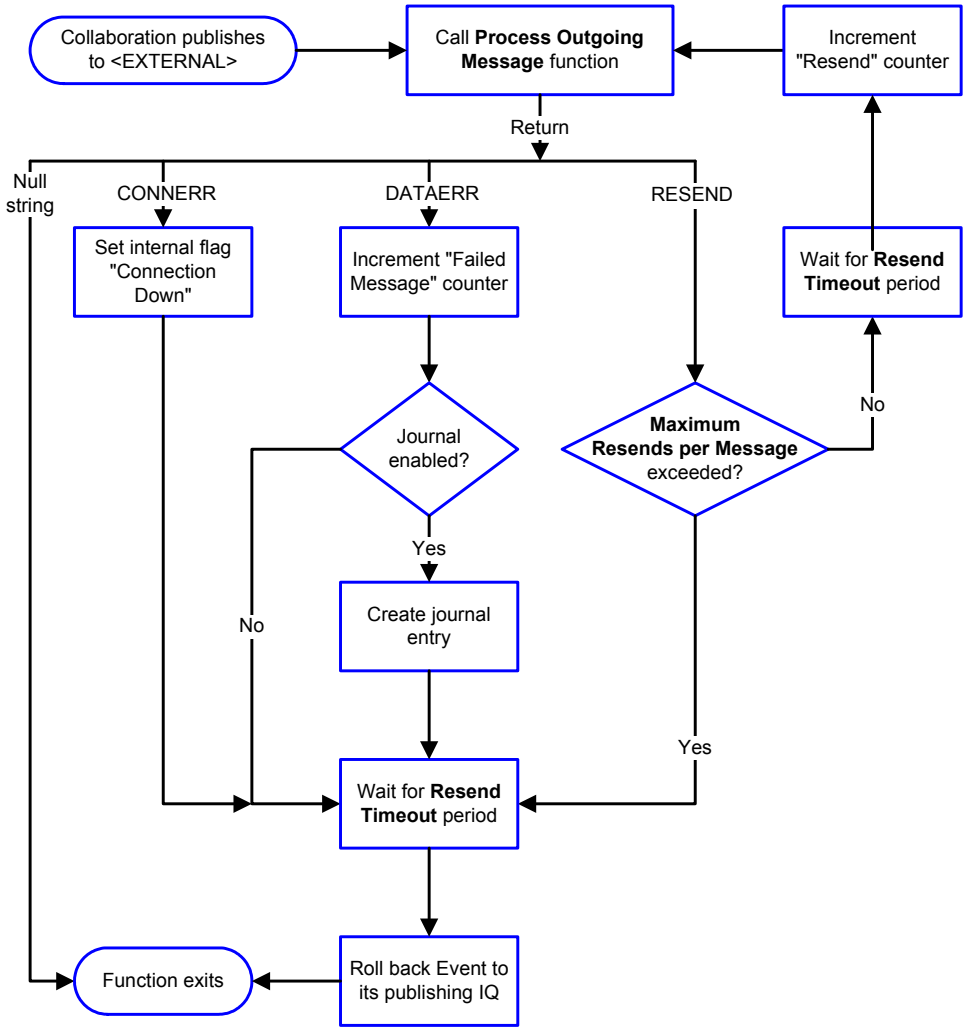
Event-driven Data Exchange Functions

Figure 7 on the next page 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



How to Specify Function Names or File Names

Parameters that require the name of a Monk function will 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

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 will be 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 will automatically be 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. This parameter is optional and may be left blank. The default is **monk_library/ewwap**.

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.

Monk Environment Initialization File

Specifies a file that contains environment initialization functions, which will be 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 will be appended to the "load path." See **"Additional Path" on page 22** for more information about the "load path."The default is **wap-init.monk**. See **wap-init** on page 35 for more information.

Additional information

Any environment-initialization functions called by this file accept no input, and must return a string. The e*Way will load this file and try 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 16**).

Startup Function

Description

Specifies a Monk function that the e*Way will load 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 **wap-startup**. See **wap-startup** on page 38 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 will be called after the e*Way loads the specified "Monk Environment Initialization file" and any files within the specified **Auxiliary Directories**.

The e*Way will load this file and try to invoke a function of the same base name as the file name (see **Figure 2 on page 16**). 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 **wap-outgoing**. See **wap-outgoing** on page 37 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 Schema Designer). The function returns one of the following (see **Figure 7 on page 21** 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 than the above is returned, the e*Way will create 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 29 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 **wap-exchange**. See **wap-exchange** on page 34 for more information.

Additional Information

The function accepts no input and must return a string (see **Figure 5 on page 19** for more details):

- Null string: Indicates that the data exchange was completed successfully. No information will be 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 19** 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 will immediately call the **Exchange Data with External** function again; otherwise, the e*Way will 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 32 for more information).

External Connection Establishment Function

Description

Specifies a Monk function that the e*Way will call 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 **wap-connect**. See **wap-connect** on page 34 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 will call 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 will execute the **External Connection**

Establishment function in its place. The default is **wap-verify**. See **wap-verify** on page 39 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 will call to shut down the connection to the external system.

Required Values

The name of a Monk function. This parameter is optional. The default is **wap-shutdown**. See **wap-shutdown** on page 37 for more information.

Additional Information

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

This function will only be invoked when the e*Way receives a "suspend" command from a Control Broker. When the "suspend" command is received, the e*Way will invoke 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 will call 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 **wap-ack**. See **wap-ack** on page 33 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 will be 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 will call 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 **wap-nack**. See **wap-nack** on page 35 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 will be 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 will be 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 **wap-notify** on page 36 for more information.

Additional Information

When the Control Broker issues a shutdown command to the e*Way, the e*Way will call 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 will not proceed until the Monk function **shutdown-request** is executed (see **shutdown-request** on page 31).

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.*

Chapter 4

WAP e*Way Functions

The WAP e*Way's functions fall into the following categories:

4.1 Basic Functions

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

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

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 Schema Designer).

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 27**). 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 will be 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.

4.2 WAP Standard Functions

The functions in this section control the e*Way's communications center and are defined within the configuration file.

The current suite of Standard functions are:

wap-ack

Syntax

```
(wap-ack arg)
```

Description

wap-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
arg	string	The Event for which an acknowledgment is sent.

Return Values

string

An empty string indicates a successful operation. The e*Way will then be 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 will be called again.

Additional Information

See **"Positive Acknowledgment Function" on page 26** for more information.

wap-exchange

Syntax

```
(wap-exchange)
```

Description

wap-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 will be reexecuted with the same input Event.

Throws

None.

Additional Information

See **"Exchange Data with External Function" on page 24** for more information.

wap-connect

Syntax

```
(wap-connect)
```

Description

wap-connect 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 25** for more information.

wap-init

Syntax

```
(wap-init)
```

Description

wap-init begins the initialization process for the e*Way. This function loads the **stc_monkwap.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 will shutdown. 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 22** for more information.

wap-nack

Syntax

```
(wap-nack arg)
```

Description

wap-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
arg	string	The Event for which a negative acknowledgment is sent.

Return Values

string
An empty string indicates a successful operation.

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

Throws

None.

Additional Information

See **"Negative Acknowledgment Function" on page 27** for more information.

wap-notify

Syntax

```
(wap-notify command)
```

Description

wap-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 will pass the string "SHUTDOWN_NOTIFICATION" as the parameter.

Return Values

string
Returns a null string.

Throws

None.

Additional Information

See **"Shutdown Command Notification Function" on page 27** for more information.

wap-outgoing

Syntax

```
(wap-outgoing event-string)
```

Description

wap-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 will be reexecuted with the same input Event.

"DATAERR" indicates the function had a problem processing data. If the e*Gate journal is enabled, the Event is journaled and the failed Event count is increased. (The input Event is essentially skipped in this process.) Use the **event-send-to-egate** function to place bad events in a bad event queue. See **event-send-to-egate** on page 29 for more information on this function.

Throws

None.

Additional Information

See **"Process Outgoing Message Function" on page 23** for more information.

wap-shutdown

Syntax

```
(wap-shutdown shutdown)
```

Description

wap-shutdown requests that the external connection 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** on page 31) 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 will pass 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 26** for more information.

wap-startup

Syntax

```
(wap-startup)
```

Description

wap-startup is used for function loads that are specific to this e*Way and invokes startup.

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 23** for more information.

wap-verify

Syntax

```
(wap-verify)
```

Description

wap-verify is used to verify whether the connection to the external system 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 25** for more information.

4.3 WAP Native Functions

The native WAP functions control the flow of information to and from the WAP application.

The functions described in this section can only be called from within a Collaboration Rules script.

The WAP native functions are

wap-cancel-alert

Syntax

```
(wap-cancel-alert subscriberID URL)
```

Description

wap-cancel-alert cancels the alert sent out by **wap-send-alert**.

Parameters

Name	Type	Description
subscriberID	string	The full subscriber ID of the mobile device to which to send the alert.
URL	string	The URL of the page that is to be accessed by the mobile device.

Return Values

Boolean

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

Throws

None.

wap-cancel-alert-with-params

Syntax

```
(wap-cancel-alert-with-params subscriberID URL timeOut port)
```

Description

wap-cancel-alert-with-params cancels the alert sent out by **wap-send-alert-with-params**.

Parameters

Name	Type	Description
subscriberID	string	The full subscriber ID of the mobile device to which to send the alert.
URL	string	The URL of the page that is to be access by the mobile device.
timeOut	integer	The number of seconds the function will wait for a response from the server.
port	integer	The port number on which the server is listening.

Return Values

Boolean

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

Throws

None.

Additional Information

To use the default for the **timeOut**, and **port** parameters, the value should be set to 0 (zero).

This function can be used to can an alert sent by wap-send-alert, but the port parameter must be set to the correct port number.

wap-send-alert

Syntax

```
(wap-send-alert subscriberID URL AlertTitle)
```

Description

wap-send-alert sends an alert message to the specified subscriber.

Parameters

Name	Type	Description
subscriberID	string	The full subscriber ID of the mobile device to which to send the alert.

Name	Type	Description
URL	string	The URL of the page that is to be accessed by the mobile device.
AlertTitle	string	The description of the alert.

Return Values

Boolean

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

Throws

None.

wap-send-alert-with-params

Syntax

```
(wap-send-alert-with-params subscriberID URL AlertTitle TTL timeOut
port)
```

Description

wap-send-alert-with-params sends an alert message to the specified subscriber with additional parameter limitations.

Parameters

Name	Type	Description
subscriberID	string	The full subscriber ID of the mobile device to which to send the alert.
URL	string	The URL of the page that is to be accessed by the mobile device.
AlertTitle	string	The description of the alert.
TTL	integer	The time-to-live for the alert, in seconds. If the alert has not been delivered before the TTL has expired the server will cancel the delivery.
timeOut	integer	The number of seconds the function will wait for a response from the server.
port	integer	The port number on which the server is listening.

Return Values

Boolean

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

Throws

None.

Additional Information

To use the default for the **TTL**, **timeOut**, and **port** parameters, the value should be set to 0 (zero).

Index