SeeBeyond ICAN Suite

e*Way Intelligent Adapter for Bloomberg Portfolio Management User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Contents

Chapter 1

Introduction 6

Overview 6
Components 6
Operational Overview 7
Supported Operating Systems 8
System Requirements 8
External System Requirements 8

Chapter 2

Installation 9

Installing the Bloomberg e*Way on Windows 9
Pre-installation 9
Installation Procedure 9
Installing the Bloomberg e*Way on UNIX 10
Pre-installation 10
Installation Procedure 10
Files/Directories Created by the Installation 10

Chapter 3

Configuration 14

Configuration Overview 14
e*Way Configuration Parameters 14
General Settings 15
Journal File Name 15
Max Resends Per Message 15
Max Failed Messages 16
Forward External Errors 16
Communication Setup 16
Start Exchange Data Schedule 16
Stop Exchange Data Schedule 17
Exchange Data Interval 17

Down Timeout 18
Up Timeout 18
Resend Timeout 18
Zero Wait Between Successful Exchanges 18
Monk Configuration 19
Operational Details 20
How to Specify Function Names or File Names 26
Additional Path 27
Auxiliary Library Directories 27
Monk Environment Initialization File 28
Startup Function 28
Process Outgoing Message Function 29
Exchange Data With External Function 29
External Connection Establishment Function 30
External Connection Verification Function 31
External Connection Shutdown Function 31
Positive Acknowledgment Function 32
Negative Acknowledgment Function 32
Shutdown Command Notification Function 33
Database Setup 33
Database Type 33
Database Name 33
User Name 34
Encrypted Password 34
TCPIP Configuration 34
Host 34
Port 34
PacketSize 34
Timeout 35
NoDelay 35
ACKValue 35
NACKValue 35
Bloomberg Settings 35
BloombergID 35
Pricing Number 36
Client ID 36

External Configuration Requirements 36
The External Bloomberg System 36
The External ODBC Database 36

Chapter 4

Implementation 37

Creating the Database Tables 37
Create the Tables 38
Editing the SQL Scripts 38

Installing the Bloomberg Sample Schema 39
Install the Sample Schema 39
Configure the Participating Hosts and e*Ways to Run the Schema 39
Configure the Participating Hosts 39
Configure the e*Ways 40

Run the Schema 40
The Bloomberg ETD Files 41

Chapter 5
Bloomberg e*Way Functions 43
Basic Functions 43
event-send-to-egate 43
get-logical-name 44
send-external-down 44
send-external-up 45
shutdown-request 45
start-schedule 46
stop-schedule 46
Bloomberg e*Way Functions 47
BB-conn-estab 47
BB-conn-shutdown 48
BB-conn-ver 48
BB-data-exchg 49
BB-init 50
BB-neg-ack 51
BB-poller-init 51
BB-pos-ack 52
BB-proc-outgoing 52
BB-shutdown 53
BB-startup 54
stcBBpoll 55
tcpip-exchange-BB 55
tcpip-outgoing-BB 56
tcpip-startup-BB 57
tcpip-verify-BB 57

Index 59

Chapter 1

Introduction

This document provides instructions for installing and configuring the SeeBeyond™ Technology Corporation's (SeeBeyond™) e*Way™ Intelligent Adapter for Bloomberg Portfolio Management. This chapter provides an introduction to the e*Way.

1.1 Overview

The Bloomberg e*Way provides a means for obtaining data from a Bloomberg Portfolio Trading System in near real-time.

This e*Way is intended to enhance the exchange of data with the Bloomberg system. The e*Way's connectivity solves problems associated with native file-based data exchange systems. Rather than exchanging a file—which leads to performance issues due to throughput and data parsing—the Bloomberg Portfolio Management e*Way provides a direct link to the Bloomberg system.

The Bloomberg data is quickly parsed and queued for subscribing e*Gate Integrator system components. The result is a fast and dependable stream of data exchanged with the Bloomberg network.

This Chapter Includes:

- **"Overview" on page 6**
- **"Operational Overview" on page 7**
- **"Supported Operating Systems" on page 8**
- **"System Requirements" on page 8**
- **"External System Requirements" on page 8**

1.1.1 Components

The Bloomberg e*Way is comprised of the following components:

- **stcewgenericmonk.exe**, the executable component
- Configuration files, which the e*Way Editor uses to define configuration parameters
- Monk function scripts, discussed in **Chapter 5**.

A complete list of installed files appears in **Table 1 on page 10**.

1.2 Operational Overview

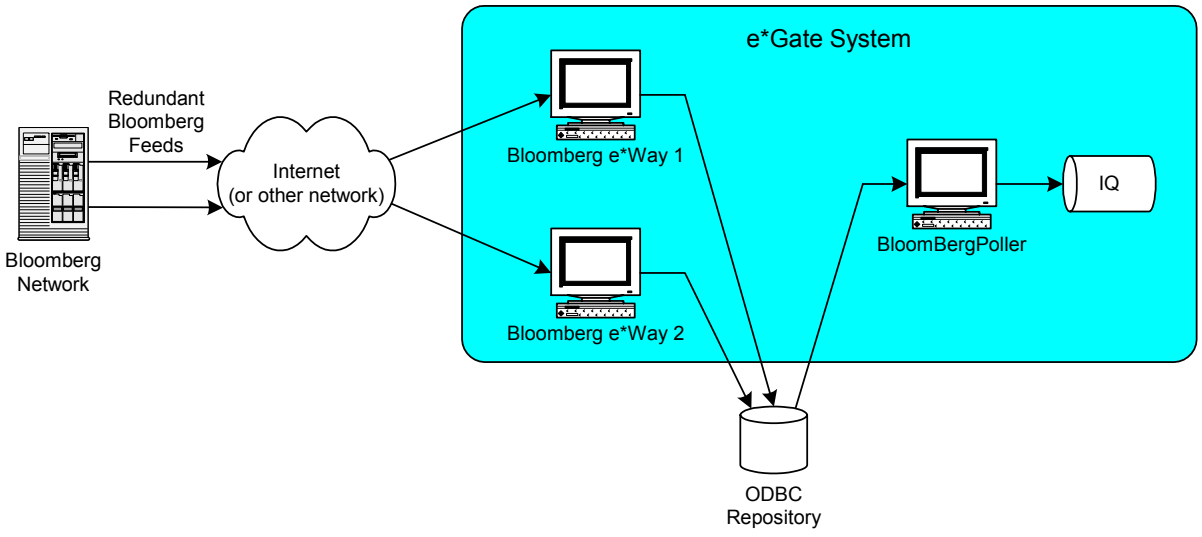
To ensure a fault-tolerant data exchange, the connection to the Bloomberg Portfolio Trading System requires redundant feeds to the Bloomberg network. Two (or more) connections to the Bloomberg network retrieving the same data ensure that no data is lost if one of the connections becomes broken. This requirement presents a unique challenge to process the redundant data without loading any duplicate transactions.

Bloomberg suggests that customers install an additional redundant feed to their network so if one feed goes down, the other still continues to function. A typical e*Gate scenario uses three Bloomberg e*Ways to load data from the Bloomberg network into the e*Gate system. Two Bloomberg e*Ways residing in different e*Gate schemas load data from the Bloomberg network via redundant TCP/IP connections.

Note: *For complete information on the e*Gate system, see the* ***e*Gate Integrator User's Guide****.*

These e*Ways save the transactions to an external ODBC-compliant data repository. A third Bloomberg e*Way — called the Bloomberg *Poller* e*Way — retrieves the transactions and filters out the duplicate records. The Bloomberg Poller e*Way publishes the Events (packets of data) to an Intelligent Queue (IQ) making them available to other e*Gate components (see **Figure 1 on page 7**).

Figure 1 Data Exchange with the Bloomberg Portfolio Trading System



1.3 Supported Operating Systems

The Bloomberg e*Way is available on the following operating systems:

- Windows 2000, Windows XP, and Windows Server 2003
- Sun Solaris 8 and 9

1.4 System Requirements

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

- An e*Gate Participating Host.
- Two TCP/IP network connections providing a redundant connection to the Bloomberg Portfolio Trading System.
- An ODBC-compliant database serving as a temporary repository for the data to be exchanged with the Bloomberg system.

Note: *The amounts of disk space listed previously are required on both the Participating and the Registry Host. Additional disk space is required to process and queue the data that this e*Way processes; the amount necessary can vary based on the type and size of the data being processed, as well as any external applications doing the processing.*

1.5 External System Requirements

The Bloomberg e*Way requires the following external systems/connections:

- A valid Bloomberg IP address for connecting to the Bloomberg system
- A valid Bloomberg Client ID for authenticating to the Bloomberg system

Chapter 2

Installation

This chapter describes how to install the Bloomberg Portfolio Management e*Way.

This Chapter Includes:

- **"Installing the Bloomberg e*Way on Windows" on page 9**
- **"Installing the Bloomberg e*Way on UNIX" on page 10**
- **"Files/Directories Created by the Installation" on page 10**

2.1 Installing the Bloomberg e*Way on Windows

2.1.1 Pre-installation

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

2.1.2 Installation Procedure

To install the Bloomberg e*Way on Windows systems

1. Log in as an Administrator to the workstation on which you are installing 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 launches automatically; skip ahead to step 4. Otherwise, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the file **setup.exe** on the CD-ROM drive.
4. The InstallShield setup application launches. Follow the installation instructions until you come to the **Please choose the product to install** dialog box.
5. Select **e*Gate Integrator**, then click **Next**.
6. Follow the on-screen instructions until you come to the second **Please choose the product to install** dialog box.

Note: *A typical Bloomberg Portfolio Management e*Way implementation will include two e*Ways using the* ***Bloomberg*** *template and one using the* ***Bloomberg Poller*** *template. For more information on implementing the e*Way in an e*Gate scenario, see* **Chapter 4**.

2.2 Installing the Bloomberg e*Way on UNIX

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 Bloomberg 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 will appear. Select the **Install e*Way** option. Then, follow the additional on-screen directions.

2.3 Files/Directories Created by the Installation

The Bloomberg 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 Name
[registry root]	stcewbbpts.ctl
configs\stcewgenericmonk\	Bloomberg.def BloombergPoller.def

Table 1 Files created by the installation

e*Gate Directory	File Name
monk_library\ewbbpts\	BB-db-utils.monk BB-funcs.monk BB-poller-init.monk tcpip-exchange-BB.monk tcpip-outgoing-BB.monk tcpip-startup-BB.monk tcpip-verify-BB.monk
monk_scripts\common\	BBPollDispatch.tsc checkbatchEOF.tsc putBBbatchdata.dsc putBBdata.dsc stcBBconnect.dsc stcBBconnectprocess.dsc stcBBcreatestatus.dsc stcBBpoll.dsc

Table 1 Files created by the installation

e*Gate Directory	File Name
monk_scripts\templates\bb_pts\	BatchPositionFeed.ssc BatchPriceFeed.ssc BBData.ssc BBDataHeader.ssc BBOnlineCounterPartyFeed_Complete.ssc BBOnlineCounterPartyFeed_Download.ssc BBOnlineCounterPartyFeed_Short.ssc BBOnlineNewSecurityFeed_Commodities.ssc BBOnlineNewSecurityFeed_Equities.ssc BBOnlineNewSecurityFeed_EquityOptionWarrants.ssc BBOnlineNewSecurityFeed_GovtsCorpsMunisPfrds.ssc BBOnlineNewSecurityFeed_Mortgages.ssc BBOnlineNewSecurityFeed_Munis.ssc BBOnlineNewSecurityFeed_Swaps.ssc BBOnlinePositionFeed.ssc BBOnlinePriceFeed.ssc BBOnlineTradeFeed.ssc BBOnlineTradeFeed_Repos.ssc BBOnlineTradeFeed_Swaps.ssc BBPacket.ssc BBPayloadWithHeader.ssc BBStatusHeartbeat.ssc CashFeed.ssc OnlineCompletedTransferMessage.ssc OnlineCounterPartyFeed_Complete.ssc OnlineCounterPartyFeed_Download.ssc OnlineCounterPartyFeed_Short.ssc OnlineNewSecurityFeed.ssc OnlineNewSecurityFeed_Commodities.ssc OnlineNewSecurityFeed_Equities.ssc OnlineNewSecurityFeed_EquityOptionWarrants.ssc OnlineNewSecurityFeed_GovtsCorpsMunisPfrds.ssc OnlineNewSecurityFeed_Mortgages.ssc OnlineNewSecurityFeed_Munis.ssc OnlineNewSecurityFeed_Swaps.ssc

Table 1 Files created by the installation

e*Gate Directory	File Name
	OnlinePositionFeed.ssc OnlinePriceFeed.ssc OnlineTradeFeed.ssc OnlineTradeFeed_Repos.ssc OnlineTradeFeed_Swaps.ssc stcBBbatchdata-mssql.ssc stcBBbatchdata-oracle.ssc stcBBbatchdata-sybase.ssc stcBBdata-mssql.ssc stcBBdata-oracle.ssc stcBBdata-sybase.ssc stcRoot.ssc WEBeSTP_Header.ssc
SQL\	ewbbpts_create_tables-mssql.sql ewbbpts_create_tables-oracle.sql ewbbpts_create_tables-sybase.sql

Chapter 3

Configuration

This chapter explains how to configure the Bloomberg e*Way.

3.1 Configuration Overview

Before you can run the Bloomberg e*Way, you must configure it using the e*Way Editor, which is accessed from the e*Gate Schema Manager GUI. The Bloomberg e*Way package includes a default configuration file which you can modify using this editor.

This Chapter Includes

- **"Configuration Overview" on page 14**
- **"General Settings" on page 15**
- **"Communication Setup" on page 16**
- **"Monk Configuration" on page 19**
- **"Database Setup" on page 33**
- **"TCPIP Configuration" on page 34**
- **"Bloomberg Settings" on page 35**
- **"External Configuration Requirements" on page 36**

3.2 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 Manager'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 **Bloomberg** 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.

For more information about how to use the e*Way Editor, see the e*Way Editor's online Help or the *Working with e*Ways* chapter in the *e*Gate Integrator User's Guide*.

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

- General Settings
- Communication Setup
- Monk Configuration
- Database Setup
- TCPIP Configuration
- Bloomberg Settings

3.2.1 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** 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 16** 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 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. If this parameter is set to **No**, then error messages will be ignored. See **"Exchange Data With External Function" on page 29** for more information.

Required Values

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

See **Figure 7 on page 24** for more information about how the e*Way uses this function.

3.2.2 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 Manager 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 Acknowledgment Function** and **Negative Acknowledgment Function**) 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 29**, **"Exchange Data Interval" on page 17**, and **"Stop Exchange Data Schedule" on page 17** 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 1 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.

See **"Down Timeout" on page 18** and **"Stop Exchange Data Schedule" on page 17** 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 30** 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 31** 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**. The default is **No**.

Additional Information

If this parameter is set to **Yes** and the previous exchange function returned data, then the e*Way will immediately invoke the **Exchange Data With External Function**. 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**.

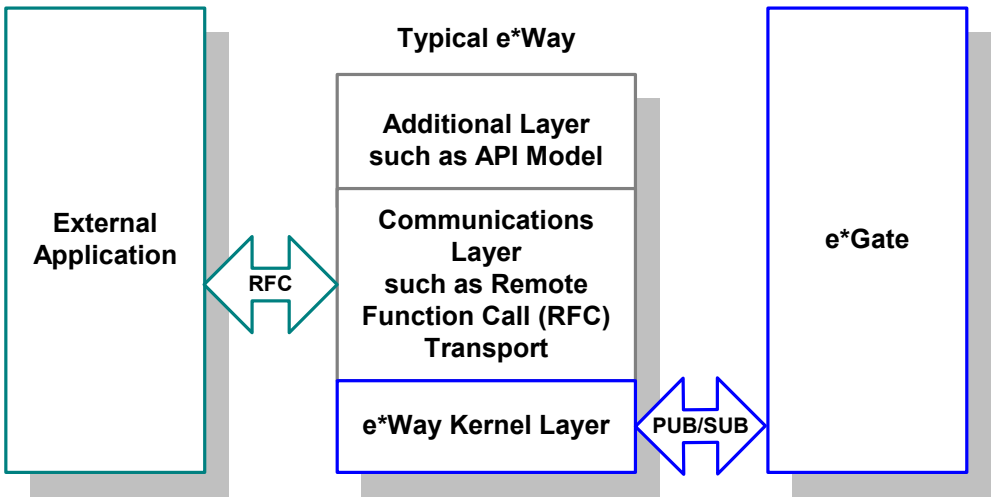
See **"Exchange Data With External Function" on page 29** for more information.

3.2.3 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 can be viewed as a multi-layered structure, consisting of one or more layers that handle communication with the external application, built upon an e*Way Kernel layer that manages the processing of data and subscribing or publishing to other e*Gate components (see Figure 2)

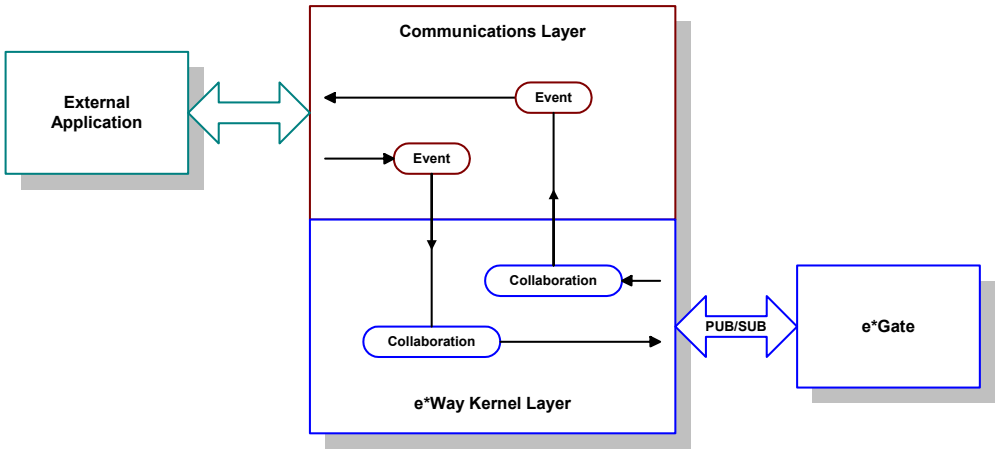
Figure 2 Typical e*Way Architecture



Each layer contains Monk scripts and/or functions, and makes use of lower-level Monk functions residing in the layer beneath. You, as user, primarily use the highest-level functions, which reside in the upper layer(s).

The upper layers of the e*Way use Monk functions to start and stop scheduled operations, exchange data with the external system, package data as e*Gate "Events," send those Events to Collaborations, and manage the connection between the e*Way and the external system (see Figure 3).

Figure 3 Basic e*Way Operations



Configuration options that control the Monk environment and define the Monk functions used to perform these basic e*Way operations are discussed in **Chapter 4**. You can create and modify these functions using the SeeBeyond Collaboration Rules Editor or a text editor (such as Windows **Notepad** or UNIX **vi**).

The upper layers of the e*Way are single-threaded. Functions run serially, and only one function can be executed at a time. The e*Way Kernel 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.

The basic set of e*Way Kernel Monk functions is described in **Chapter 5**. Generally, e*Way Kernel Monk functions should be called directly only when there is a specific need not addressed by higher-level Monk functions, and should be used only by experienced developers.

Operational Details

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

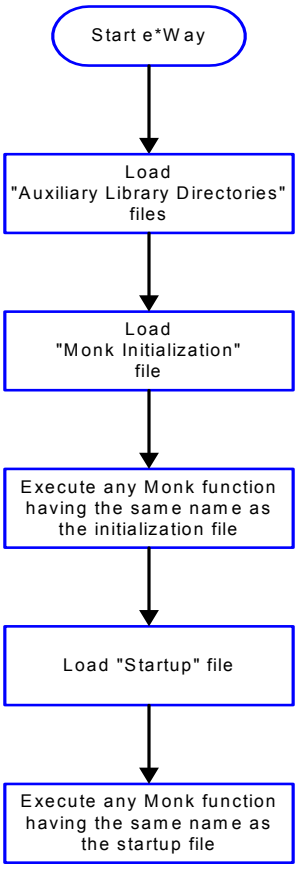
Type of Operation	Name
Initialization	**Startup Function** on page 28 (also see **Monk Environment Initialization File** on page 28)
Connection	**External Connection Establishment Function** on page 30 **External Connection Verification Function** on page 31 **External Connection Shutdown Function** on page 31
Schedule-driven data exchange	**Exchange Data With External Function** on page 29 **Positive Acknowledgment Function** on page 32 **Negative Acknowledgment Function** on page 32
Shutdown	**Shutdown Command Notification Function** on page 33
Event-driven data exchange	**Process Outgoing Message Function** on page 29

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

Initialization Functions

Figure 4 illustrates how the e*Way executes its initialization functions.

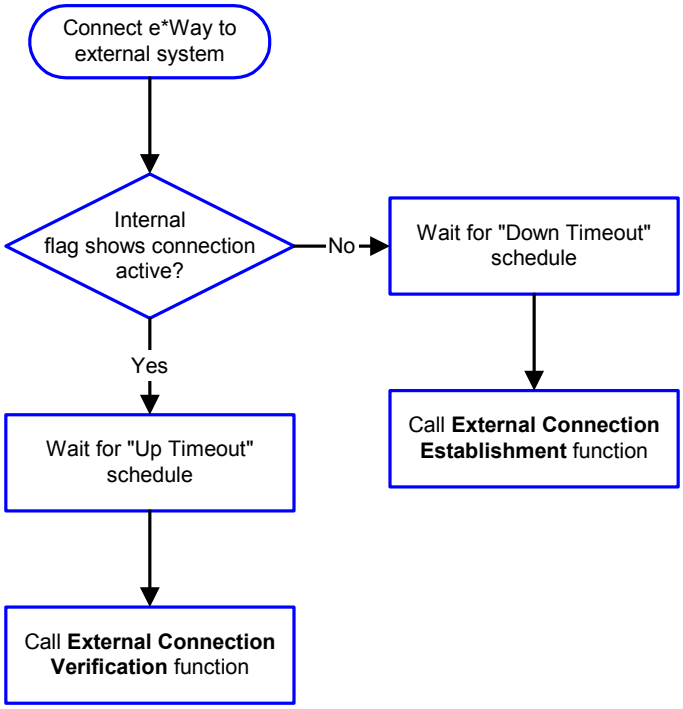
Figure 4 Initialization Functions



Connection Functions

Figure 5 illustrates how the e*Way executes the connection establishment and verification functions.

Figure 5 Connection establishment and verification functions

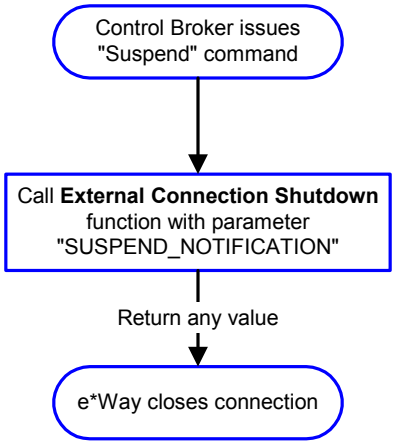


Note: *The e*Way selects the connection function based on an internal "up/down" flag rather than a poll to the external system. See* **Figure 7 on page 24** *and* **Figure 9 on page 26** *for examples of how different functions use this flag.*

User functions can manually set this flag using Monk functions. See **send-external-up** *on page 45 and* **send-external-down** *on page 44 for more information.*

Figure 6 illustrates how the e*Way executes its "connection shutdown" function.

Figure 6 Connection shutdown function



Schedule-driven Data Exchange Functions

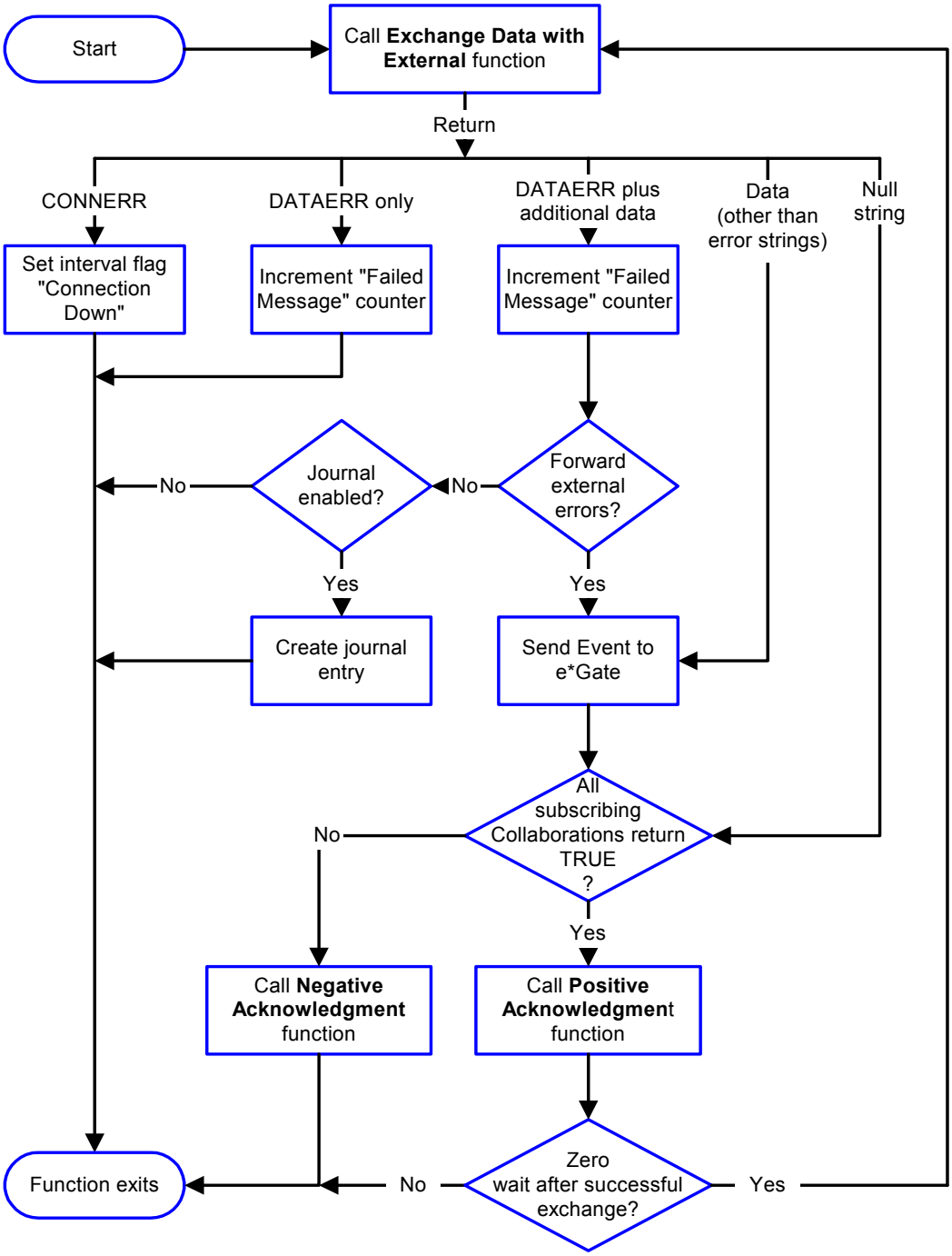
Figure 7 (on the next page) illustrates how the e*Way performs schedule-driven data exchange using the **Exchange Data With External Function**. The **Positive Acknowledgment Function** and **Negative Acknowledgment Function** are also called during this process.

"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.

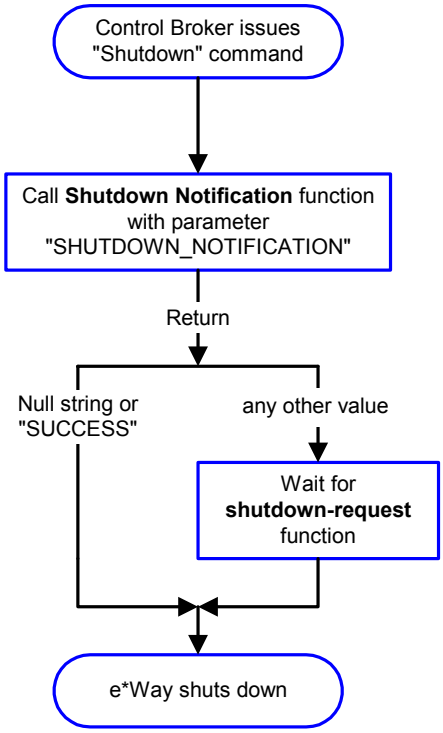
Figure 7 Schedule-driven data exchange functions



Shutdown Functions

Figure 8 illustrates how the e*Way implements the shutdown request function.

Figure 8 Shutdown functions



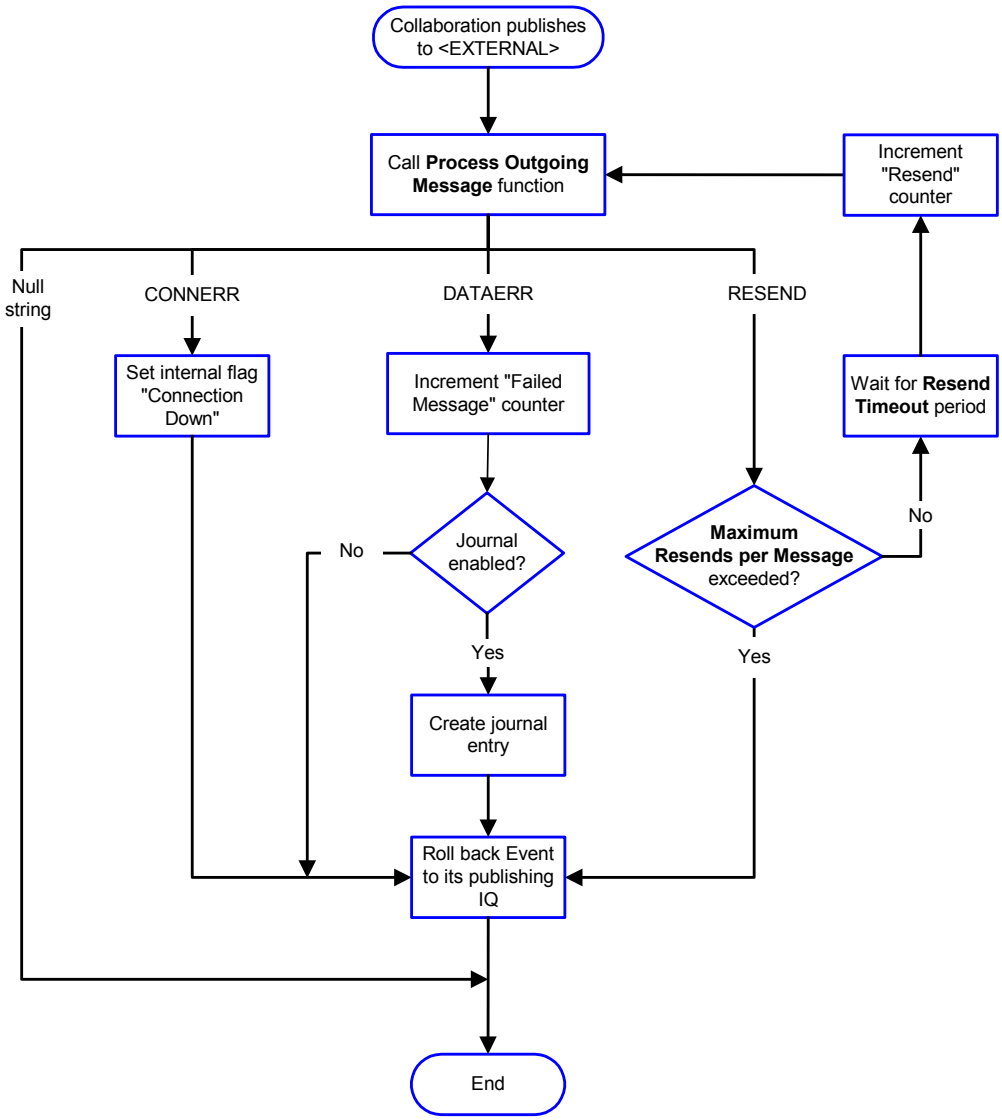
Event-driven Data Exchange Functions

Figure 9 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 9 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/ewtcpipext;monk_library/dart;monk_library/ewbbpts.
```

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.

In addition to loading it's own libraries, the Bloomberg e*Way loads the libraries normally loaded by the TCP/IP e*Way and ODBC e*Way.

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 27** for more information about the "load path."

The default is **BB-init**. See **BB-init** on page 50 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 4 on page 21**).

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 **BB-startup**. See **BB-startup** on page 54 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 4 on page 21**). 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 **BB-proc-outgoing**. See **BB-proc-outgoing** on page 52 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 Manager). The function returns one of the following (see **Figure 9 on page 26** 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 queue 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 43* *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. However, this parameter is required if a schedule was set using the **Start Exchange Data Schedule** parameter. If so, you must also define the following:

- **Positive Acknowledgment Function**
- **Negative Acknowledgment Function**

The default is **BB-data-exchg**. See **BB-data-exchg** on page 49 for more information.

Additional Information

The function accepts no input and must return a string (see **Figure 9 on page 26** 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 9 on page 26** 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 Exchange Data 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 46 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 **BB-con-estab**. See **BB-conn-estab** on page 47 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 **BB-con-ver**. See **BB-conn-ver** on page 48 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 or is unknown.

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 **BB-con-shutdown**. See **BB-conn-shutdown** on page 48 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 queued 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 **BB-pos-ack**. See **BB-pos-ack** on page 52 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 **BB-neg-ack**. See **BB-neg-ack** on page 51 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.

The default is **BB-shutdown**. See **BB-shutdown** on page 53 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 45).

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.*

3.2.4 Database Setup

Database Type

Description

Specifies the type of database.

Required Values

ODBC

Database Name

Description

Specifies the name of the database.

Required Values

None. Any valid string.

User Name

Description

Specifies the name used to access the database.

Required Values

Any valid string.

Encrypted Password

Description

The password that provides access to the database.

Required Values

Any valid string.

3.2.5 TCPIP Configuration

Host

Description

Specifies the Host on which the server is running.

Required Values

A string containing a valid hostname.

Port

Description

Specifies the port on which the server is listening for connection requests.

Required Values

An integer between 1 and 864,000. 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 864,000. The default is 4096.

Timeout

Description

Specifies the amount of time in milliseconds that the e*Way will await a response from the server.

Required Values

An integer between 1 and 864,000. The default is 50,000.

NoDelay

Description

Specifies whether connections or requests be delayed.

Required Values

FALSE or **TRUE**. The default is **TRUE**.

ACKValue

Description

Specifies the positive acknowledgment return value.

Required Values

Any valid string. The default is **ACK**.

NACKValue

Description

Specifies the negative acknowledgment return value.

Required Values

Any valid string. The default is **NACK**.

3.2.6 Bloomberg Settings

BloombergID

Description

Specifies whether the e*Way is a primary or secondary e*Way.

Required Values

Primary or **Secondary**. The default is **Primary**.

Additional Information

The designation of primary and secondary e*Ways is used as the Bloomberg data is saved to the database. Each record is marked as having been loaded by either the

primary or secondary Bloomberg e*Way. This designation makes it possible for the downstream e*Gate components to load only one instance of each record.

Pricing Number

Description

Specifies the pricing number provided by Bloomberg.

Required Values

A string containing a valid pricing number.

Client ID

Description

Specifies the client identifier provided by Bloomberg.

Required Values

A string containing a valid client identifier.

3.3 External Configuration Requirements

3.3.1 The External Bloomberg System

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

3.3.2 The External ODBC Database

The Bloomberg e*Way requires that specific tables be set up in the external database. A series of SQL scripts are included in the Bloomberg e*Way's installation package. These SQL scripts will create the necessary tables.

For more information regarding the use of these SQL scripts, see **"Create the Tables" on page 38**.

Chapter 4

Implementation

This chapter provides instructions for creating the database tables used by the Bloomberg e*Way and installing the sample schema from the e*Gate CD.

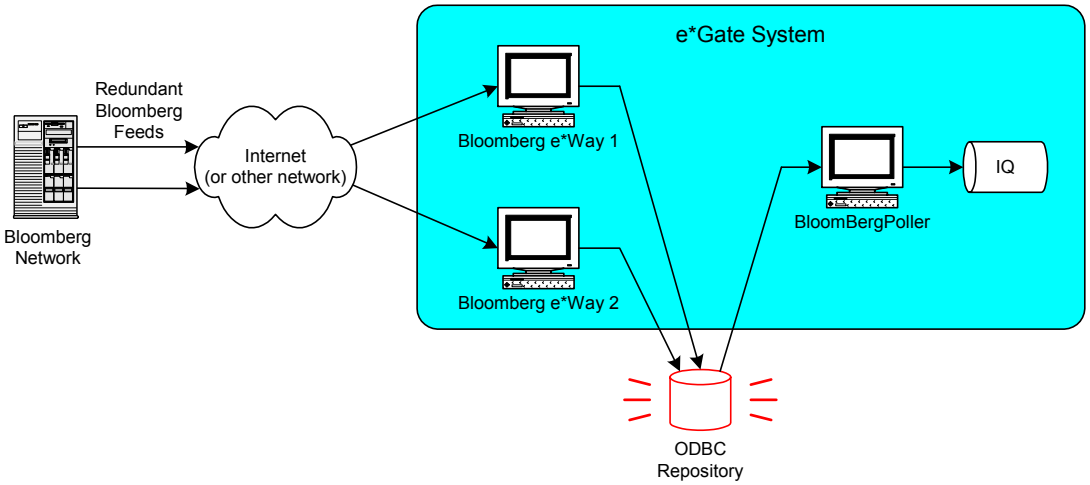
This Chapter Describes:

- **"Creating the Database Tables" on page 37**
- **"Installing the Bloomberg Sample Schema" on page 39**

4.1 Creating the Database Tables

The Bloomberg e*Way uses an external database as a repository for the transactions loaded by the redundant Bloomberg connections. These redundant feeds from the Bloomberg network create a collection of duplicate transactions. The role of the external database is to store these duplicate transactions until the Bloomberg Poller e*Way can process them. The Bloomberg Poller e*Way will load a single instance of each of the duplicate records and mark both of the records as "processed."

Figure 10 The ODBC Data Repository



4.1.1 Create the Tables

The e*Gate installation CD includes a set of SQL scripts to be used to create the tables required for the Bloomberg e*Way. The SQL scripts are located in the ***d:*\samples\ewbbpts** directory (where ***d:*** is the drive letter for your CD-ROM).

The three SQL scripts are:

- **ewbbpts_create_tables_oracle.sql** – This SQL script is intended for use with Oracle databases. It can be executed from the command line using the SQL*Plus utility.
- **ewbbpts_create_tables_sybase.sql** – This SQL script is intended for use with Sybase databases. It can be executed from the command line using the ISSQL utility.
- **ewbbpts_create_tables_mssql.sql** – This SQL script is intended for use with MS SQL databases. However, it can be modified to work with other ODBC-compliant databases.

Note: *The SQL scripts contain a basic framework that will work for most databases. However, the specific syntax of the instructions may not work on all DBMS versions. For best results, review the contents of these scripts before executing them.*

Editing the SQL Scripts

In addition to creating the database tables, the SQL scripts create initial records that will be used to synchronize the inbound Bloomberg e*Ways with the Bloomberg feed. Before executing the scripts, they must be edited so that the initial request date matches the data in the Bloomberg network.

To edit the SQL script:

1. Copy the appropriate SQL script from the ***d:*\samples\ewbbpts** directory (where ***d:*** is the drive letter for your CD-ROM) onto your local machine.

Note: *Copy the SQL script that is appropriate for your database type: Oracle, Sybase, or MS SQL.*

2. Use a text editor to open the SQL script.
3. Change the TradeDate to correspond to a recent trading date for your Bloomberg account. See the example below:

```
INSERT INTO stcBBseq (ID, TradeDate, SeqNextVal)
VALUES ('Primary', '05/05/01', 0)
GO

INSERT INTO stcBBseq (ID, TradeDate, SeqNextVal)
VALUES ('Secondary', '05/05/01', 0)
GO
```

Note: *The date must be in **mm/dd/yy** format.*

4. Save the changes to the SQL script.

4.2 Installing the Bloomberg Sample Schema

The e*Gate Installation CD contains a sample schema to demonstrate a simple scenario using the Bloomberg e*Way. To optimize fault tolerance, the Bloomberg e*Way features two redundant feeds from the Bloomberg network, each of which utilize a participating host on a separate machine.

4.2.1 Install the Sample Schema

The following procedure explains how to install the sample schema on the Registry Host.

To install the sample schema

1. Copy the file named **BloombergPTS.zip** from the **samples\ewbbpts** directory on your install CD-ROM to your desktop or to a temporary directory, and then unzip the file.
2. Start e*Gate Schema Manager.
3. On the **File** menu, click **Import Definitions from File**.
4. On the first page of the **Import Wizard**, click **Next**.
5. On the **Select Import** page, click **Schema**, and then click **Next**.
6. On the **Import Schema** page, browse to the directory that contains the sample schema, click **BloombergPTS.zip**, and then click **Open**.

   The sample schema is installed.

Configure the Participating Hosts and e*Ways to Run the Schema

As mentioned previously, the first participating host, **Host1,** receives one of the redundant feeds from the Bloomberg network, while the other host machine, **Host2**, receives the other redundant feed.

The first participating host machine, **Host1**, uses two e*Ways: **BBInbound1**, which receives the Events from the Bloomberg feed and passes them to the ODBC (database) repository, and **BBPoller**, which pulls a single Event copy out of the ODBC repository and passes it to e*Gate.

The second host machine, **Host2**, only uses one e*Way, **BBInbound2**, to receive the Events from the Bloomberg feed and pass them to the ODBC repository. The steps to configure each participating hosts are described below.

Configure the Participating Hosts

To configure the participating hosts to run the sample schema:

1. Once the schema is imported, open the Schema Manager--you will see **Host1** and **Host2**.

2 Highlight and right-click on **Host1**, select **Properties**, and then select the **General** tab.

3 In the **General** tab, input appropriate information into the **Network Host Name or IP Address** field and, if necessary, into the **Network Domain Name** field.

Configure the e*Ways

To modify the configurations of the BBInbound1, BBInbound2, and BBPoller e*Ways:

1 To modify the configuration of the **BBInbound1** e*Way of Host1 and **BBInbound2** e*Way of Host2, select the e*Way, right-click, and select **Configuration**.

2 In the **Goto** section, use the drop-down menu to select, and go to, the following Configuration sections and select configuration settings appropriately for your system:

- **Database Setup**
- **TCPIP Configuration**
- **Bloomberg Settings** (for **BBInbound1**, this setting should be ***Primary***; for **BBInbound2**, the setting should be ***Secondary***)
- **General Settings** and **Configuration Setup** (typically, review only)
- **Monk** settings should be considered READ-ONLY

3 To modify the configuration of the **BBPoller** e*Way of Host1, select the e*Way, right-click, and select **Configuration**.

4 In the **Goto** section, use the drop-down menu to select, and go to, the following Configuration sections and select configuration settings to create a configuration file appropriate for your system:

- **Database Type** - select ***ODBC***
- **Database Name**
- **User Name**
- **Password**
- **General Settings and Communication Setup**

4.2.2 Run the Schema

Note: *To run the schema successfully and ensure that it performs the function for which it was intended,* ***ensure that each host has its schema running when data is transferred****.*

To run the sample schema on Participating Host 1:

1 Start the Control Broker "**host1_cb**" at the host1 machine (the machine containing the Registry Host and Participating Host 1).

2 At the command prompt, type:

```
stccb.exe -1n host1_cb -rh <registry host name> -rs BloombergPTS
-un <user name> -up <user password>
```

To run the sample schema on Participating Host 2:

1. Start the Control Broker "**host2_cb**" at the host2 machine (the second machine, containing only Participating Host 2)
2. At the command prompt, type:

```
stccb.exe -1n host2_cb -rh <registry host name> -rs BloombergPTS
-un Administrator -up STC
```

Result:

A single Bloomberg data feed is put into the "iqBB" and is currently output into files by "TestFileOut" e*Way. Modify these downstream components at will.

4.3 The Bloomberg ETD Files

The Bloomberg e*Way installation program installs six ETDs:

- **stcBBdata_mssql.ssc** – The ETD file to be used with MS SQL databases.
- **stcBBbatchdata_mssql.ssc** – The ETD file to be used with MS SQL databases for batch transactions.
- **stcBBdata_oracle.ssc** – The ETD file to be used with Oracle databases.
- **stcBBatchdata_oracle.ssc** – The ETD file to be used with Oracle databases for batch transactions.
- **stcBBdata_sybase.ssc** – The ETD file to be used with Sybase databases.
- **stcBBbatchdata_sybase.ssc** – The ETD file to be used with Sybase databases for batch transactions.

Note: *For a complete list of files included with this e*Way, see* **Table 1 on page 10**.

These ETD files are formatted to support the data restraints within each of the corresponding DBMS systems.

To implement the ETD file:

1. Use Windows Explorer to navigate to the **monk_scripts\templates\bb_pts** directory within your e*Gate schema.
2. Make a copy of the appropriate ETD file in this same directory:
   - **stcBBdata_mssql.ssc**,
   - **stcBBbatchdata_mssql.ssc**,
   - **stcBBdata_oracle.ssc**,
   - **stcBBbatchdata_oracle.ssc**,
   - **stcBBdata_sybase.ssc**,

- **stcBBbatchdata_sybase.ssc**,

3 Rename the copied ETD file—name the file **stcBBdata.ssc** or **stcBBbatchdata.ssc**, as appropriate.

Chapter 5

Bloomberg e*Way Functions

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

- **Basic Functions** on page 43
- **Bloomberg e*Way Functions** on page 47

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

The basic functions are

event-send-to-egate on page 43

get-logical-name on page 44

send-external-down on page 44

send-external-up on page 45

shutdown-request on page 45

start-schedule on page 46

stop-schedule on page 46

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 Schema 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 requests the e*Way to perform the shutdown procedure when there is no outstanding incoming/outgoing event. When the e*Way is ready to act on the shutdown request, in invokes the **Shutdown Command Notification Function** (see**"Shutdown Command Notification Function" on page 33**). Once this function is called, the shutdown proceeds immediately.

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.

5.2 Bloomberg e*Way Functions

The functions described in this section control the way the e*Way interacts with the external database as well as the external Bloomberg system.

The Bloomberg e*Way functions are:

BB-conn-estab on page 47

BB-conn-shutdown on page 48

BB-conn-ver on page 48

BB-data-exchg on page 49

BB-init on page 50

BB-neg-ack on page 51

BB-poller-init on page 51

BB-pos-ack on page 52

BB-proc-outgoing on page 52

BB-shutdown on page 53

BB-startup on page 54

stcBBpoll on page 55

tcpip-exchange-BB on page 55

tcpip-outgoing-BB on page 56

tcpip-startup-BB on page 57

tcpip-verify-BB on page 57

BB-conn-estab

Syntax

```
(BB-conn-estab)
```

Description

BB-conn-estab is used to establish connections with the external database and TCP/IP systems. This function uses the **db-stdver-conn-estab** function to establish the connection with the external database and the **tcpip-startup-BB** function to establish the connection with the external Bloomberg system.

Parameters

None.

Return Values

string
An **UP** string is returned after both the **db-stdver-conn-estab** and **tcpip-startup-BB** functions return **SUCCESS** (or **UP**). If either of these functions return any other string, a **Connection Estab FAILED** string is returned.

Throws

None.

Additional Information

See **"External Connection Establishment Function" on page 30** for more information.

BB-conn-shutdown

Syntax

```
(BB-conn-shutdown)
```

Description

BB-conn-shutdown is called by the system to request that the interface disconnect from the external systems, preparing for a suspend/reload cycle. Any return value indicates that the suspend can occur immediately, and the interface will be placed in the down state.

This function uses the **db-stdver-conn-shutdown** and **tcpip-shutdown** functions to initiate the shutdown of the connections to the external database and external Bloomberg system.

Parameters

None.

Return Values

string
A **SUCCESS** string is returned after both the **db-stdver-conn-shutdown** and **tcpip-shutdown** functions return **SUCCESS** (or **UP**). If either of these functions return any other string, a **Shutdown FAILED** string is returned.

Throws

None.

Additional Information

See **"External Connection Shutdown Function" on page 31** for more information.

BB-conn-ver

Syntax

```
(BB-conn-ver)
```

Description

BB-conn-ver is used to verify whether the external system connection has been established. This function uses the **db-stdver-conn-ver** and **tcpip-verify-BB** functions to verify the state of the connections to the external database and external Bloomberg system.

Parameters

None.

Return Values

string
An **UP** string is returned after both the **db-stdver-conn-ver** and **tcpip-verify-BB** functions return **SUCCESS** (or **UP**). If either of these functions return any other string, a **Verify FAILED** string is returned and the e*Way will shut down.

Throws

None.

Additional Information

See **"External Connection Verification Function" on page 31** for more information.

BB-data-exchg

Syntax

```
(BB-data-exchg)
```

Description

BB-data-exchg exchanges data with the external Bloomberg system and saves it to the external database. This function uses the **tcpip-exchange-BB** function to determine starting and ending signals from the Bloomberg server and uses these signals to conduct the data exchange before saving the received Events to the external database.

Parameters

None.

Return Values

string
An empty string indicates a successful operation. The e*Way will then be able to proceed with the next exchange.

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

This function is a Dart script to be used by the Bloomberg e*Way to exchange data with the external Bloomberg system.

See **"Exchange Data With External Function" on page 29** for more information. For more information about the **tcpip-exchange-BB** function, see **tcpip-exchange-BB** on page 55.

BB-init

Syntax

```
(BB-init)
```

Description

BB-init begins the initialization process for the e*Way. This function defines the e*Way's global variables and loads the **db-stdver-init** and **tcpip-init** functions.

Parameters

None.

Return Values

string

A **SUCCESS** string is returned after both the **db-stdver-init** and **tcpip-init** functions return **SUCCESS**. If either of these functions return any other string, a **FAILURE** string is returned and the e*Way will shut down.

Throws

None.

Additional Information

Within this function, the following global variables are defined:

- BB_db_conn
- BB_tcipip_conn
- BB_accept_recvd
- BB_last_data
- BBbatchseq
- BloombergID

Additional global variables can also be defined within this function. These global variables will be available to other functions used by this e*Way. The internal function that loads this file is called once when the e*Way first starts up.

See **"Monk Environment Initialization File" on page 28** for more information.

BB-neg-ack

Syntax

```
(BB-neg-ack message-string)
```

Description

BB-neg-ack uses the **tcpip-outgoing-BB** function to send 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 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.

Throws

None.

Additional Information

See **"Negative Acknowledgment Function" on page 32** for more information. For more information about the **tcpip-outgoing-BB** function, see **tcpip-outgoing-BB** on page 56.

BB-poller-init

Syntax

```
(BB-poller-init)
```

Description

BB-poller-init loads the functions defined in **BB-db-utils.monk** and **BB-funcs.monk** then loads the **db-stdver-init** function.

Parameters

None.

Return Values

string

An empty string indicates a successful operation.

If a **FAILURE** string is returned, the e*Way will shut down.

Throws

None.

Additional Information

See **"Monk Environment Initialization File" on page 28** for more information.

BB-pos-ack

Syntax

```
(BB-pos-ack message-string)
```

Description

BB-pos-ack uses the **tcpip-outgoing-BB** function to send a positive acknowledgment to the external system after all Collaborations to which the e*Way sent data have processed and queued 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 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.

Throws

None.

Additional Information

See **"Positive Acknowledgment Function" on page 32** for more information. For more information about the **tcpip-outgoing-BB** function, see **tcpip-outgoing-BB** on page 56.

BB-proc-outgoing

Syntax

```
(BB-proc-outgoing message-string)
```

Description

BB-proc-outgoing is used for sending a received message (Event) from e*Gate to the external system.

Parameters

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

Return Values

string

An empty string indicates a successful operation.

RESEND causes the message to be immediately resent. The e*Way will compare the number of attempts it has made to send the Event to the number specified in the **Max Resends Per Messages** parameter, and does one of the following:

1. If the number of attempts does not exceed the maximum, the e*Way will pause the number of seconds specified by the **Resend Timeout** parameter, increment the "resend attempts" counter for that message, then repeat the attempt to send the message.
2. If the number of attempts exceeds the maximum, the function returns false and rolls back the message to the e*Gate IQ from which it was obtained.

CONNERR indicates that there is a problem communicating with the external system. First, the e*Way will pause the number of seconds specified by the **Resend Timeout** parameter. Then, the e*Way will call the **External Connection Establishment function according to the Down Timeout schedule, and** will roll back the message (Event) to the IQ from which it was obtained.

DATAERR indicates that there is a problem with the message (Event) data itself. First, the e*Way will pause 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 (see **"Journal File Name" on page 15**) the message (Event) will be journaled.

If any other string 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.

Throws

None.

BB-shutdown

Syntax

```
(BB-shutdown shutdown)
```

Description

BB-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 will pass the string **SUSPEND_NOTIFICATION** as the parameter.

Return Values

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

Throws

None.

Additional Information

See **"External Connection Shutdown Function" on page 31** for more information.

BB-startup

Syntax

```
(BB-startup)
```

Description

BB-startup loads the **db-stdver-startup** function to start the connection to the external database.

Parameters

None.

Return Values

string
A **SUCCESS** string is returned after the **db-stdver-startup** function returns **SUCCESS** or **UP**. If the **db-stdver-startup** function returns any other string, a **FAILURE** string is returned and the e*Way will shut down.

Throws

None.

Additional Information

This function should be used to initialize the external system before data exchange starts.

See **"Startup Function" on page 28** for more information.

stcBBpoll

Syntax

```
(stcBBpoll)
```

Description

stcBBpoll retrieves the first instance of each transaction from the external database where **PROCESSED** = **NO**. Once the records have been retrieved and queued in the e*Gate system, the **PROCESSED** value is set to **YES** in the external database.

Parameters

None.

Return Values

string

An empty string indicates a successful operation. The e*Way will then be able to proceed with the next exchange.

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

This function is a Dart script to be used by the BloombergPoller e*Way to exchange data with the external database.

See **"Exchange Data With External Function" on page 29** for more information.

tcpip-exchange-BB

Syntax

```
(tcpip-exchange-BB)
```

Description

tcpip-exchange-BB 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 re-executed with the same input Event.

Throws

None.

Additional Information

See **"Exchange Data With External Function" on page 29** for more information.

tcpip-outgoing-BB

Syntax

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

Description

tcpip-outgoing-BB 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 re-executed 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 43 for more information.

Throws

None.

Additional Information

See **"Process Outgoing Message Function" on page 29** for more information.

tcpip-startup-BB

Syntax

```
(tcpip-startup-BB)
```

Description

tcpip-startup-BB 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 28** for more information.

tcpip-verify-BB

Syntax

```
(tcpip-verify-BB)
```

Description

tcpip-verify-BB 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 31** for more information.

Index

A

Additional Path parameter 27
Auxiliary Library Directories parameter 27

B

BatchPositionFeed.ssc 12
BatchPriceFeed.ssc 12
BB-conn-estab 47
BB-conn-shutdown 48
BB-conn-ver 48
BBData.ssc 12
BB-data-exchg 49, 55
BBDataHeader.ssc 12
BB-db-utils.monk 11
BB-funcs.monk 11
BB-init 50
BB-neg-ack 51
BBOnlineCounterPartyFeed_Complete.ssc 12
BBOnlineCounterPartyFeed_Download.ssc 12
BBOnlineCounterPartyFeed_Short.ssc 12
BBOnlineNewSecurityFeed_Commodities.ssc 12
BBOnlineNewSecurityFeed_Equities.ssc 12
BBOnlineNewSecurityFeed_EquityOptionWarrants.ssc 12
BBOnlineNewSecurityFeed_GovtsCorpsMunisPfrds.ssc 12
BBOnlineNewSecurityFeed_Mortgages.ssc 12
BBOnlineNewSecurityFeed_Munis.ssc 12
BBOnlineNewSecurityFeed_Swaps.ssc 12
BBOnlinePositionFeed.ssc 12
BBOnlinePriceFeed.ssc 12
BBOnlineTradeFeed.ssc 12
BBOnlineTradeFeed_Repos.ssc 12
BBOnlineTradeFeed_Swaps.ssc 12
BBPacket.ssc 12
BBPayloadWithHeader.ssc 12
BBPollDispatch.tsc 11
BB-poller-init 51
BB-poller-init.monk 11
BB-pos-ack 52
BB-proc-outgoing function 52
BB-shutdown 53
BB-startup 54
BBStatusHeartbeat.ssc 12
Bloomberg e*Way functions
  BB-conn-estab 47
  BB-conn-shutdown 48
  BB-conn-ver 48
  BB-data-exchg 49, 55
  BB-init 50
  BB-neg-ack 51
  BB-poller-init 51
  BB-pos-ack 52
  BB-proc-outgoing 52
  BB-shutdown 53
  BB-startup 54
  tcpip-exchange-BB 55
  tcpip-outgoing-BB 56
  tcpip-startup-BB 57
  tcpip-verify-BB 57
Bloomberg settings 35
Bloomberg.def 10
BloombergPoller 10
BloombergPoller.def 10

C

CashFeed.ssc 12
checkbatchEOF.tsc 11
communication setup 16
configuration parameters
  Additional Path 27
  Auxiliary Library Directories 27
  Down Timeout 18
  Exchange Data Interval 17
  Exchange Data With External Function 29
  External Connection Establishment Function 30
  External Connection Shutdown Function 31
  External Connection Verification Function 31
  Forward External Errors 16
  Journal File Name 15
  Max Failed Messages 16
  Max Resends Per Message 15
  Monk Environment Initialization File 28
  Negative Acknowledgment Function 32
  Positive Acknowledgement Function 32
  Process Outgoing Message Function 29
  Resend Timeout 18
  Shutdown Command Notification Function 33
  Start Exchange Data Schedule 18
  Startup Function 28
  Stop Exchange Data Schedule 17
  Up Timeout 18
  Zero Wait Between Successful Exchanges 18

D

database name 33
database setup 33
  database name 33
  database type 33
  encrypted password 34
  user name 34
database type 33
Down Timeout parameter 18

E

e*Way configuration parameters 14
encrypted password 34
event-send-to-egate 43
ewbbpts_create_tables-mssql.sql 13
ewbbpts_create_tables-oracle.sql 13
ewbbpts_create_tables-sybase.sql 13
Exchange Data Interval parameter 17
Exchange Data With External Function parameter 29
External Connection Establishment Function parameter 30
External Connection Shutdown Function parameter 31
External Connection Verification Function parameter 31

F

Forward External Errors parameter 16
functions
  BB-conn-estab 47
  BB-conn-shutdown 48
  BB-conn-ver 48
  BB-data-exchg 49, 55
  BB-init 50
  BB-neg-ack 51
  BB-poller-init 51
  BB-pos-ack 52
  BB-proc-outgoing 52
  BB-shutdown 53
  BB-startup 54
  event-send-to-egate 43
  get-logical-name 44
  send-external-down 44
  send-external-up 45
  start-schedule 46
  stop-schedule 46
  tcipip-exchange-BB 55
  tcpip-outgoing-BB 56
  tcpip-startup-BB 57
  tcpip-verify-BB 57

G

General Settings 15
get-logical-name 44

I

installation 10
  files created 10
  UNIX 10

J

Journal File Name parameter 15

M

Max Failed Messages parameter 16
Max Resends Per Message parameter 15
Monk Configuration 19
Monk Environment Initialization File parameter 28

N

Negative Acknowledgment Function parameter 32

O

OnlineCompletedTransferMessage.ssc 12
OnlineCounterPartyFeed_Complete.ssc 12
OnlineCounterPartyFeed_Download.ssc 12
OnlineCounterPartyFeed_Short.ssc 12
OnlineNewSecurityFeed.ssc 12
OnlineNewSecurityFeed_Commodities.ssc 12
OnlineNewSecurityFeed_Equities.ssc 12
OnlineNewSecurityFeed_EquityOptionWarrants.ssc 12
OnlineNewSecurityFeed_GovtsCorpsMunisPfrds.ssc 12
OnlineNewSecurityFeed_Mortgages.ssc 12
OnlineNewSecurityFeed_Munis.ssc 12
OnlineNewSecurityFeed_Swaps.ssc 12
OnlinePositionFeed.ssc 13
OnlinePriceFeed.ssc 13
OnlineTradeFeed.ssc 13
OnlineTradeFeed_Repos.ssc 13
OnlineTradeFeed_Swaps.ssc 13

P

parameters
  database name 33
  database setup 33
  database type 33

encrypted password **34**
user name **34**
Positive Acknowledgement Function parameter **32**
Process Outgoing Message Function parameter **29**
putBBbatchdata.dsc **11**
putBBdata.dsc **11**

R

Resend Timeout parameter **18**

S

send-external-down **44**
send-external-up **45**
Shutdown Command Notification Function parameter **33**
Start Exchange Data Schedule parameter **18**
start-schedule **46**
Startup Function parameter **28**
stcBBbatchdata-mssql.ssc **13**
stcBBbatchdata-oracle.ssc **13**
stcBBbatchdata-sybase.ssc **13**
stcBBconnect.dsc **11**
stcBBconnectprocess.dsc **11**
stcBBcreatestatus.dsc **11**
stcBBdata-mssql.ssc **13**
stcBBdata-oracle.ssc **13**
stcBBdata-sybase.ssc **13**
stcBBpoll.dsc **11**
stcewbbpts.ctl **10**
stcRoot.ssc **13**
Stop Exchange Data Schedule parameter **17**
stop-schedule **46**

T

tcipip-startup-BB **57**
tcpip-exchange-BB **55**
tcpip-exchange-BB.monk **11**
tcpip-outgoing-BB **56**
tcpip-outgoing-BB.monk **11**
tcpip-startup-BB.monk **11**
tcpip-verify-BB **57**
tcpip-verify-BB.monk **11**

U

Up Timeout parameter **18**
user name **34**

W

WEBeSTP_Header.ssc **13**

Z

Zero Wait Between Successful Exchanges parameter **18**