SeeBeyond™ eBusiness Integration Suite

Batch e*Way Intelligent Adapter User's Guide

Release 4.5.2

SEEBEYOND™

Version 20011221172932.

Contents

Chapter 1

Introduction

This chapter introduces you to SeeBeyond™ Technology Corporation's (SeeBeyond™) Batch e*Way™ Intelligent Adapter, which enables the e*Gate system to exchange data with other network hosts, using the file transfer protocol (FTP).

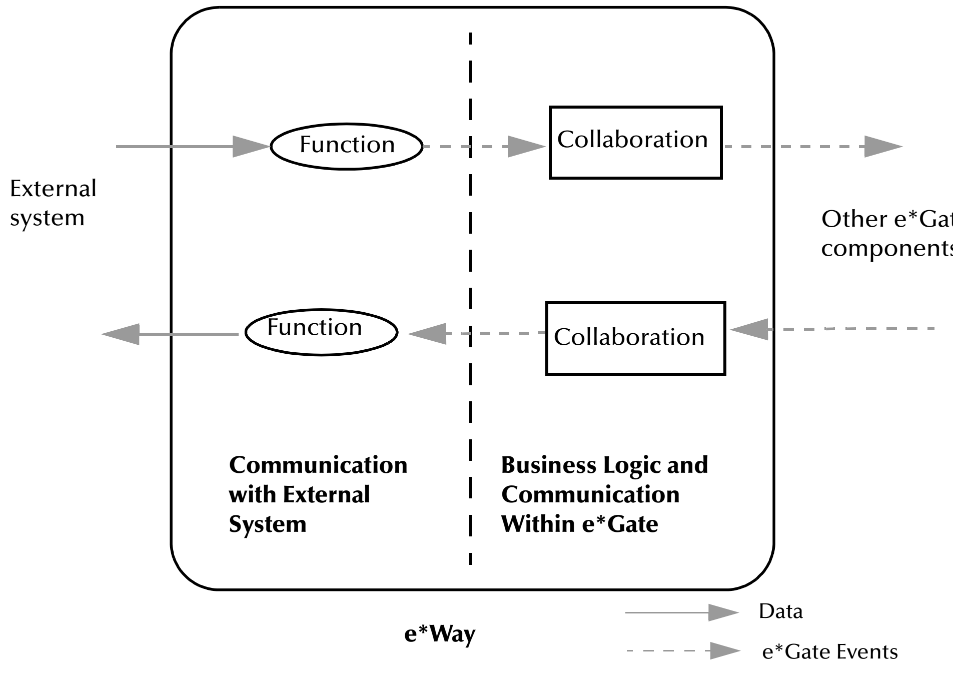
1.1 Overview

This document explains how to install and configure the Batch e*Way. This e*Way is enabled by the Monk programming language.

Note: *The FTP Event Type Definition (ETD) is enabled by the Java programming language. For more information, see* **Chapter 7**.

Figure 1 shows a diagram of how the e*Way operates.

Figure 1 e*Way Internal Architecture



Conceptually, an e*Way is divided into two halves. One half of the e*Way (shown on the left in **Figure 1 on page 11**) handles communication with the external system; the other half manages the Collaborations that process data and subscribe or publish to other e*Gate components.

The communications side 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 SeeBeyond Collaboration Rules Editor or a text editor (such as **Notepad** or UNIX **vi**).

The communications side 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.

The Batch e*Way has the following behavior models:

- Messages are published to the e*Way, then it collects the messages in temporary files until its next scheduled release time. It then sends them out, either as single files per message or multiple messages per single file, depending on configuration.
- The e*Way subscribes to messages and polls an external system based on a schedule and searches for files based on specific criteria. It then retrieves the files that match the criteria, stores them locally, and then reads the records in the files, while simultaneously keeping track of its own progress by maintaining state information in a separate file.
- A Dynamic Configuration is available that requires the use of the flag, **Enable Message configuration** (See **"Enable Message Configuration" on page 71**). If this flag is turned on, the e*Way has a subscription that determines its activity. This subscription is an XML message, with all relevant parameters governing the transfer, including the file to be sent (if it is an outbound transfer).

The Batch e*Way supports standard FTP commands according to RFC-959, for example:

APPE	NOOP	RNTO
CWD	PASS	SITE
DELE	QUIT	STOR
LIST	RETR	TYPE
MKD	RNFR	USER

1.1.1 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the e*Gate system, to have expert-level knowledge of Windows operations and administration, to be thoroughly familiar with Windows-style GUI operations, and to have an understanding of FTP.

1.1.2 Components

The Batch e*Way comprises the following elements:

- **stcewgenericmonk.exe**, the executable component
- Configuration files, which the e*Way Editor uses to define configuration parameters
- Monk function scripts; the scripts themselves are discussed in **Chapter 3**; the functions they call, in **Chapter 6**
- Library files, which provide access to additional Monk application programming interfaces (APIs); the APIs are discussed in **Chapter 6**.

A complete list of installed files appears in **Table 1 on page 18**.

1.1.3 Compatible Systems

Windows Systems: The e*Gate system is fully compliant with both Windows NT and Windows 2000 platforms. When this document references Windows, such statements apply to both Windows platforms.

UNIX Systems: This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions ("/").

Compaq Tru64 Systems: For the purposes of this document, the same instructions for UNIX apply to this system. e*Gate is fully compatible with Compaq Tru64 Version 4.0F, and Version 5.0A.

1.2 Supported Operating Systems

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

- Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Windows NT 4.0 SP6a
- Solaris 2.6, 7, and 8
- AIX 4.3.3
- HP-UX 11.0 and HP-UX 11i
- Compaq Tru64 V4.0F and V5.0A
- Red Hat Linux 6.2
- Japanese Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Japanese Windows NT 4.0 SP6a
- Japanese Solaris 2.6, 7, and 8
- Japanese HP-UX 11.0
- Korean Solaris 8
- Korean Windows 2000, Windows 2000 SP1, and Windows 2000 SP2
- Korean Windows NT 4.0 SP6a

1.3 System Requirements

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

- An e*Gate Participating Host, version 4.5 or later. For AIX operating systems, you need an e*Gate Participating Host, version 4.5.1. or later.
- A TCP/IP network connection.

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

1.4 External System Requirements

This section explains external system requirements for the Batch e*Way.

1.4.1 Client Components

Any client components of the Batch e*Way have their own requirements; see the subject system's documentation for more details.

To communicate with the Batch e*Way, the external system must run an FTP server compliant with RFC-959.

A user name and password granting appropriate access to the FTP server must be available for the e*Way's use.

1.4.2 FTP ETD Requirements

To use the FTP ETD feature, you must have the following third-party packages:

- **Montana\Java\ThirdParty\NetComponents\NetComponents-1.3.8a.jar**
- **Montana\Java\ThirdParty\gnu.regexp\gnu-regexp-1.1.1.jar**

For more information, visit the following Web sites:

http://www.cacas.org/~wes/java/

http://www.savarese.org/oro/software/NetComponents.html

1.4.3 SOCKS Support

SOCKS is a generic proxy protocol for TCP/IP-based networking applications. When an application client needs to connect to an application server, the client connects to a SOCKS proxy server. The proxy server connects to the application server on behalf of the client, and relays data between the client and the application server. For the application server, the proxy server is the client.

The Batch e*Way now supports the SOCKS version 5 Authentication protocol. To enable SOCKS 5 support, the SOCKS server name and port number, as well as the user name and encrypted password, must be specified in the configuration file. Details of these configuration parameters are provided in the chapter **"Configuration" on page 21**.

See also **"ftp-open-host-through-SOCKS" on page 200**.

See also the subsection **"Mode" on page 51**, describing options for data transfer modes to an FTP server.

Chapter 2

Installation

This chapter explains the system requirements and procedures for installing the Batch e*Way.

2.1 Windows NT or Windows 2000

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 Batch e*Way on Windows NT or Windows 2000 systems

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

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

5. After the installation is complete, exit the install utility and launch the Enterprise Manager.
6. In the Component editor, create a new e*Way.

7. Display the new e*Way's properties.
8. On the General tab, under **Executable File**, click **Find**.
9. Select the file **stcgenericmonk.exe**.
10. Click **OK** to close the properties sheet, or continue to configure the e*Way. Configuration parameters are explained in **Chapter 3**.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, Intelligent Queues (IQs), and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, please see the online Help system.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the **Working with e*Ways** user's guide.*

2.2 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 privileges to create files in the e*Gate directory tree.

Note: *Installation instructions for Compaq Tru64 Version 4.0f are the same as those for UNIX.*

2.2.2 Installation Procedure

To install the Batch 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/setup
   ```
4. Start the installation script by typing:

   ```
   setup.sh
   ```
5. A menu of options appear. Select the **e*Gate Addon Applications** option. Then, follow any additional on-screen directions.

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

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

6. After installation is complete, exit the installation utility and launch the Enterprise Manager.
7. In the Component editor, create a new e*Way.
8. Display the new e*Way's properties.
9. On the General tab, under **Executable File**, click **Find**.
10. Select the file **stcewgenericmonk.exe**.
11. Click **OK** to close the properties sheet, or continue to configure the e*Way. Configuration parameters are discussed in **Chapter 3**.

Note: *Once you have installed and configured this e*Way, you must incorporate it into a schema by defining and associating the appropriate Collaborations, Collaboration Rules, IQs, and Event Types before this e*Way can perform its intended functions. For more information about any of these procedures, please see the online Help system.*

*For more information about configuring e*Ways or how to use the e*Way Editor, see the **e*Gate Integrator User's Guide**.*

2.3 Files/Directories Created by the Installation

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

Table 1 Files Created by the Installation

e*Gate Directory	File(s)
bin\	stcewgenericmonk.exe stc_ewftp.dll stc_monkfilesys.dll
configs\stcewgenericmonk\	batch.def
monk_library\batch\	batch-ack.monk batch-dynamic-init.monk batch-dynamic-proc-out.monk batch-dynamic-send-to-egate.monk batch-exchange-data.monk batch-exchange-utils.monk batch-ext-connect.monk

Table 1 Files Created by the Installation (Continued)

e*Gate Directory	File(s)
	batch-ext-shutdown.monk batch-ext-verify.monk batch-fetch-files-from-remote.monk batch-fetch-named-files.monk batch-init.monk batch-nak.monk batch-persist.monk batch-post-transfer.monk batch-proc-out.monk batch-regular-init.monk batch-regular-proc-out batch-send-path-file.monk batch-shutdown-notify.monk batch-startup.monk batch-utils.monk batch-validate-params.monk file-ext-connect.monk file-ext-shutdown.monk file-ext-verify.monk file-fetch.monk file-fetch-path.monk file-init.monk file-remote-path-list.monk file-remote-post-transfer.monk file-rmt-list.monk file-rmt-post-transfer.monk file-send.monk file-send-path-file.monk file-startup.monk file-vaildate-params.monk ftp-connect.monk ftp-disconnect.monk ftp-ext-connect.monk ftp-ext-shutdown.monk ftp-ext-verify.monk ftp-fetch.monk ftp-fetch-path.monk ftp-init.monk ftp-pre-post-commands.monk ftp-remote-path-list.monk ftp-remote-post-transfer.monk ftp-rmt-list.monk ftp-rmt-post-transfer.monk ftp-send.monk ftp-send-path-file.monk ftp-startup.monk ftp-validate-params.monk local-post-transfer.monk

Table 1 Files Created by the Installation (Continued)

e*Gate Directory	File(s)
eGate\client\monk_scripts\common	batch_eway_data.jar batch_eway_error.jar batch_eway_order.jar batch_eway_data.xsc batch_eway_error.xsc batch_eway_order.xsc
\eGate\client\etd\batchclient\	FtpFileETD.xsc

Chapter 3

Configuration

This chapter explains the parameters used to configure the Batch e*Way.

3.1 e*Way Configuration Parameters

Set the e*Way configuration parameters, using the e*Way Editor graphical user interface (GUI) available through the e*Gate Enterprise Manager.

To change e*Way configuration parameters:

1. In the Enterprise Manager's Component editor, select the e*Way you want to configure and display its properties.
2. Under **Configuration File**, click **New** to create a new file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file.
3. In the **Additional Command Line Arguments** box, type any additional command-line arguments that the e*Way may require, taking care to insert them *at the end* of the existing command-line string.

Caution: *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 GUI, see the e*Way Editor's online Help or the *e*Gate Integrator User's Guide*. The e*Way's configuration parameters are organized into the following sections:

- **General Settings** on page 22
- **Communication Setup** on page 23
- **Monk Configuration** on page 26
- **External Host Setup** on page 40
- **Subscribe to External** on page 42
- **Publish to External** on page 45
- **Sequence Numbering** on page 49
- **Recourse Action** on page 50
- **FTP** on page 50
- **SOCKS** on page 52
- **Dynamic Configuration** on page 71

3.1.1 General Settings

The General Settings control basic operational parameters.

Journal File Name

Description

Specifies the name of the journal file.

Required Values

A valid file name, including the absolute path (for example, **c:\temp\filename.txt**). 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**)
- When its receipt is due to an external error, but **Forward External Errors** is set to **No**. (See **"Forward External Errors" on page 23** for more information.)

Max Resends Per Message

Description

Specifies the number of times the e*Way will attempt to resend an Event (message) 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 Events (messages) 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 **“Schedule-driven Data Exchange Functions” on page 30** for more information about how the e*Way uses this function.

Required Values

Yes or **No**. The default value, **Yes**, specifies that error messages are to be forwarded.

3.1.2 Communication Setup

The Communication Setup parameters control the schedule by which the e*Way obtains data from the external system. These parameters are affected by the **Dynamic Configuration** section. See **Table 5 on page 72**.

Note: *The schedule (that is, timetable) you set using the e*Way’s properties in the Enterprise 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 (see **“Exchange Data with External Function” on page 36**).

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 functions:

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

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

See **"Exchange Data with External Function" on page 36**, **"Exchange Data Interval" on page 25**, and **"Stop Exchange Data Schedule" on page 24** for more information. See also, **"Exchange-if-in-window-on-startup" on page 26**.

Additional Information

When the schedule starts, the e*Way determines whether it is waiting to send a positive or negative acknowledgment to the external system (using the **Positive Acknowledgment** and **Negative Acknowledgment** functions) and whether the connection to the external system is active.

If no positive or negative acknowledgements are 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.

Also, see **start-schedule** on page 90.

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).

Also, see **stop-schedule** on page 91.

Exchange Data Interval

Description

Specifies the number of seconds the e*Way waits between calls to the **Exchange Data with External** function (see **"Exchange Data with External Function" on page 36**). If the **Start Exchange Data Schedule** and **Stop Exchange Data Schedule** parameters have been set to create a scheduled data-exchange window, then this interval only operates during this window. If these parameters have not been set to create such a window, then the **Exchange Data Interval** operates on a continuous basis, in conjunction with the **Exchange Data with External** function.

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 0 (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 25** and **"Stop Exchange Data Schedule" on page 24** 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 37** 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 38** 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 **Yes**.

See **"Exchange Data with External Function" on page 36** for more information.

Exchange-if-in-window-on-startup

If this parameter is set to **Yes**, and the e*Way starts within an exchange data window, the e*Way immediately invokes the **Exchange Data with External Function**.

Required Values

Yes or **No**. The default is **No**.

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. These parameters are affected by the **Dynamic Configuration** section. See **Table 5 on page 72**.

Operational Details

The Monk functions in the communications side of the e*Way fall into the groups shown in **Table 2 on page 27**.

Table 2 Monk Communications Functions

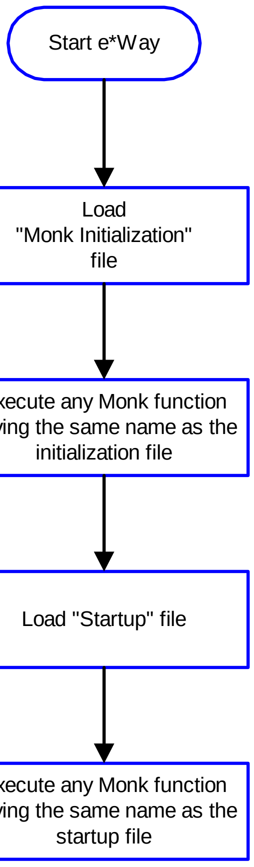
Type of Operation	Name
Initialization	**Startup Function** on page 35 (also see **Monk Environment Initialization File** on page 34)
Connection	**External Connection Establishment Function** on page 37 **External Connection Verification Function** on page 38 **External Connection Shutdown Function** on page 38
Schedule-driven data exchange	**Exchange Data with External Function** on page 36 **Positive Acknowledgment Function** on page 38 **Negative Acknowledgment Function** on page 39
Shutdown	**Shutdown Command Notification Function** on page 40
Event-driven data exchange	**Process Outgoing Message Function** on page 35

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

Initialization Functions

Figure 2 on page 28 illustrates how the e*Way executes its initialization functions.

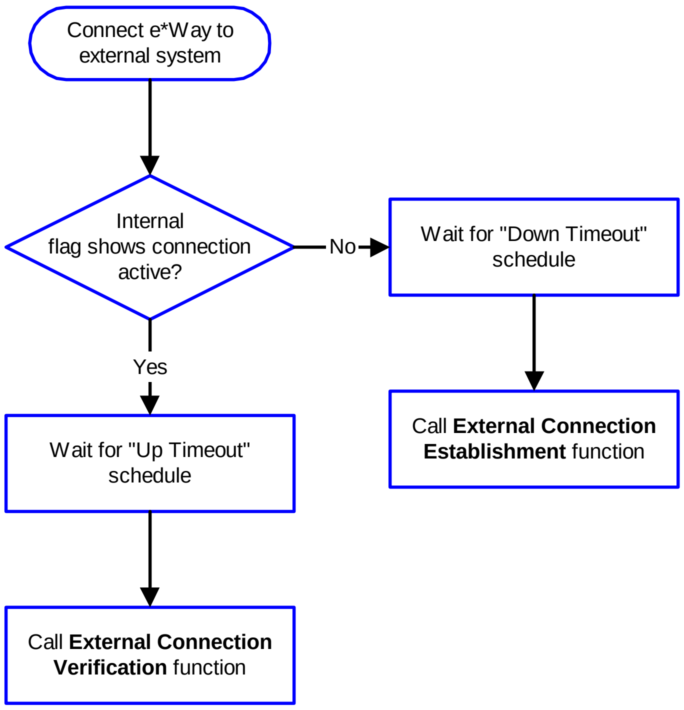
Figure 2 Initialization Functions


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

Connection Functions

Figure 3 on page 29 illustrates how the e*Way executes the connection establishment and verification functions.

Figure 3 Connection Establishment and Verification Functions

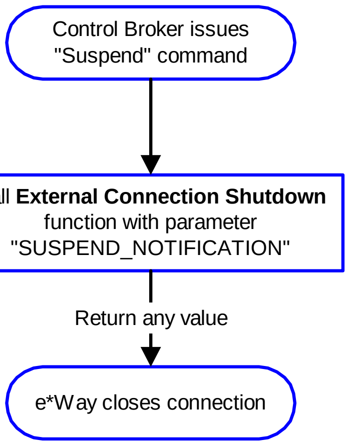

Connect e*Way to external system
Internal flag shows connection active?
No
Wait for "Down Timeout" schedule
Call **External Connection Establishment** function
Yes
Wait for "Up Timeout" schedule
Call **External Connection Verification** function

Note: *The e*Way selects the connection function based on an internal "up/down" flag rather than a poll to the external system. See* **Figure 5 on page 31** *and* **Figure 7 on page 33** *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 88** *and* **"send-external-down" on page 87** *for more information.*

Figure 4 on page 30 illustrates how the e*Way executes its connection shutdown function.

Figure 4 Connection Shutdown Functions


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

Schedule-driven Data Exchange Functions

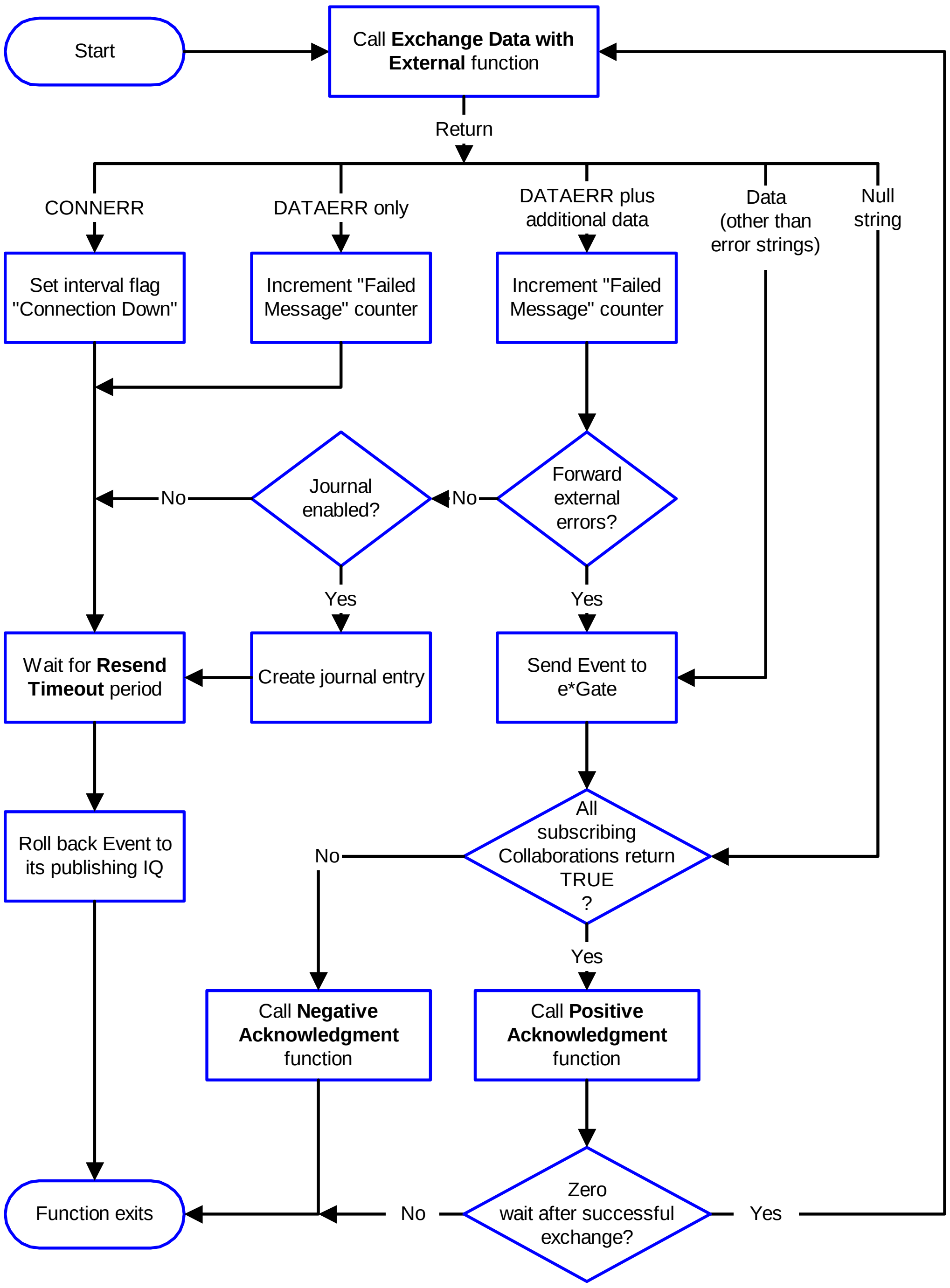
Figure 5 on page 31 illustrates how the e*Way performs schedule-driven data exchange using the **Exchange Data with External** function. The **Positive Acknowledgment Function** and **Negative Acknowledgment** function are also called during this process.

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

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

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

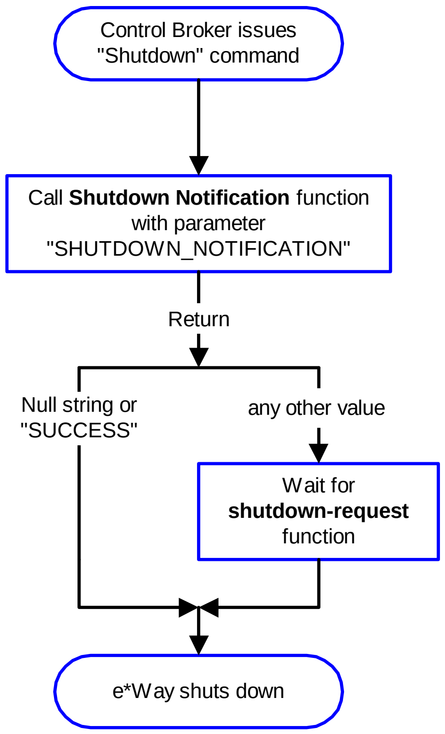
Figure 5 Schedule-driven Data Exchange Functions


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

Shutdown Functions

Figure 6 on page 32 illustrates how the e*Way implements the shutdown request function.

Figure 6 Shutdown Functions


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

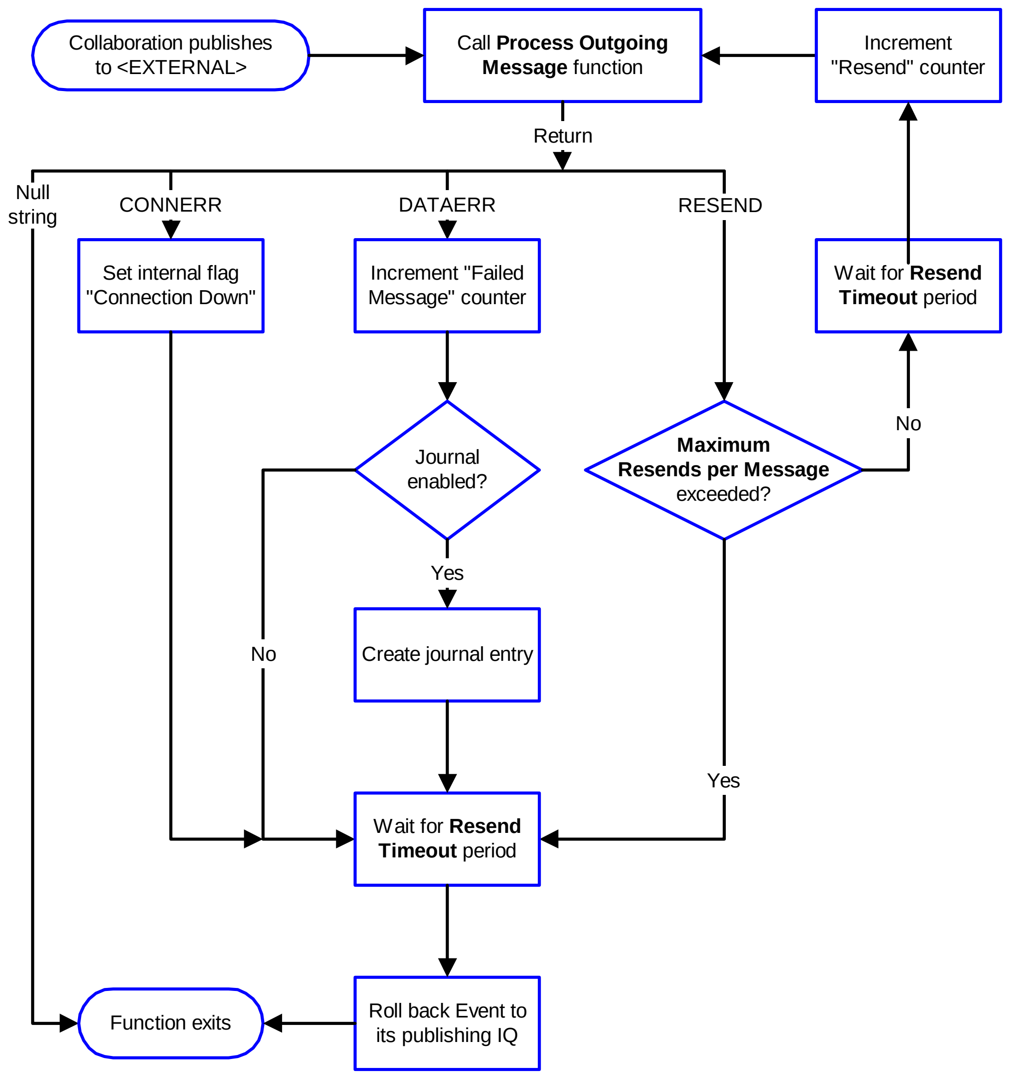
Event-driven Data Exchange Functions

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 on page 33 illustrates event-driven data-exchange using the **Process Outgoing Message Function**.

Figure 7 Event-driven Data Exchange Functions


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

How to Specify Function/File Names

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

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

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 path, 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 path name, or a series of paths separated by semicolons.

Additional Information

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

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

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

This parameter is optional and may be left blank.

Monk Environment Initialization File

Specifies a file that contains environment initialization functions, which 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 file name within the "load path", or file name 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 34** for more information about the "load path."

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 28**).

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.

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 28**). 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.*

Additional Information

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

The e*Way invokes this function when one of its Collaborations publishes an Event to an <EXTERNAL> destination (as specified within the Enterprise Manager). The function returns one of the following (see **Figure 7 on page 33** 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.
- **Any other string**: If a string other than the preceding 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 85** 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.

Additional Information

The function accepts no input and must return a string (see **Figure 5 on page 31** 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 31** 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 positively or negatively acknowledged (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 90** 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.*

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.

Additional Information

The function accepts no input and must return a string as follows:

- **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.

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.

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 reestablished, 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.

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 reestablished, 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 is called when the e*Way receives a **shutdown** command from the Control Broker. This parameter is optional.

Required Values

The name of a Monk function.

Additional Information

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

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

- **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 89**).

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

3.1.4 External Host Setup

The **External Host Setup** parameters describe the FTP server to which the e*Way will connect.

Note: *These parameters may be overridden depending on how parameters in the **Dynamic Configuration** section are set. See* **Table 5 on page 72**.

Host Type

Description

Specifies the operating system of the FTP Server. The e*Way uses this parameter when analyzing the output of the FTP **list** command.

Required Values

The default is **UNIX**. Use any one of following supported host types:

Host Type	Directory structure
UNIX	/dir1/dir2/file.ext
NT	\dir1\dir2\file.ext
VMS	disk1:[dir1.dir2]file.ext;1
MVS	dir1.dir2(file)
MVS PS	dir1.dir2.file

Host Type	Directory structure
MVS GDG	dir1.dir2.file(version)
AS400	dir1/file.ext
AS400-UNIX	/dir1/dir2/file.ext
HCLFTPD	/dir1/dir2/file.ext
HCLFTPD	/dir1/dir2/file.ext
MSFTPD	/dir1/dir2/file.ext
VM/ESA	file.ext

External Host Name

Description

Specifies the host name of the FTP server.

Required Values

A valid host name. The default is **localhost**.

User Name

Description

Specifies the user name the e*Way will use when gaining access to the FTP server.

Required Values

A valid user name. The default is **anonymous**.

Encrypted Password

Description

Specifies the password the e*Way will use when gaining access to the FTP server.

Required Values

The password appropriate for the user name specified earlier. First enter the user name then enter the password in cleartext; the e*Way editor will store the password encrypted. The encrypted form of the password is based on the combined username and the password in cleartext. Therefore, an environment variable can not be used in lieu of the username.

File Transfer Method

Description

Selects whether files are transferred via FTP protocol or by a simple file-copy operation.

Required Values

FTP or **File Copy**. The default is **FTP**.

Additional Information

The **File Copy** parameter can be used when transferring files between physically different systems across NFS mounts.

3.1.5 Subscribe to External

The **Subscribe to External** parameters control how the e*Way retrieves files from an external system. Note that when you are archiving a local file, the archive destination must be on the same volume as the source.

Note: *These parameters may be overridden or ignored altogether depending on how parameters in the **Dynamic Configuration** section are set. See* **Table 5 on page 72***.*

Additional Information

When you are using the Batch e*Way's **Subscribe to External**-related features to retrieve files from external systems, keep the following facts in mind:

- The FTP process can copy an open file from the external system and into e*Gate. If the file is currently being modified and correct results depend on the completed file, an unready file could be copied into the e*Gate system. To avoid this problem, you can set up external files to be copied using a signal to tell you whether the file is open. For example, you can have the system try to rename the file first, and if the rename operation fails, the file is not ready for use and not copied.
- Keep in mind that the FTP process copies files in the directory list order. You can verify this operation by checking the **persist.dat** file. You can modify the list command in this file to change the order.

Remote Directory Name

Description

Specifies a directory path on the external system from which the e*Way retrieves files.

Required Values

Leave this parameter blank to use the default directory assigned to the user name by which the e*Way logs in (most often, the user's home directory). Otherwise, specify an absolute path. The path must exist on the FTP server's system. There is no default specified.

Remote File Regexp

Description

Specifies a regular expression that describes files to be retrieved.

Required Values

A valid regular expression. See below for restrictions on special characters.

Additional Information

Special characters can be used, which are expanded by the e*Way before the file is transmitted. See **"Using Special Characters" on page 53** for details.

Record Type

Description

Specifies the record structure of the files being retrieved.

Required Values

One of **Delimited**, **Fixed**, or **Single Record**. The default is **Delimited.**

Additional Information

- For delimited files, the delimiter characters are defined by the **Record Delimiter** parameter
- For fixed-record files, the record size is defined by the **Record Size** parameter
- For single-record files, it is recommended that you use message sequencing to prevent any messages from being overwritten (see **"Sequence Numbering" on page 49** for more information)

Record Delimiter

Description

Specifies the record delimiter in delimited files.

Required Values

A string. The delimiter can be entered in ASCII, escaped ASCII, octal, or hex. The default is **\n** (new line).

Additional Information

The delimiter is stripped and is not queued as part of the record data.

Delimiter on Last Record

Description

Specifies whether the last record in a delimited file is terminated by a delimiter.

Required Values

Yes or **No**. The default is **Yes**.

Additional Information

This parameter is only used when **Record Type** is set to Delimited.

Record Size

Description

Specifies the record length for fixed-record files, in bytes.

Required Values

A positive integer between 1 and 214,783,647.

Remote Command After Transfer

Description

Specifies the command that the e*Way executes on the external system after a successful file transfer.

Required Values

One of **Rename**, **Archive**, or **None**. The default is **None**.

Additional Information

The Archive command moves the file to the directory specified in the **Remote Rename or Archive Name** (see that section) parameter.

The **Rename** and **Archive** values may not be available on all systems because they rely on the FTP command **RNFR** being available on the external system. If the external system does not support **RNFR**, these commands do not work.

If you are receiving multiple files, using **Rename** overwrites the file each time another file is transferred. Do not use **Rename** unless you are providing your own handler for manipulating the file name (see the **Remote Rename or Archive Name** section).

Note: *MVS does not permit the renaming of partitioned data sets into different partitioned data sets. Therefore, neither the Remote Rename nor Archive Name commands are supported on MVS systems.*

Remote Rename or Archive Name

Description

Depending on the value of **Remote Command After Transfer**, the parameter specifies either the name to which to rename the external file (for **Rename**) or the directory in which to archive the external file (for **Archive**).

Required Values

A file name or path name. There is no default specified.

Additional Information

Special characters can be used, which are expanded by the e*Way before the file is transmitted. See **“Using Special Characters” on page 53** for details. The expansion of any special character is carried out each time this parameter is used.

Note: *If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the e*Way interprets the back slash as a special character and not a path separator. For example, use* ***c:/temp/dir*** *for that path location,* **not** ***c:\temp\dir****.*

Local Command After Transfer

Description

Specifies the action to be performed on the temporary file after all the records in it have been queued.

Required Values

One of **Delete** or **Archive**. The default is **Delete**.

Local Archive Directory

Description

Specifies the directory in which to archive the file.

Required Values

A path to a directory. There is no default specified.

Additional Information

The local file must be removed from the working directory by archiving.

Special characters can be used, which are expanded by the e*Way before the file is transmitted. See **"Using Special Characters" on page 53** for details. The expansion of any special character is carried out each time this parameter is used.

Note: *If you are entering a path, use the forward slash (/) instead of the back slash (\) because the e*Way interprets the back slash as a special character and not a path separator. For example, use* ***c:/temp/dir*** *for that path location,* **not** ***c:\temp\dir****.*

3.1.6 Publish to External

The **Publish to External** parameters control how the e*Way publishes data to an external system.

Note: *These parameters may be overridden or ignored altogether depending on how parameters in the* ***Dynamic Configuration*** *section are set. See* **Table 5 on page 72***.*

Remote Directory Name

Description

Specifies a path to the directory on the external system to which the e*Way will transfer files.

Required Values

Leave this parameter blank to use the default directory assigned to the user name by which the e*Way will log in (most often, the user's home directory). Otherwise, specify an absolute path. The path must exist on the FTP server's system. There is no default specified.

Remote File Name

Description

Specifies the file name on the external system to be used for the file transfer.

Required Values

Any valid file name, as an absolute path. A file name must be specified; do not specify a directory name.

Additional Information

Special characters can be used which are expanded by the e*Way before the file is transmitted. See **"Using Special Characters" on page 53** for details.

Append or Overwrite when Transferring Files

Description

Specifies whether to append the records in the file being transferred to the existing file on the external system, or to overwrite the existing file on the external system with the file being transferred.

Required Values

One of **Append** or **Overwrite**. The default is **Append**.

Record Type

Description

Specifies the record structure of the files being transferred to the external system.

Required Values

One of **Delimited**, **Fixed**, or **Single Record**. The default is **Delimited.**

Additional Information

- For delimited files, the delimiter characters are defined by the **Record Delimiter** parameter
- For fixed-record files, the record size is defined by the **Record Size** parameter
- For single-record files, it is recommended that you use message sequencing to prevent any messages from being overwritten (see **"Sequence Numbering" on page 49** for more information)

Record Delimiter

Description

Specifies the record delimiter in delimited files.

Required Values

A string. The delimiter can be entered in ASCII, escaped ASCII, octal, or hex. The default is **\n** (new line).

Delimiter on Last Record

Description

Specifies whether the last record in a delimited file is terminated by a delimiter.

Required Values

Yes or **No**. The default is **Yes**.

Additional Information

This parameter is only used when **Record Type** is set to Delimited.

Record Size

Description

Specifies the record length for fixed-record files, in bytes.

Required Values

A positive integer. The range is between 1 and 214,783,647.

Remote Command After Transfer

Description

Specifies the command that the e*Way executes on the external system after a successful file transfer.

Required Values

One of **Rename**, **Archive**, or **None**. The default is **None**.

Additional Information

The **Archive** command moves the file to the directory specified in the **Remote Rename or Archive Name** (see that section) parameter.

The **Rename** and **Archive** values may not be available on all systems because they rely on the FTP command **RNFR** being available on the external system. If the external system does not support **RNFR**, these commands do not work.

If you are receiving multiple files, using **Rename** overwrites the file each time another file is transferred. Do not use **Rename** unless you are providing your own handler for manipulating the file name (see the **Remote Rename or Archive Name** section).

Note: *MVS does not permit the renaming of partitioned data sets into different partitioned data sets. Therefore, neither the **Remote Rename** nor **Archive Name** commands are supported on MVS systems.*

Remote Rename or Archive Name

Description

Depending on the value of **Remote Command After Transfer**, the parameter specifies either the name to which to rename the external file (for **Rename**) or the directory in which to archive the external file (for **Archive**).

Required Values

A file name or path. There is no default specified.

Additional Information

Special characters can be used, which are expanded by the e*Way before the file is transmitted. See **"Using Special Characters" on page 53** for details. The expansion of any special character is carried out each time this parameter is used.

Note: *If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the e*Way interprets the back slash as a special character and not a path separator. For example, use **c:/temp/dir** for that path location,* **not** ***c:\temp\dir**.*

Local Command After Transfer

Description

Specifies the action to be performed on the temporary file after all the records in it have been queued.

Required Values

One of **Delete** or **Archive**. The default is **Delete**.

Local Archive Directory

Description

Specifies the directory in which to archive the file.

Required Values

A file name or path. There is no default specified.

Additional Information

The local file must be removed from the working directory by archiving.

Special characters can be used, which are expanded by the e*Way before the file is transmitted. See **"Using Special Characters" on page 53** for details. The expansion of any special character is carried out each time this parameter is used.

Note: *If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the e*Way interprets the back slash as a special character and not a path separator. For example, use* ***c:/temp/dir*** *for that path location,* **not** ***c:\temp\dir****.*

3.1.7 Sequence Numbering

The **Sequence Numbering** parameters determine how to use sequence numbers to generate file names. These parameters are affected by the **Dynamic Configuration** section. See **Table 5 on page 72**.

If sequence numbering is used, the file name must contain a single occurrence of a special format string that designates the sequence number (see **"Sequence Numbering" on page 55**). The sequence number is incremented by one after each successful file "get."

Note: *When composing external file names, do not use wildcard characters immediately before or after the special format string because these may cause file name expansion ambiguities. Wild cards may not be used in the name of a sending file.*

Starting Sequence Number

Description

Specifies the starting sequence number used if there is no number from a previous run (if there is, the previous number will be used).

Required Values

A non-negative integer. The default range is from 0 to 1, but you may change the upper limit of the range. No default is specified.

Additional Information

To change the default range in the e*Way Editor, simply change the value in the **To** box. You will only be able to add a starting value higher than 1 after you change the limit.

The number must be less than the **Max Sequence Number**. When the **Max Sequence Number** is reached, the current sequence number rolls back to this parameter.

Max Sequence Number

Description

Specifies the last sequence number to be used before rolling over to the **Starting Sequence Number**.

Required Values

A positive integer. The default range is between 1 and 214,783,647. No default is specified.

Additional Information

This number must be greater than the **Starting Sequence Number**.

3.1.8 Recourse Action

The **Recourse Action** parameters determine the action to be taken if the FTP transfer fails. This action will depend on the interface to the external system and the data contained in the files. The default action is to shut down the e*Way, which we recommend as the safest course of action. These parameters are affected by the **Dynamic Configuration** section. See **Table 5 on page 72**.

Action on Fetch Failure

Description

Specifies the recourse action to be taken if the FTP operation failed when retrieving a file from the external system.

Required Values

One of **Exit**, **Skip File**, or **Next Schedule**. The default is **Exit**.

- **Exit**: Shuts down the e*Way immediately.
- **Skip File:** Ignores the file that could not be retrieved, leaving it on the external server. The e*Way will retry the retrieval on the next scheduled attempt.
- **Next Schedule**: Stops the e*Way from retrieving more files until the next schedule. However, any files that are already retrieved will be processed.

Action on Send Failure

Specifies the recourse action to be taken if the FTP operation failed when sending a file to the external system.

Required Values

One of **Exit**, **Skip File**, or **Next Schedule**. The default is **Exit**.

- **Exit**: Shuts down the e*Way immediately.
- **Skip File:** Ignores the file that could not be retrieved, leaving it on the external server. The e*Way will try the retrieval again on the next scheduled attempt.
- **Next Schedule**: Stops the e*Way from retrieving more files until the next schedule. However, any files that are already retrieved will be processed.

3.1.9 FTP

This section contains the parameters for communicating with a FTP server.

Server Port

Description

Specifies the port number to use for connection to the FTP server.

Required Values

A integer from **0** through **100000**. Default is **21**.

Mode

Description

Specifies the mode to use for the transfer of data to, or from, the FTP server.

Required Values

A, **B**, or **E** where A = ASCII, B = BINARY (or Image) and E = EBCDIC. The default is **A**.

Note: *The **E** value is supported only within AIX systems. To transport EBCDIC data to an ASCII-based system (UNIX or Windows), you must use the **ebcdic->ascii** Monk function. See the Monk Developer's Reference Guide.*

Additional Information

The mode selected will produce different results, depending on the type of data transferred, and the types of systems involved. The table below illustrates the possible different configurations of systems, data, and modes, with the corresponding results.

Table 3 Results of Modes Under Different Configurations

Configuration	Mode	Results
Batch e*Way on ASCII machine retrieving data from an EBCDIC machine.	ASCII	Data converts to ASCII which can be read on ASCII machine.
	EBCDIC	Data converts to ASCII which can be read on ASCII machine.
	Binary	Data remains in EBCDIC.
Batch e*Way on ASCII machine retrieving data from an ASCII machine.	ASCII	Data remains in ASCII.
	EBCDIC	Data converts to unreadable format.
	Binary	Data will be in ASCII.

Pretransfer Commands

Description

Specifies a set of FTP commands to use before a FTP file transfer. The command delimiter is ';'. For example:

```
SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE
PRI=5;SITE SEC=5
```

Posttransfer Commands

Description

Specifies a set of FTP commands to use after a FTP file transfer. The command delimiter is ';'.

3.1.10 SOCKS

This section contains the parameters the e*Way will use when it connects to a SOCKS5 server. These parameters are affected by the **Dynamic Configuration** section. See **Table 5 on page 72**.

Server Host Name

Description

Specifies the SOCKS server name to use to communicate with a SOCKS5 server.

Required Values

A string indicating the name of the SOCKS server.

Server Port

Description

Specifies the port number to use on the SOCKS server for connection. A non-negative integer implies that the e*Way will be connecting to a SOCKS server. Therefore, leave this parameter empty if the e*Way will not be connecting to a SOCKS server.

Required Values

An integer from **0** through **100000**. Default is **0**.

Leave this field blank if the e*Way will not connect to a SOCKS server. Otherwise, enter a non-negative integer in the range 0 through 100,000.

Note: *Check with your System Administrator to verify the availability and necessity of configuring the SOCKS server.*

Method

Description

Specifies the SOCKS5 method-dependent subnegotiation, and determines whether a user name and encrypted password are required.

Required Values

No Authentication (the default) or **User/Password**. These are the only two methods that the Batch e*Way supports. **No Authentication** indicates that neither the User Name nor a password are required.

If **User/Password** is selected, specifying the values for these two parameters is required:

- User Name
- Encrypted Password

User Name

Description

When User/Password is selected for the **Method** parameter, the user name specified here (and the Encrypted Password) is used for authentication with the SOCKS server.

Required Values

String value of the user name. The default value is **anonymous**.

Encrypted Password

Description

When User/Password is selected for the **Method** parameter, the password specified here is used (with the User Name) for authentication with SOCKS server.

Required Values

String value of the Password.

Note: *The Batch e*Way does not support* Kerberos *authentication protocol.*

3.1.11 Using Special Characters

Directory and file names can contain special characters. In most cases, these characters are undesirable for directory names and for outbound file names, but are not prohibited.

Literal Characters

If a literal character is required, the special character must be preceded by a backslash (\), for example, * for the asterisk character.

Wildcard Expansion

The wildcard characters can be used when retrieving files. After the Batch e*Way requests and receives a list from remote directory, it filters the list using **Remote File Regexp**.

(For more information, see **"Remote File Regexp" on page 42**. See also, **"batch-fetch-files-from-remote" on page 104**, **"file-remote-path-list" on page 154**, **"ftp-remote-path-list" on page 203**.)

The filtering is Batch e*Way implementation-specific. These wildcard characters are:

Wildcard Character	Description
*	Zero or more matches of the preceding character
+	One or more matches of the preceding character
.	Any single character

Hexadecimal and Octal

To insert a hexadecimal value, use the notation **\x*NN*** where ***NN*** is a hexadecimal number.

To insert an octal value, use the notation **\o*NNN*** where ***N*** is a valid octal digit.

Unprintable Characters

A number of common characters have a well-defined representation. These characters are frequently used as record delimiters, especially \n and \r. They are:

Special Character	Description
\0	Null character (\x00)
\a	Audible bell character (\x07)
\b	Backspace (\x08)
\f	Form feed (\x0c)
\n	New line (\x0a)
\r	Line feed (\x0d)
\t	Tab (\x09)
\c	Vertical tab (\x0b)

Date and Time Expansion

The following expansions relate to those provided by the C **strftime()** function (the expansion is site-specific):

Special Character	Description
%a	Abbreviated weekday
%A	Full weekday
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation (location-specific)
%d	Day of month (01-31)
%H	Hour (00-23)
%I	Hour (01-12)
%j	Day of the year (001-366)
%m	Month (01-12)
%M	Minute (00-59)
%p	AM or PM

Special Character	Description (Continued)
%S	Seconds (00-61) **Note:** Seconds may be as high as 61 if there are leap seconds to be accounted for.
%U	Week number, starting on the first Sunday
%W	Week number, starting on the first Monday
%w	Day of the week, (Sunday=0)
%x	Date representation (location-specific)
%y	Year (00-99)
%Y	Year including century
%Z	Time zone

Sequence Numbering

Special characters used for sequence numbering are

Special Character	Description
%#	Sequence number
%5#	Sequence number padded to five places

File Name Replacement

Use the special character **%f** if you need a working file name replacement. For example, if the original working file name is **abcd**, **%f.txt** stands for **abcd.txt**.

3.2 FTP Heuristics

The FTP heuristics are a set of parameters which the e*Way uses (via the FTP **.dll** file) to interact with external FTP daemons on a system-specific level. The primary functions create and parse both path and file names in the style required by the external system.

You do not normally need to change any of the FTP heuristics, since the default parameters have been set up for the most commonly used operating systems. This section is provided as a reference should any changes be necessary to accommodate your site's requirements.

The FTP heuristics are stored in the file **FtpHeuristics.cfg**, which are downloaded from the e*Gate Registry when the e*Way invokes the Monk function **ftp-init** (see ftp-init on page 167 for more information), and configuration changes can be made using the e*Gate Enterprise Manager's e*Way Editor GUI.

3.2.1 Operating System or File Type Selection

Each operating system defined within the FTP heuristics file sets the same parameters, described below. In the e*Way Editor GUI, the operating system is selected by the **Goto Selection** list.

FTP Heuristics support the following file types:

- UNIX
- Windows NT 3.5
- Windows NT 4.0
- HCLFTPD 5.1
- HCLFTPD 6.0.1.3
- VMS
- MSFTPD 2.0
- MVS (Partitioned Data Set (PDS))
- MVS PS (Physical Sequential)
- MVS GDG (Generation Data Group)
- VM/ESA
- AS400
- AS400-UNIX
- MPE

The FTP heuristic functions used for communication with MVS PDS, MVS PS, and MVS GDG for the Batch e*Way are designed for FTP servers (at the mainframe) that use IBM IP stack.

Therefore, when you use FTP to an MVS, MVS PS, or MVS GDG file system on a Mainframe, you need to make sure that the FTP server is using IBM IP stack. If any other IP stack is in place, the FTP heuristic functions do not work or can require modification.

Note: *The following MONK function APIs are not supported on heuristics MVS GDG:*
ftp-rename
ftp-rename-path
ftp-archive
ftp-archive-path
ftp-delete
ftp-delete-path

For more information, see **Advanced FTP Functions** *on page 172*

3.2.2 Configuration Parameters

The section explains the configuration parameters for FTP heuristics feature of the Batch e*Way.

Commands Supported by FTP Server

Description

Specifies the commands that the FTP server on the given host supports.

Required Values

One or more FTP commands as selected from the list.

Header Lines To Skip

Description

Specifies the number of beginning lines from a **LIST** command to be considered as a potential header (subject to the **Header Indication Regex Expression** configuration parameter, discussed below) and skipped.

Required Values

A non-negative integer. Enter zero if there are no headers.

Additional Information

In the example below, the line “total 6” comprises a one-line header.

```
total 6
-rw-r-----   1 ed        usr           110 Apr 15 13:43 AAA
-rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
```

Header Indication Regex Expression

Description

Specifies a regular expression used to help identify lines which comprise the header in the output of a LIST command. All the declared lines of the header (see **Header Lines To Skip**, above) must match the regular expression.

Required Values

A regular expression. The default varies based on the FTP server’s operating system. If there is no reliable way of identifying the header lines in the **LIST** command’s output, leave this parameter undefined.

Additional Information

The regular expression “^ *total” indicates that each line in the header starts with “total,” possibly preceded by blanks. For example,

```
total 6
-rw-r-----   1 ed        usr           110 Apr 15 13:43 AAA
-rw-r--r--   1 ed        usr           110 Apr 15 13:33 aaa
```

If the regular expression is undefined, then the header is solely determined by the value of the configuration parameter **Header Lines To Skip**.

Trailer Lines To Skip

Definition

Specifies the number of ending lines from a **LIST** command that are to be considered as a potential Trailer (subject to the **Trailer Indication Regex Expression**) and skipped.

Required Values

A non-negative integer. Enter zero if there are no trailers.

Trailer Indication Regex Expression

Definition

Specifies the regular expression used to help identify lines which comprise the trailer in the output of a **LIST** command. All the declared lines of the trailer (see **Trailer Lines To Skip**) must match the regular expression.

Required Values

A regular expression. If there is no reliable way of identifying the trailer lines in the **LIST** output, then leave this parameter undefined.

Additional Information

If the regular expression is undefined, then the header is determined solely by the value of the **Trailer Lines To Skip** configuration parameter.

Directory Indication Regex Expression

Definition

Specifies a regular expression used to identify external directories in the output of a **LIST** command. Directories cannot be retrieved and must be filtered out of the file list.

Required Values

A regular expression. If there is no reliable way of identifying the trailer lines in the **LIST** output, then leave this parameter undefined.

Additional Information

The regular expression “^ *d” specifies that a directory is indicated by a line starting with the lowercase ‘d,’ possibly preceded by blanks. For example,

```
drwxr-xr-x   2 ed    usr     2048 Apr 17 17:43 public_html
```

File Link Real Data Available

Definition

Specifies whether a file may be a file link (a pointer to a file) on those operating systems whereon an FTP server will return the data for the real file as opposed to the content of the link itself.

Required Values

Yes or **No**.

File Link Indication Regex Expression

Definition

Specifies a regular expression that identifies external file links in the output of a **LIST** command. File links are pointers to the real file and usually have some visual symbol, such as ->, mixed in with the file name in the output of the **LIST** command. Only the link name is desired within the returned list.

Required Values

A regular expression. If there is no reliable way of identifying a file link within a **LIST** output, then leave this parameter undefined.

Additional Information

The regular expression "^ *l" specifies that a file link is indicated by a line starting with the lowercase "l," preceded possibly by blanks. For example,

```
lrwxr-xr-x   2 ed        usr   2048 Apr 17 17:43 p -> public_html
```

File Link Symbol Regex Expression

Definition

Specifies a regular expression that parses the external file link name in the output of a **LIST** command. Only the link name is required for the file list to be returned.

Required Values

A regular expression. If there is no reliable way of identifying a file link within a **LIST** output, then leave this parameter undefined.

Additional Information

The regular expression "[ ] ->[ ]" defines that a file link symbol is represented by an arrow surrounded by spaces (" -> "). When parsed, only the file name to the right of the symbol will be used. In the following example, only the **public_html** would be used, not the "p" character:

```
lrwxrwxrwx   2 ed        usr  4 Apr 17 17:43 p -> public_html
```

List Line Format

Definition

Specifies whether fields in each line are blank delimited or fixed, that is, whether information always appears at certain columns.

Required Values

Blank Delimited or **Fixed**.

Additional Information

Even though some lines appear to be blank delimited, be wary of certain fields continuing their maximum value when juxtaposed with the next field without any separating blank. In such a case, we recommend you declare the line as “Fixed”. For example,

```
-rw-r--r--    1 ed         usr            110 Apr 15 13:33 aaa
^^^^^^^^^^    ^ ^^         ^^^            ^^^ ^^^ ^^ ^^^^^ ^^^
     1        2  3          4              5   6  7    8    9
```

Valid File Line Minimum Position

Definition

Specifies the minimum number of positions (inclusive) a listing line must have in order to be considered as a possible valid file name line.

Required Values

For a **Fixed** list line format, enter a value equal to the number of columns, counting the first column at the far left as column 1. For a **Blank Delimited** list line format, enter a value equal to the number of fields, counting the first field on the far left as field 1.

For either case, if no minimum can be determined, set this value to zero (0).

Additional Information

For example, in the **Blank Delimited** line below, the minimum number of fields is 9:

```
-rw-r--r--    1 ed         usr            110 Apr 15 13:33 aaa
^^^^^^^^^^    ^ ^^         ^^^            ^^^ ^^^ ^^ ^^^^^ ^^^
     1        2  3          4              5   6  7    8    9
                                                             File Name
```

Note: *The URL FTP Proxy will fail on ascertaining file names that have leading and/or trailing blanks.*

File Name Is Last Entity

Definition

Specifies whether the file name is the last entity on each line. This allows the file name to have imbedded blanks (however, leading or trailing blanks are not supported).

Required Values

Yes or **No**.

File Name Position

Definition

Specifies the starting position (inclusive) of a file name.

Required Values

For **Fixed** list line format, enter the column number, counting the first column on the far left as column 1. For **Blank Delimited** list line format, enter the field number, counting the first field on the extreme left as field 1.

Additional Information

For **Blank Delimited** List Line Format only, if the file name has imbedded blanks, then it can span over several fields.

For example,

```
-rw-r--r--   1 ed         usr           110 Apr 15 13:33 aaa
^^^^^^^^^^   ^ ^^         ^^^           ^^^ ^^^ ^^ ^^^^^ ^^^
    1        2  3          4             5   6  7    8    9
                                                          File Name
```

File Name Length

Definition

Represents the maximum width of a file name; valid only for **Fixed** list line format.

Required Values

- **An Integer:** Positive lengths imply that the file name is right-justified within the maximum field width, and thus leading-blanks are discarded.
- **Negative Lengths:** That is, compared to the absolute length, imply that the file name is left-justified and trailing-blanks are discarded.
- **Zero (0) Value Length:** If the file name is at the end of a file listing line, this value implies that the file name field extends to the end of the line.

*Note: For **Blank Delimited** list line format, this value is usually zero (0). However, if the **File Name Length** parameter is supplied even though a **Blank Delimited** list line format is specified, this implies that if the file name field exceeds the given length, then the rest of the **List Line** data occurs on the following line.*

File Extension Position

Definition

Specifies the left-most position of the file extension for those operating systems that present the file name extension separated from the main file name.

Required Values

For **Fixed** list line format, enter the column number, counting the first column at the extreme left as column 1. For **Blank Delimited** list line format, enter the field number, counting the first field at the far left as field 1. If there is no file extension (as on UNIX systems) set the value to zero (0).

File Extension Length

Definition

Specifies the maximum width of the file extension; valid only for **Fixed** list line format.

Required Values

- **An Integer**
- **Positive Lengths:** Imply that the file extension is right-justified within the maximum field width and therefore leading-blanks are discarded.
- **Negative Lengths:** Imply that the file extension is left-justified and trailing-blanks are discarded (the absolute length is used).
- **Value of Zero (0):** *Always* for the **Blank Delimited** list line format.

File Size Verifiable

Definition

Specifies whether the file size is verifiable, significant, and accurate within a directory listing.

Required Values

Yes or **No**. The **File Size Stability Check** configurable parameter must also be enabled.

Additional Information

Even if the file size field of a listing line is not significant (that is, it is there but only represents an approximate value), the value of this parameter should be **No**, but the file size location should still be declared in the **File Size Position** parameter below to assist determining which line of listing represents a valid file name.

Note: *Use of this parameter does not guarantee that the file is actually stable. We do not recommend that you rely on this functionality for critical data; the feature is intended only for backward compatibility with previous FTP implementations.*

Example

```
    -rw-r--r--   1 ed       usr           110 Apr 15 13:33 aaa
                                          ^^^
                                          File Size
```

File Size Position

Definition

Specifies the left-most position in the listing line that represents the size of the file. Even though for some operating systems the value shown might not truly reflect the file size, this position is still important in ascertaining that the line contains a valid file name.

Required Values

A non-negative integer. For **Fixed** list line format, the position value is the column number (starting with one (1) on the far left). For **Blank Delimited**, this value represents the field number (starting with one (1) on the far left). If the **LIST** line does not have a size field, set this parameter to zero (0).

Additional Information

Example

```
-rw-r--r--   1 ed       usr           110 Apr 15 13:33 aaa
^^^^^^^^^^   ^ ^^       ^^^           ^^^ ^^^ ^^ ^^^^^ ^^^
     1        2  3        4             5   6  7    8    9
                                    File
                                       Size
```

The following represent valid number representations of file sizes:

```
1234 or 1,234,567 or -12345 or +12345 or '  1234  ' or 12/34 or
1,234/56
```

The following represent invalid number representations of file sizes (the ^ indicates where the error occurs):

```
'12 34' or 123,45,678 or 123-456-789 or --123 or 123-
   ^          ^                 ^        ^          ^
or 12345678901 or any number > 4294967295 or < -2147483647
   ^ (too large)
or 123.45 or 12AB34 or 0x45 or ,123,456 or 12//34
      ^        ^        ^      ^              ^
or /123 or 123/ or 12,3/45
   ^          ^        ^
```

File Size Length

Definition

Specifies the maximum width (number of columns) of the file size field, only valid for **Fixed** List Line Format.

Required Values

A non-negative integer. For **Blank Delimited** list line format, set this value to zero (0).

Special Envelope For Absolute Path Name

Definition

Specifies special enveloping characters required to surround an absolute path name (for example, single quotes are used in MVS). Only use a single quote at the start of the directory name.

Required Values

A pair of enveloping characters. Even if the leading and trailing character is identical, enter it twice.

If no enveloping characters are required for an operating system, leave this parameter undefined.

Note: *On UNIX, this parameter is always undefined.*

Listing Directory Yields Absolute Path Names

Definition

Specifies whether, when the **DIR** command is used on a directory name, the resulting file names are absolute.

Required Values

Yes or **No**.

Note: *On UNIX, this character is always set to* ***No****.*

Absolute Path Name Delimiter Set

Definition

Specifies any absolute path requiring certain delimiters to separate directory names (or their equivalent) from each other and from the file name.

Required Values

Enter the delimiters for the absolute path, starting from the left, for:

- Initial (left-most) directory delimiter
- Intermediate directory delimiters
- Initial (left-most) file name delimiter
- Optionally, the ending (right-most) file name delimiter

Wherever there is no specific delimiter, use "\0" (backslash zero) to act as a placeholder. Delimiters that are backslashes need to be escaped with another backslash.

Additional Information

OS	**Path Name Format**	**Delimiter Set**				
		1	**2**	**3**	**4**	**Enter**
UNIX	/dir1/dir2/file.ext	/	/	/		///
Windows	C:\dir1\dir2\file.ext	\\	\\	\\		\\\\\\
VMS	disk1:[dir1.dir2]file.ext;1	[	.	]	;	[.];
MVS	dir1.dir2(file)	\0	.	(	)	\0.()
MVS PS	dir1.dir2.filename	\0	.	.		\0..
MVS GDG	dir1.dir2.file(version)	\0	.	.		\0..
AS400	dir1/file.ext	\0	/	.		\0/.

Change Directory Before Listing

Definition

Determines whether a change directory (**cd**) needs to be done before issuing the **DIR** command to get a listing of files under the desired directory.

Required Values

Yes or **No**.

Directory Name Requires Terminator

Definition

Determines whether a directory name that is not followed immediately by a file name requires the ending directory delimiter as a terminator (for example, as on VMS).

Required Values

Yes or **No**.

3.3 Environment Configuration

To support the operation of this e*Way, no changes are necessary, either in the Participating Host's operating environment or in the e*Gate system.

Note: *Changes to Monk files can be made using the Collaboration Rules Editor (available from within the e*Gate Enterprise Manager) or with a text editor. However, if you use a text editor to edit Monk files directly, you **must** commit these changed files to the e*Gate Registry or your changes will not be implemented.*

*For more information about committing files to the e*Gate Registry, see the Enterprise Manager's online Help system, or the **stcregutil** command-line utility section in the **e*Gate Integrator System Administration and Operations Guide**.*

3.4 External Configuration Requirements

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

Chapter 4

Message-based Configuration

This chapter explains the message-based operation of the Batch e*Way and explains how to use the Dynamic Configuration parameters for this e*Way.

4.1 General Operation

There are the following cases for the ordering transmission:

- Order e*Way to send one time (possibly to multiple destinations)
- Order e*Way to receive one time (possibly from multiple sources)

In either of these cases, the "order" Extensible Markup Language (XML) message has the following basic format:

```
<batch_e*Way_order>
    <command>            (command)    </command>
    <order_record>
        <error_record>
        </error_record>
    </order_record>
    <order_record>
        <error_record>
        </error_record>
    </order_record>
    <payload>            (DATA)       </payload>
</Batch_e*Way_Order>
```

The main record has the following subrecords:

- **Command** can be "send" or "receive."
- **Order** contains the details for sending or retrieving to/from a single source/destination.
- **Error** contains error information published by the e*Way after attempting to execute the order. This subrecord is only sent if **Publish Status Record on Error** (see **"Publish Status Record on Error" on page 74**) is set to **Yes**. See also **"Publish Status Record on Success" on page 74**.
- **Payload** (send only) specifies the data to be sent.

The data can come in the following forms:

- In the first case, the payload node can contain base64 data, in which case it has a **payload** attribute set to **base64Insitu**.
- In the second case, the payload node represents the directory for the payload, in which case it has a **payload** attribute equal to **localDir**.

Dynamic Configuration

Use the following Document Type Definition (DTD) and e*Gate Event Type Definition (ETD) files with dynamic configuration:

Table 4 Dynamic Configuration Files

DTD File	Corresponding ETD File
batch_eway_data.dtd	batch_eway_data.xsc
batch_eway_error.dtd	batch_eway_error.xsc
batch_eway_order.dtd	batch_eway_order.xsc

4.1.1 Sending Data with a “Send” Order

The following example shows an XML message:

```
<batch_e*way_order>
    <command>            send             </command>
    <order_record>
    <external_host_setup>
        <host_type>      Unix             </host_type>
        <user_name>      Alincoln         </user_name>
        <encrypted_password> liasdfLIJB   </encrypted_password>
        <file_transfer_method> ftp        </file_transfer_method>
    </external_host_setup>
    <communication_setup>
        <down_timeout>   10               </down_timeout>
        <up_timeout>     20               </up_timeout>
        <resend_timeout> 20               </resend_timeout>
    </communication_setup>
    <publish_to_external>
        <remote_directory_name>/usr/home/honest_abe/to
            </remote_directory_name>
        <remote_file_name> X1.tmp         </remote_file_name>
        <append_or_overwrite>overwrite    </append_or_overwrite>
        <remote_rename_or_archive name>X1.dat
            </remote_rename_or_archive_name>
    </publish_to_external>
    </order_record>
    <payload>            (DATA)           </payload>
</batch_e*way_order>
```

The previous example shows the delivery of a file to an external system. It is one XML message, **batch_e*way_order**, that contains a command record, one or more order records, and finally a single payload. The order record represents one destination for this payload. If any of the individual orders fails, then the e*Way publishes an error record.

Please see **"Send or Receive XML Messages" on page 263** for the corresponding DTD file.

4.1.2 Receiving Data with a "Receive" Order

Receiving from a file is similar to sending, as shown in this example.

```
<batch_e*way_order>
    <command>             receive          </command>
    <external_host_setup>
        <host_type>       Unix             </host_type>
        <user_name>       Alincoln         </user_name>
        <encrypted_password>liasdfLIJB     </encrypted_password>
        <file_transfer_method> ftp         </file_transfer_method>
        <return_tag>      Factor order     </return_tag>
    </external_host_setup>
    <communication_setup>
        <down_timeout>    10               </down_timeout>
        <up_timeout>      20               </up_timeout>
        <resend_timeout>  20               </resend_timeout>
    </communication_setup>
    <subscribe_to_external>
        <remote_directory_name> /usr/home/honest_abe/from
            </remote_directory_name>
        <remote_file_regexp> Y*.dat    </remote_file_regexp>
    </subscribe_to_external>
</batch_e*way_order>
```

In this case, the e*Way retrieves all of the files in the designated directory that match the given regular expression, and stores them in a temporary directory. It then reads the entire contents of each file and sends it to e*Gate as a publication (using the **event-send-to-egate** function).

The message sent is similar to the XML message that initiated the transfer, except for the following characteristics:

- There is one return message per order in the command, instead of one return per command. Thus, if a command is received with orders for three transfers, the e*Way attempts three transfers and returns the three files so retrieved as three "receive" responses.
- The message contains a "payload" field with the data received. See the following example:

```
<batch_e*way_order>
    <command>             receive          </command>
    <external_host_setup>
        <host_type>       Unix             </host_type>
        <user_name>       Alincoln         </user_name>
        <encrypted_password> liasdfLIJB    </encrypted_password>
        <file_transfer_method> ftp         </file_transfer_method>
        <return_tag>      Factor order     </return_tag>
    </external_host_setup>
    <communication_setup>
        <down_timeout>    10           </down_timeout>
        <up_timeout>      20           </up_timeout>
        <resend_timeout>  20           </resend_timeout>
    </communication_setup>
    <subscribe_to_external>
        <remote_directory_name> /usr/home/honest_abe/from
```

```
            </remote_directory_name>
        <remote_file_regexp> Y*.dat       </remote_file_regexp>
    </subscribe_to_external>
    <payload>              (DATA)          </payload>
</batch_e*way_order>
```

The e*Way only acknowledges ("ACK") the order command message after all records have been sent. The **<return_tag>** field of the XML message is used to store a unique tag generated by the originator of the command. This tag allows the e*Gate system administrator to determine, as each response comes back, which system gave that response.

As a final example of the receive command, consider this example of a command to go to three different systems for three different kinds of data, Factory Orders, Builds of Materials, and Engineering Updates.

First, note the following command record (transfer details omitted for brevity):

```
<batch_e*way_order>
    <command>              receive             </command>
        <return_tag>       Factory order       </return_tag>
        <return_tag>       Build of Materials  </return_tag>
        <return_tag>       Engineering Updates </return_tag>
</batch_e*way_order>
```

In this example, the Batch e*Way tries each receive transfer and follows its normal procedures for retrying and raising exceptions, if there are problems. As each transfer succeeds, it returns an XML message with the payload and the corresponding return tag. If it fails, it returns an XML message with the error record.

The e*Way begins with the Factory Order as follows:

```
<batch_e*way_order>
    <command>              receive         </command>
        <return_tag>       Factory order   </return_tag>
    <payload>              (DATA)          </payload>
</batch_e*way_order>
```

The e*Way then continues with each of the other two (Build of Materials and Engineering Updates) as follows:

```
<batch_e*way_order>
    <command>        receive             </command>
        <return_tag> Build of Materials  </return_tag>
    <payload>        (DATA)              </payload>
</batch_e*way_order>


<batch_e*way_order>
    <command>            receive             </command>
        <return_tag>     Engineering Updates </return_tag>
    <payload>            (DATA)              </payload>
</batch_e*way_order>
```

See **"Send or Receive XML Messages" on page 263** for the corresponding DTD file.

4.1.3 Error Reporting

If the parameter **Publish Status Record on Error** (see **“Publish Status Record on Error” on page 74**) is set to **Yes**, and the e*Way has problems with one order, it publishes the command message with all orders stripped out, except those that failed, as well as the population of the corresponding error records.

See the following template:

```
<batch_e*way_order>
    <command>       (command)       </command>
    <order_record>
    <error_record>
    <error_code>                    </error_code>
    <error_text>                    </error_text>
    <last_action>                   </last_action>
    </error_record>
    </order_record>
    <payload>       (DATA)          </payload>
</batch_e*way_order>
```

The “last action” record contains whatever command the e*Way can indicate. Thus, if there is a failure on renaming a file after the transfer, the e*Way populates this field with the rename command it is trying to carry out.

Please see **“Error Messages” on page 264**, for the corresponding DTD file.

4.2 Configuration

The Batch e*Way consists of several sections that contain parameters for configuring the Batch e*Way, one of which is **Dynamic Configuration**. For details about the other configuration sections for the Batch e*Way, see **“e*Way Configuration Parameters” on page 21**.

The configuration parameters are set by using the e*Way Editor. To change e*Way configuration parameters, do the following operations:

1. In the Enterprise Manager’s Component editor, select the e*Way you want to configure and display its properties.
2. Under **Configuration File**, click **New** to create a new file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file.
3. In the **Additional Command Line Arguments** box, type any additional command line arguments that the e*Way may require, taking care to insert them *at the end* of the existing command-line string. Be careful not to change any of the default arguments unless you have a specific need to do so.

For more information about how to use the e*Way Editor, see the e*Way Editor’s online Help or the *e*Gate Integrator User’s Guide*.

4.2.1 Dynamic Configuration

This section explains the following parameters for a dynamic Batch e*Way:

- **"Enable Message Configuration" on page 71**
- **"Publish Status Record on Success" on page 74**
- **"Publish Status Record on Error" on page 74**
- **"Include Order Record in Error Record" on page 74**
- **"Include Payload in Error Record" on page 75**
- **"Action on Mal-formed Command" on page 75**

Enable Message Configuration

Description

Use this parameter to indicate that the e*Way contains an XML message which determines its activities. The XML message should contain all relevant parameters that govern the transfer, including the data to be sent (if it is an outbound transfer). See **Appendix A**, **"Document Type Definitions" on page 263** for details about the DTD.

Note: *When the XML message sets the e*Way to receive, Batch retrieves the external file and wraps it into XML payload (see* **Data Message** *on page 265), and transforms the data into Base64 format. To send the data back in its original format, use the* ***Base64-to-Raw*** *Monk function. Details on how to use this function are explained in the* ***Monk Developer's Reference Guide****.*

Required Values

Yes or **No**. (**No** is the default).

When this parameter is set to **Yes**, the Batch e*Way becomes Event-driven, so it does NOT exchange data based on scheduling, and the record type is always a single record.

Note: *If the fields marked as "Overridden by message" are set by the XML message, then the table below holds true. However, if the fields are NOT set by the XML message, then those fields marked as "Overridden by message" MUST be specified in the* ***.cfg*** *file. Only publication OR subscriptions fields must be set, unless this e*Way is a publisher AND a subscriber.*

Furthermore, when the Message Configuration is enabled, certain Configuration Sections (see page **"e*Way Configuration Parameters" on page 21**) and parameters are affected, as shown in the **Table 5 on page 72**.

Table 5 Effect of Message Configuration Enabled

Section	Parameter	Effect
Communication Setup	Start Exchange Data Schedule	Ignored.
	Stop Exchange Data Schedule	Ignored.
	Exchange Data Interval	Ignored.
	Zero Wait Between Successful Exchanges	Ignored.
	Down Timeout	Ignored.
	Up Timeout	Ignored.
External Host Setup	External Host Name	Overridden by message.
	Host Type	Overridden by message.
	User Name	Overridden by message.
	Encrypted Password	Overridden by message.
	File Transfer Method	Overridden by message.
Monk Configuration	Process Outgoing Message Function	The XML event is parsed and processed.
	Exchange Data With External Function	Ignored.
	Positive Acknowledgment Function	Ignored.
	Negative Acknowledgment Function	Ignored.
	Startup Function	Normal behavior, but the value assigned to transfer method is ignored.
Publish To External	Remote Directory Name	Overridden by message.

Table 5 Effect of Message Configuration Enabled (Continued)

Section	Parameter	Effect
Publish To External	Remote File Name	Overridden by message.
	Append or Overwrite when Transferring Files	Overridden by message.
	Record Type	Automatically set to **Single Record**. Any other value is ignored.
	Record Delimiter	Ignored.
	Delimiter on Last Record	Ignored.
	Record Size	Ignored.
	Remote Command After Transfer	Overridden by message.
	Remote Rename or Archive Name	Overridden by message.
	Local Command After Transfer	Overridden by message.
	Local Archive Directory	Overridden by message.
Recourse Action	Action on Fetch Failure	Normal behavior, but an additional option is needed to publish the Event that contains the configuration message.
	Action on Send Failure	Normal behavior, but an additional option is needed to publish the Event that contains the configuration message.
Sequence Numbering	Starting Sequence Number	Ignored.
	Max Sequence Number	Ignored.
SOCKS	Server Host Name	Overridden by message.
	Server Port	Overridden by message.
Subscribe To External	Remote Directory Name	Overridden by message.
	Remote File Regexp	Overridden by message.
	Record Type	Ignored.
	Record Delimiter	Ignored.
	Delimiter on Last Record	Ignored.
	Record Size	Ignored.
	Remote Command After Transfer	Overridden by message.
	Remote Rename or Archive Name	Overridden by message.
	Local Command After Transfer	Overridden by message.
	Local Archive Directory	Overridden by message.

Table 5 Effect of Message Configuration Enabled (Continued)

Section	Parameter	Effect
FTP	server_port	Overridden by message.
	mode	Overridden by message.
	Pretransfer_Commands	Overridden by message.
	Posttransfer_Commands	Overridden by message.

Publish Status Record on Success

Description

When this parameter is set to **Yes**, the Batch e*Way will publish a "good error" record to e*Gate, with the same format that is specified in **batch_eway_error.dtd**. (See **“Error Messages” on page 264**.) The “good error” record is published only when the payload has been successfully sent to the remote host.

Note: *The user must configure an inbound topic and process this event.*

The **<error_code>** element of the XML message will be zero (0) to indicate that there are no errors, and the **<error_text>** will represent the time the file was successfully transferred.

An example follows:

```
Successfully sent on: Fri, 29 Jun 2001 at 14:02:30 PDT
```

See also **“Enable Message Configuration” on page 71** and **“Publish Status Record on Error” on page 74**.

Required Values

Yes or **No**. **No** is the default.

Publish Status Record on Error

Description

This parameter determines whether or not the Batch e*Way publishes an error record to e*Gate. The error record is in the format of **batch_eway_error.dtd** (See **“Error Messages” on page 264**). However, you are required to configure an inbound topic to process this Event.

Required Values

Yes or **No**. **No** is the default.

Include Order Record in Error Record

Description

If this parameter is set to **Yes**, the Batch e*Way includes an Order Record as part of an error record when **Publish Status Record on Error** is enabled.

Required Values

Yes or **No**. **No** is the default.

Include Payload in Error Record

Description

If this parameter is set to **Yes**, the Batch e*Way includes the Payload as part of an Error Record when the **Order Record Command** is **Send**.

Required Values

Yes or **No**. **No** is the default.

Action on Mal-formed Command

Description

If **Enable Message Configuration** is set to **Yes**, the Batch e*Way requires a specific XML message structure. This parameter specifies the action that the Batch e*Way takes when the Outgoing Event doesn't match the XML message structure the e*Way requires.

Required Values

One of the following values:

- Exit
- Ignore
- Raise Alert
- Publish Error Record

Exit is the default.

Chapter 5

Implementation

This chapter explains how the Batch e*Way is implemented and provides sample configurations.

5.1 Implementation Notes

In implementing the Batch e*Way, you need to know the following important functions:

- How the e*Way uses temporary files
- Record type configuration

Note: *The Batch e*Way is unable to detect that a file is being transmitted and only checks for a file entry, without trying to determine whether the file is still actively being transmitted.*

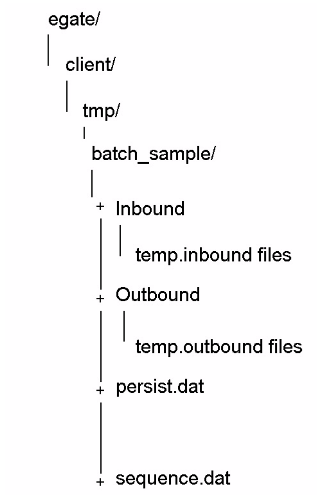
5.1.1 How the e*Way Uses Temporary Files

For each e*Way, a directory tree is created under the e*Way's name in the following directory:

egate/client/tmp

For example, the directory structure shown in Figure 8 below would be created for the e*Way called **batch_sample**.

Figure 8 Batch e*Way Directory Tree



The file **sequence.dat** is used for holding the current sequence number, if sequence numbering is being used.

The file **persist.dat** is used only for inbound e*Ways. It holds information about the current state of processing of temporary files.

Table 6 below shows the file structure of **persist.dat.**

Table 6 File Structure of persist.dat

Bytes	Description
0-19	file offset
20-29	list index
30+	file name list
	"-*- End of Files -*-" (terminator)

The Batch e*Way fetches one or more files needed from the external system. Records are read from these files one at a time. The **persist.dat** file stores the following information:

- The list of files received
- The file being worked on (the index)
- The position within that file (the offset) is stored after every read from a file

Note: *If the e*Way is shut down, it continues at the point where it left off.*

Files retrieved from the external system are stored in the inbound directory, and outbound files (waiting to be sent to the external system) are stored in the outbound directory.

Note: *It is recommended that you do not edit* ***persist.dat*** *or move files around in the inbound or outbound directories.*

5.1.2 Record Type Configuration

The Batch e*Way works with delimited files, fixed length record files and with single-record files, as set by the **Record Type** parameters in the **Subscribe to External Settings** (see **"Subscribe to External" on page 42**) and **Publish to External Settings** (see **"Publish to External" on page 45**) configuration parameter sections.

The behavior of each **Record Type** differs with the direction of transfer.

Delimited Record

Subscribing to the External System

When subscribing to the external system, the Batch e*Way expects each record in an inbound file to be separated with a delimiter defined by the **Record Delimiter** parameter. The delimiter can be a multi-character text string, and it can contain special characters (see **"Using Special Characters" on page 53**). These characters are expanded once only, at initialization.

Records with the given delimiter are read from local temporary files, one at a time. If the **Delimiter on Last Record** parameter is set to **Yes**, then a final record, which does not have a terminating delimiter is ignored.

Publishing to the External System

When publishing to the external system, the e*Way creates a single local file containing records for transmission to the external system. The file is sent to the external system at the next scheduled time for data transfer.

All received records are appended to the local file separated by a single or multi-character string as defined by the **Record Delimiter** parameter. The delimiter may include special characters (see **"Using Special Characters" on page 53**). These characters are expanded once only, at initialization.

If the **Delimiter on Last Record** parameter is set to **Yes**, then the delimiter character(s) are added to the final record.

Fixed-length Record

Subscribing to the External System

Records of the size given in the parameter **Subscribe to External: Record Size** are read from the local temporary files, and passed back through the e*Way one at a time.

Publishing to the External System

Only records of the size given in the parameter **Publish to External: Record Size** are accepted and stored in the temporary file for later transmission. Records with a different length are rejected as data errors.

The file is sent to the external system at the next scheduled time for data transfer.

Single Record

Subscribing to the External System

Each local temporary file is treated as if it contains a single record. The file is read in its entirety and passed through the e*Way.

Publishing to the External System

This setting means that only one record is written to the temporary file. Under normal circumstances, this means that only one file will be created, containing a single record. However, multiple files may be created if the parameter **Publish To External File Name**:

- Contains a sequence number.
- Is the name of a Monk function (beginning with **monk-**). It is assumed that this routine will return a different file name each time, and multiple files will be created.

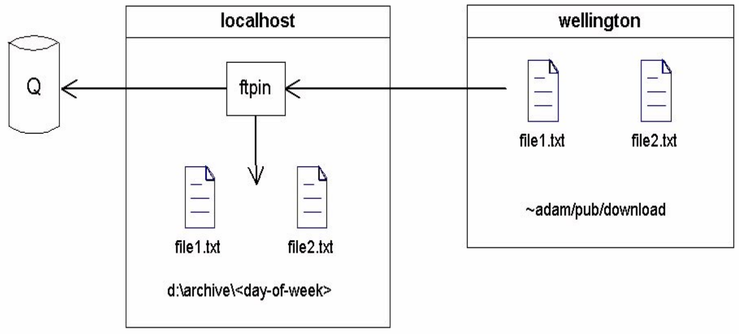
5.2 Sample Configurations

5.2.1 Subscribing to an External System

In this example, the Batch e*Way fetches two files from the remote UNIX machine **wellington** every 24 hours, using the FTP protocol. These files are stored in the home directory of user **adam**, under the subdirectory **pub/download.**

Figure 9 below shows a diagram of this setup.

Figure 9 Subscribe-to-external-system Setup



This setup has the following additional characteristics:

- The names of the two files are **file1.txt** and **file2.txt**. No other files are required.
- The two files contain multiple records delimited by a new line (**\n**) character.
- After retrieving the files from the remote system, the Batch e*Way deletes the remote copy.
- The last seven day's worth of files on the local system are kept.

The **Table 7 on page 80** lists the most critical parameters and the settings required to achieve the setup described previously.

Table 7 Parameters for the Input Example

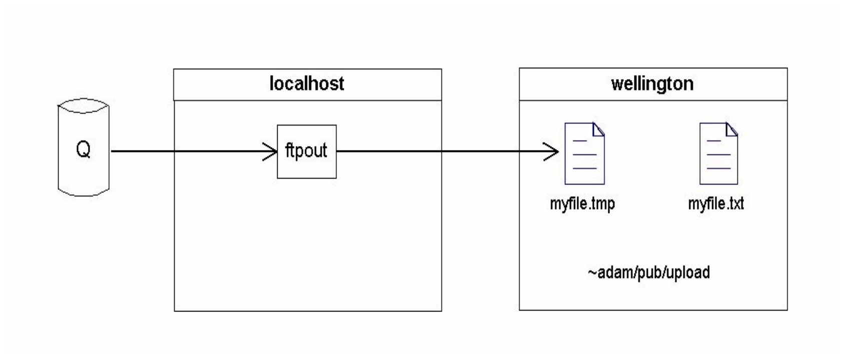
Section	Parameter	Value
Communication Setup	Start Exchange Data Schedule	Repeatedly, every 24 hours
External Host Setup	Host Type	UNIX
	External Host Name	wellington
	User Name	adam
	Encrypted Password	********
	File Transfer Method	FTP
Subscribe To External	Remote Directory Name	pub/download
	Remote File Regexp	**^file[12].txt$**
	Record Type	Delimited
	Record Delimiter	\n
	Delimiter on Last Record	Yes
	Remote Command After Transfer	delete
	Local Command After Transfer	archive
	Local Rename or Archive Name	**d:\\archive/%a**

5.2.2 Publishing to an External System

In this example, the Batch e*Way sends a file containing new line (**\n**) delimited messages to the remote UNIX machine **wellington,** using the FTP protocol. The file is created in the subdirectory **pub/upload**, under the user **adam**.

Figure 10 shows a diagram of this setup.

Figure 10 Publish-to-external-system Setup



This file is sent once every hour under the name **myfile.tmp**, and will be renamed after it arrives to **myfile.txt**. This technique can be used if there is a process on the remote machine watching for a file to be created, but we want to make sure that it does not see the file until it is there in its entirety.

A copy of the file on the local system is not required and is deleted.

The Table 8 lists the most critical parameters and the settings required to achieve the setup described previously.

Table 8 Parameters for the Output Example (Sheet 1 of 2)

Section	Parameter	Value
Communication Setup	Start Exchange Data Schedule	Repeatedly, every 1 hour
External Host Setup	Host Type	UNIX
	External Host Name	wellington
	User Name	adam
	Encrypted Password	********
	File Transfer Method	FTP

Table 8 Parameters for the Output Example (Sheet 2 of 2)

Section	Parameter	Value
Publish To External	Remote Directory Name	pub/upload
	Remote File Name	**myfile.tmp**
	Append or Overwrite when Transferring Files	Overwrite
	Record Type	Delimited
	Record Delimiter	\n
	Delimiter on Last Record	Yes
	Remote Command After Transfer	rename
	Local Command After Transfer	delete
	Remote Rename or Archive Name	**myfile.txt**

Note: *For AIX system, you must use the* ***ebcdic->ascii*** *Monk function to convert any EBCDIC data before transporting it to an ASCII-based system. See the* ***Monk Developer's Reference Guide*** *for more details about this function.*

Chapter 6

Batch e*Way Functions

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

Basic Functions on page 83

Core Functions on page 91

Connection and File Functions on page 102

File Name Expansion Functions on page 131

Post-transfer Routines on page 143

File Copy Transfer Functions on page 146

FTP Transfer Functions on page 160

Advanced FTP Functions on page 172

File System Functions on page 217

Note: *For AIX and system, you must use the* ***ebcdic->ascii*** *Monk function to convert any EBCDIC data before transporting it to an ASCII-based system. See the Monk Developer's Reference Guide for more details about this functio*n.

When the Batch e*Way is executing a send command, and a Monk exception goes undetected while a message is being sent, the command is aborted, the Batch e*Way will shut down, and the connection to the FTP server will be lost. This behavior is by design to prevent the loss of a message.

6.1 Basic Functions

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

The basic functions are

event-send-to-egate on page 85

get-logical-name on page 86

send-external-down on page 87

event-send-to-egate

Syntax

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

Description

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

Parameters

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

Return Values

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

Throws

None.

Additional Information

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

get-logical-name

Syntax

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

Description

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

Parameters

None.

Return Values

String

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

Throws

None.

send-external-down

Syntax

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

Description

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

Parameters

None.

Return Values

None.

Throws

None.

send-external-up

Syntax

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

Description

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

Parameters

None.

Return Values

None.

Throws

None.

shutdown-request

Syntax

```
(shutdown-request)
```

Description

shutdown-request completes the e*Gate shutdown procedure that was initiated by the Control Broker but was interrupted by returning a non-null value within the **Shutdown Command Notification Function** (see **"Shutdown Command Notification Function" on page 40**). 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 open a one-time window for the exchange of data with the external system (see **"Exchange Data with External Function" on page 36**). This function operates with the **Exchange Data Interval** parameter (see **"Exchange Data Interval" on page 25**), starting the exchange of data, according to this parameter, until you close the window using the stop-schedule function (see **stop-schedule** on page 91).

The **start-schedule** function does not affect any defined schedules. See also **"Start Exchange Data Schedule" on page 23**.

Note: *Use this function only when the **Start Exchange Data Schedule** and **Stop Exchange Data Schedule** parameters are in operation. Otherwise, data exchange is already occurring on a continuous basis, and no window needs to be opened.*

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 (see **"Exchange Data with External Function" on page 36**). Execution will be stopped when the e*Way concludes any open transaction and does not halt the e*Way process itself.

This function does not affect any defined schedules. See also **"Stop Exchange Data Schedule" on page 24**.

Parameters

None.

Return Values

None.

Throws

None.

6.2 Core Functions

The functions in this category are those called by e*Way configuration parameters (see **"Monk Configuration" on page 26**).

The core functions are

batch-ack on page 92

batch-exchange-data on page 93

batch-ext-connect on page 94

batch-ext-shutdown on page 95

batch-ext-verify on page 96

batch-init on page 97

batch-nak on page 98

batch-proc-out on page 99

batch-regular-proc-out on page 100

batch-shutdown-notify on page 101

batch-startup on page 102

batch-ack

Syntax

```
(batch-ack command)
```

Description

batch-ack is called automatically when the e*Way successfully processes and queues Events from the external system.

Parameters

Name	Type	Description
command	string	Any non-null string

Return Values

String
Returns “FAILURE” on all errors; otherwise, returns a null string.

Throws

None.

Location

batch-ack.monk

Additional Information

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 a Collaboration for further processing. If the Event’s processing is completed successfully, the e*Way executes the **Positive Acknowledgment** function (otherwise, the e*Way executes the **Negative Acknowledgment** function).

This function can return an Event to be queued, but the e*Way will not **ACK/NAK** the external system.

The e*Way will exit if it fails its attempt to invoke this function or this function returns a “FAILURE” string.

batch-exchange-data

Syntax

```
(batch-exchange-data)
```

Description

batch-exchange-data initiates an exchange of Events with an external system. This function can exchange either inbound or outbound Events.

Parameters

None.

Return Values

String

Returns a null string if the function processed an *outbound* Event successfully; otherwise, returns a string to be packaged as an *inbound* Event.

Throws

None.

Location

batch-exchange-data.monk

batch-ext-connect

Syntax

```
(batch-ext-connect)
```

Description

batch-ext-connect establishes (or re-establishes) a connection to the external system.

Parameters

None.

Return Values

String
Returns "UP" if the connection was made successfully; otherwise, returns "DOWN."

Throws

except-method

Location

batch-ext-connect.monk

batch-ext-shutdown

Syntax

```
(batch-ext-shutdown command)
```

Description

batch-ext-shutdown shuts down the connection between the external system and the e*Way.

Parameters

Name	Type	Description
command	string	Any non-null string

Return Values

String
Returns a null string.

Throws

except-method

Location

batch-ext-shutdown.monk

batch-ext-verify

Syntax

```
(batch-ext-verify)
```

Description

batch-ext-verify confirms that the external system is operating and available.

Parameters

None.

Return Values

String
Returns “UP” if the connection was verified successfully; otherwise, returns “DOWN.”

Throws

except-method

Location

batch-ext-verify.monk

batch-init

Syntax

```
(batch-init)
```

Description

batch-init defines a number of variables upon which other e*Way functions will rely, defines exceptions, and loads the library file **stc_monkfilesys.dll**.

Parameters

None.

Return Values

String
Returns “FAILURE” on all errors; otherwise, returns a null string.

Throws

except-method

Location

batch-init.monk

batch-nak

Syntax

```
(batch-nak command)
```

Description

batch-ack is called automatically when the e*Way fails to process and queue Events from the external system.

Parameters

Name	Type	Description
command	string	Any non-null string

Return Values

String
Returns “FAILURE” on all errors; otherwise, returns a null string.

Throws

None.

Location

batch-nak.monk

Additional Information

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 a Collaboration for further processing. If the Event’s processing is completed unsuccessfully, the e*Way executes the **Negative Acknowledgment** function; otherwise, the e*Way executes the **Positive Acknowledgment** function.

This function can return an Event to be queued, but the e*Way does not return a positive or negative acknowledgement to the external system.

The e*Way exits if it fails its attempt to invoke this function or this function returns a “FAILURE” string.

batch-proc-out

Syntax

```
(batch-proc-out Event)
```

Description

batch-proc-out sends the outbound Event from the e*Way to the external system.

Parameters

Name	Type	Description
Event	string	The Event to be sent

Return Values

String

Returns one of the following strings:

- Null
- RESEND
- CONNERR
- DATAERR

See **Figure 7 on page 33** for an explanation of the effect of each of these return values.

Throws

None.

Location

batch-proc-out.monk

batch-regular-proc-out

Syntax

```
(batch-regular-proc-out Event)
```

Description

batch-regular-proc-out sends the outbound Event from the e*Way to the external system.

Parameters

Name	Type	Description
Event	string	The Event to be sent

Return Values

String

Returns one of the following strings:

- Null
- RESEND
- CONNERR
- DATAERR

See **Figure 7 on page 33** for an explanation of the effect of each of these return values.

Throws

None.

Location

batch-regular-proc-out.monk

batch-shutdown-notify

Syntax

```
(batch-shutdown-notify command)
```

Description

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

Parameters

Name	Type	Description
command	string	Any non-null string

Return Values

String
Returns a null string.

Throws

None.

Location

batch-shutdown-notify.monk

batch-startup

Syntax

```
(batch-startup)
```

Description

batch-startup launches a Monk function that start the e*Way. The function invoked depends on whether the e*Way uses FTP or file transfer via **copy** (selected by a configuration parameter; see **"File Transfer Method" on page 41**).

Parameters

None.

Return Values

String
Returns "FAILURE" on all errors; otherwise, returns a null string.

Throws

except-method

Location

batch-startup.monk

6.3 Connection and File Functions

These functions initiate the connections to the external system and transfer files between the e*Way and the external system. 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 connection and file functions are:

batch-fetch-files-from-remote on page 104

batch-fetch-named-files on page 105

batch-send-path-file on page 106

batch-validate-params on page 108

batch-write-file on page 109

disconnect-from-remote on page 110

fetch-files-from-remote on page 111

fetch-named-files on page 112

get-next-record on page 113

list-files-on-remote on page 115

open-next-working-file on page 116

batch-fetch-files-from-remote

Syntax

```
(batch-fetch-files-from-remote transferMethod ftpHandle ftpMode
                               remoteDirectory remoteFileRegexp
                               postTransferCommand
                               remoteRenameArchiveName)
```

Description

batch-fetch-files-from-remote attempts to fetch all files from the external system specified by an XML message.

Parameters

Name	Type	Description
transferMethod	s5tring	Identifies whether transfer method is FTP.
ftpHandle	string	The FTP handle.
ftpMode	string	The FTP mode.
remoteDirectory	string	The path name at the remote location from which the files are to be fetched.
remoteFileRegexp	string	A regular expression that describes files to be retrieved. (See **"Using Special Characters" on page 53**.)
postTransferCommand	string	The command that the e*Way will execute after a successful file transfer (for example, Delete, Rename, or Archive).
remoteRenameArchiveName	string	Depending on the value of postTransferCommand, the parameter specifies either the name to which the external file will be renamed (for Rename), or the directory in which to archive the external file (for Archive).

Return Values

List
Returns the list of files fetched.

Throws

except-abort

Location

batch-fetch-files-from-remote.monk

batch-fetch-named-files

Syntax

```
(batch-fetch-named-files transferMethod ftpHandle ftpMode
                         postTransferCommand
                         remoteRenameArchiveName file-list)
```

Description

batch-fetch-named-files attempts to fetch a list of files from the external system.

Parameters

Name	Type	Description
transferMethod	string	Identifies whether transfer method is FTP.
ftpHandle	string	The FTP handle.
ftpMode	string	The FTP mode.
postTransferCommand	string	The command that the e*Way will execute after a successful file transfer (for example, Delete, Rename, or Archive).
remoteRenameArchiveName	string	Depending on the value of postTransferCommand, the parameter specifies either the name to which the external file will be renamed (for Rename), or the directory in which to archive the external file (for Archive).
file-list	List	A list of files to be transferred from the external system.

Return Values

List
Returns a list of files successfully fetched.

Throws

except-method, **except-abort**

Location

batch-fetch-named-files.monk

batch-send-path-file

Syntax

```
(batch-send-path-file transferMethod ftpHandle ftpMode
                         appendOverwrite localFilename
                         remoteDirectory remoteFilename
                         rmtpostTransferCommand
                         remoteRenameArchiveName
                         localPostTransferCommand
                         localArchiveDirectory)
```

Description

batch-send-path-file attempts to send files to an external system.

Parameters

Name	Type	Description
transferMethod	string	Identifies whether transfer method is FTP.
ftpHandle	string	The FTP handle.
ftpMode	string	The FTP mode.
appendOverwrite	string	Specifies whether to append the records in the file being transferred to the existing file on the external system, or to overwrite the existing file on the external system with the file being transferred.
localFilename	string	The name of the file being sent to the external system.
remoteDirectory	string	The path name at the remote location to which the file is to be sent.
remoteFilename	string	The name of the file on the external system that is being overwritten or appended.
rmtPostTransferCommand	string	The command that the e*Way will execute on the remote system after a successful file transfer (for example, Delete, Rename, or Archive).
RemoteRenameArchiveName	string	Depending on the value of rmtPostTransferCommand, the parameter specifies either the name to which the external file will be renamed (for Rename), or the external system directory in which to archive the file (for Archive).

Name	Type	Description
localPostTransferCommand	string	The command that the e*Way will execute on the local after a successful file transfer (for example, Delete, Rename, or Archive).
LocalArchiveDirectory	string	Specifies the local directory in which to archive the file.

Return Values

Undefined.

Throws

except-abort

Location

batch-send-path-file.monk

batch-validate-params

Syntax

```
(batch-validate-params)
```

Description

batch-validate-params validates a number of parameters used by other functions. It provides a double-check that any modifications made to selected crucial Monk functions have not altered the validated parameters.

Parameters

None.

Return Values

Undefined.

Throws

except-param

Location

batch-validate-params.monk

batch-write-file

Syntax

```
(batch-write-file Event_data)
```

Description

batch-write-file writes a record to a temporary (outbound) file in the style defined by the **Publish To External** section of the e*Way's configuration parameters (see **"Publish to External" on page 45** for more information).

Parameters

Name	Type	Description
Event_Data	string	Event data

Return Values

Undefined.

Throws

None.

Location

batch-proc-out.monk

disconnect-from-remote

Syntax

```
(disconnect-from-remote)
```

Description

disconnect-from-remote is the top-level function that disconnects the e*Way from the remote system.

Parameters

None.

Return Values

Undefined.

Throws

The function itself does not throw any exceptions, but it catches and logs exceptions thrown by other functions.

Location

batch-exchange-utils.monk

fetch-files-from-remote

Syntax

```
(fetch-files-from-remote)
```

Description

fetch-files-from-remote attempts to fetch from the external system all the files specified by the configuration parameters in the **Subscribe To External** section of the e*Way's configuration parameters (see **"Subscribe to External" on page 42** for more information).

Parameters

None.

Return Values

List
Returns the list of files fetched.

Throws

except-abort

Location

batch-exchange-utils.monk

fetch-named-files

Syntax

```
(fetch-named-files file_list)
```

Description

fetch-named-files attempts to fetch a list of files from the external system. The method used to perform the actual transfer of each file is specified by the **File Transfer Method** configuration parameter (see **"File Transfer Method" on page 41** for more information).

Parameters

Name	Type	Description
file_list	List	A list of files to be transferred from the external system.

Return Values

List
Returns a list of files successfully fetched.

Throws

except-method, **except-abort**

Location

batch-exchange-utils.monk

get-next-record

Syntax

```
(get-next-record)
```

Description

get-next-record reads the next available record from the files in the inbound temporary directory. If there are no more records in the current file, the next file is opened and read.

Parameters

None.

Return Values

Returns one of the following values:

String
If a record is available and can be read, the function returns the record read.

Boolean
If there are no more records available for reading, the function returns **#f** (false).

Throws

None.

Location

batch-exchange-utils.monk

get-next-record-current-file

Syntax

```
(get-next-record-current-file)
```

Description

get-next-record-current-file reads and returns the next record from the currently open file (in the inbound temporary directory).

Parameters

None.

Return Values

Returns one of the following values:

String
If a record is available and can be read, the function returns the record read.

Boolean
If there are no more records available for reading, the function returns **#f** (false).

Throws

None.

Location

batch-exchange-utils.monk

list-files-on-remote

Syntax

```
(list-files-on-remote)
```

Description

list-files-on-remote lists the files in the currently connected directory on the external system, using a command appropriate to the **File Transfer Method** configuration parameter (see **"File Transfer Method" on page 41**).

Parameters

None.

Return Values

List
Returns a list of files.

Throws

except-method

Location

batch-exchange-utils.monk

open-next-working-file

Syntax

```
(open-next-working-file)
```

Description

While the e*Way is reading temporary files in the inbound temporary directory, **open-next-working-file** closes the current file, then opens a handle on the next available file.

Parameters

None.

Return Values

Returns one of the following values:

String
If a record is available and can be read, the function returns the record read.

Boolean
If there are no more records available for reading, the function returns **#f** (false).

Throws

None.

Location

batch-exchange-utils.monk

persist-get-index

Syntax

```
(persist-get-index)
```

Description

persist-get-index retrieves the current file list index from the persistency file.

Parameters

None.

Return Values

Integer
Returns the file list index.

Throws

None.

Location

batch-persist.monk

persist-get-list

Syntax

```
(persist-get-list)
```

Description

persist-get-list retrieves the current file list from the persistency file.

Parameters

None.

Return Values

List
Returns the file list.

Throws

None.

Location

batch-persist.monk

persist-get-offset

Syntax

```
(persist-get-offset)
```

Description

persist-get-offset retrieves the current file position offset from the persistency file.

Parameters

None.

Return Values

Integer
Returns the file offset.

Throws

None.

Location

batch-persist.monk

persist-init

Syntax

```
(persist-init)
```

Description

persist-init opens the persistency file if the file is not already open, creating the file if necessary.

Parameters

None.

Return Values

Undefined.

Throws

None.

Location

batch-persist.monk

Additional Information

The persistency file is used when reading records from files in inbound data transfers. The default file name is **persist.dat**.

persist-read-number

Syntax

```
(persist-read-number string_size)
```

Description

persist-read-number reads a string of the size given in the input argument from the persistency file (persist.dat), and converts it to a numeric value.

Parameters

Name	Type	Description
string_size	integer	The size (in bytes) of the string to be read from the persistency file.

Return Values

Integer
Returns the numeric value of the string read from the persistency file.

Throws

None.

Location

batch-persist.monk

persist-update-index

Syntax

```
(persist-update-index index)
```

Description

persist-update-index updates the file list index in the persistency file

Parameters

Name	Type	Description
index	integer	The file list index to update

Return Values

Undefined.

Throws

None.

Location

batch-persist.monk

persist-update-list

Syntax

```
(persist-update-list file_list)
```

Description

persist-update-list updates the file list in the persistency file.

Parameters

Name	Type	Description
file_list	integer	The file list to update

Return Values

Undefined.

Throws

None.

Location

batch-persist.monk

persist-update-offset

Syntax

```
(persist-update-offset offset)
```

Description

persist-update-offset updates the file position offset in the persistency file.

Parameters

Name	Type	Description
offset	integer	The file position offset

Return Values

Undefined.

Throws

None.

Location

batch-persist.monk

persist-update-status

Syntax

```
(persist-update-status offset list_index file_list)
```

Description

persist-update-status updates all elements of the persistency file in a single function call.

Parameters

Name	Type	Description
offset	integer	The file position offset
list_index	integer	The file list index
file_list	List	The file list

Return Values

Undefined.

Throws

None.

Location

batch-persist.monk

persist-write-pad

Syntax

```
(persist-write-pad port text_string length)
```

Description

persist-write-pad writes the text to the output port, padded with leading spaces to the specified length.

Parameters

Name	Type	Description
port	Port	The output port
text_string	string	The text to be written
length	integer	The length of the string to be written, including padding (spaces)

Return Values

Undefined.

Throws

None.

Location

batch-persist.monk

post-transfer-hook

Syntax

```
(post-transfer-hook)
```

Description

post-transfer-hook sets a variable used by the **ftp-ext-connect** and **ftp-ext-verify** functions that describes the state of the connection. The function is called by **batch-exchange-data** immediately after the **disconnect-from-remote** function is called.

Parameters

None.

Return Values

Undefined.

Throws

None.

Location

batch-exchange-utils.monk

pre-transfer-hook

Syntax

```
(pre-transfer-hook)
```

Description

pre-transfer-hook sets a variable used by the **ftp-ext-connect** and **ftp-ext-verify** functions that describes the state of the connection. The function is called by **batch-exchange-data** immediately before the **connect-to-remote** function is called.

Parameters

None.

Return Values

Undefined.

Throws

None.

Location

batch-exchange-utils.monk

send-files-to-remote

Syntax

```
(send-files-to-remote file_list)
```

Description

send-files-to-remote attempts to send a list of files stored in the temporary outbound directory to the external system according to the method defined by the **File Transfer Method** parameter.

Parameters

Name	Type	Description
file_list	List	A list of files to be sent to the external system.

Return Values

List
Returns a list of the files that were sent successfully.

Throws

except-method, **except-abort**

Location

batch-exchange-utils.monk

string-is-proc?

Syntax

```
(string-is-proc? procedurename)
```

Description

string-is-proc? tests whether the specified string is the name of a Monk procedure.

Parameters

Name	Type	Description
procedurename	string	The string to test

Return Values

Boolean
Returns **#t** (true) if the specified string is the name of a Monk procedure; otherwise, returns **#f** (false).

Throws

None.

Location

batch-utils.monk

transfer-method?

Syntax

```
(transfer-method?)
```

Description

transfer-method? returns the transfer method established by the **File Transfer Method** parameter.

Parameters

None.

Return Values

Quoted Symbol

Returns one of the following quoted symbols:

‘METHOD_FTP	File transfer method **FTP**
‘METHOD_FILE	File transfer method **File Copy**
‘METHOD_UNKNOWN	Unknown method

Throws

None.

Location

batch-utils.monk

6.4 File Name Expansion Functions

These functions are used when converting special character sequences in a string to some other sequence. 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 file name expansion functions are

char-hex? on page 133

expand-char on page 134

expand-hex on page 135

expand-octal on page 136

expand-seqno on page 137

expand-string on page 138

expand-time on page 139

get-seqno on page 141

incr-seqno on page 142

set-seqno on page 143

char-hex?

Syntax

```
(char-hex? chr)
```

Description

char-hex? determines whether a character is a valid hexadecimal character (that is, in the range 0 through 9, A through F, or a through f).

Parameters

Name	Type	Description
chr	character	The character to test

Return Values

Boolean
Returns **#t** (true) if the tested character is a valid hexadecimal character; otherwise, returns **#f** (false).

Throws

None.

Location

batch-exchange-utils.monk

expand-char

Syntax

```
(expand-char chr)
```

Description

expand-char converts certain special characters from their escaped representation to their ASCII character.

Parameters

Name	Type	Description
chr	character	The character to convert

The characters **expand-char** can convert are as follows:

Input character	Converts To
0	Null character
a	Audible bell
b	Backspace
f	Form feed
n	New line
r	Carriage return
t	Tab
v	Vertical tab

Return Values

Character
Returns a character (see the conversion table above).

Throws

None.

Location

batch-exchange-utils.monk

expand-hex

Syntax

```
(expand-hex hex_string)
```

Description

expand-hex converts two hexadecimal characters (0 through 9, A through F) to the ASCII character that they represent.

Parameters

Name	Type	Description
hex_string	string	A two-character string to be converted to an ASCII character.

Return Values

String
Returns a single ASCII character.

Throws

None.

Location

batch-exchange-utils.monk

expand-octal

Syntax

```
(expand-octal octal_string)
```

Description

expand-octal converts three octal digits (0 through 9) to a single ASCII character.

Parameters

Name	Type	Description
octal_string	string	A three-character string to be converted to an ASCII character.

Return Values

String
Returns a single ASCII character.

Throws

None.

Location

batch-exchange-utils.monk

expand-seqno

Syntax

```
(expand-seqno padding)
```

Description

expand-seqno inserts a sequence number into a string, padded with the number of zeros specified in the function call. The sequence number is incremented when this function is called.

Parameters

Name	Type	Description
padding	string	The number of zeros to add as padding to the sequence number.

Return Values

String
Returns the current sequence number, zero-padded as specified.

Throws

None.

Location

batch-utils.monk

expand-string

Syntax

```
(expand-string string)
```

Description

expand-string searches an arbitrary string for known special character sequences and replaces them with the strings they represent. This function calls, as appropriate, **expand-octal**, **expand-hex**, **expand-char**, **expand-seqno** (with zero padding), or **expand-time**.

Parameters

Name	Type	Description
string	string	The string to expand

Return Values

String
Returns the expanded string.

Throws

None.

Location

batch-utils.monk

expand-time

Syntax

```
(expand-time chr)
```

Description

expand-time returns an expansion of the supplied character as described in the **C strftime()** function call.

Parameters

Name	Type	Description
chr	character	A character representing a strftime() format

The supported formats are:

Character	Format
a	Abbreviated weekday
A	Full weekday
b	Abbreviated month name
B	Full month name
c	Date and time representation
d	Day of the month (01 through 31)
H	Hour (00 through 23)
I	Hour (01 through 12)
j	Day of the year (001 through 366)
m	Month (01 through 12)
M	Minute (00 through 59)
p	AM or PM
S	Seconds (00 through 61)
U	Week number, starting from the first Sunday
W	Week number, starting from the first Monday
w	Day of the week (Sunday = 0)
x	Date representation
X	Time representation
y	Year (00 through 99)
Y	Year, including century
Z	Time zone

Return Values

String
Returns a string containing time or date information.

Throws

None.

Location

batch-utils.monk

get-seqno

Syntax

```
(get-seqno)
```

Description

get-seqno reads the current sequence number from persistent storage (the text file **sequence.dat**) and returns it. If this file does not exist, the sequence number is taken from the configuration variable **cfg-seq-no-start**.

Parameters

None.

Return Values

String

Returns the current sequence number as a string.

Throws

None.

Location

batch-utils.monk

incr-seqno

Syntax

```
(incr-seqno)
```

Description

incr-seqno obtains the current sequence number through a call to the function **ftp-get-seqno**, and increments it by one. If the new sequence number is greater than that specified by the configuration variable **Max Sequence Number**, the number is reset to the value of the configuration variable **Start Sequence Number**.

The configuration number is then written to persistent storage in the file **sequence.dat**. This file will be created if it does not already exist and overwritten if it does.

Parameters

None.

Return Values

String

Returns a string containing a sequence number.

Throws

None.

Location

batch-utils.monk

set-seqno

Syntax

```
(set-seqno new_number)
```

Description

set-seqno sets the current sequence number to the specified value. The value is not checked, and therefore could be set outside the range defined by the configuration variables **Start Sequence Number** and **Max Sequence Number**.

Parameters

Name	Type	Description
new_number	integer	The value to which to set the sequence number

Return Values

String
Returns a string that contains a sequence number.

Throws

None.

Location

batch-utils.monk

6.5 Post-transfer Routines

These functions are invoked after either an inbound or outbound transfer has taken place. They specify actions that are defined by the settings of configuration variables, which will be performed on the local temporary file or upon the external file. Some of these operations are likely to be undesirable depending on the direction of transfer, but this is a configuration issue.

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 post-transfer routines are:

batch-local-post-transfer on page 144.

batch-rmt-post-transfer on page 145.

local-post-transfer on page 146.

batch-local-post-transfer

Syntax

```
(batch-local-post-transfer local_filename)
```

Description

batch-local-post-transfer performs the relevant post-transfer operation on a specified local file.

Parameters

Name	Type	Description
local_filename	string	The name of a local file

Return Values

Undefined.

Throws

except-local-op

Location

batch-post-transfer.monk

batch-rmt-post-transfer

Syntax

```
(batch-rmt-post-transfer rmt_filename)
```

Description

batch-rmt-post-transfer performs the relevant post-transfer operation on the specified remote file.

Parameters

Name	Type	Description
rmt_filename	string	The name of a remote file

Return Values

Undefined.

Throws

except-method

Location

batch-post-transfer.monk

local-post-transfer

Syntax

```
(local-post-transfer direction command archiveDirectory filename)
```

Description

local-post-transfer performs the relevant post-transfer operation on a local system, after the transfer of working files is complete.

Parameters

Name	Type	Description
direction	string (“inbound” or “outbound”)	Indicates whether the e*Way is inbound or outbound.
command	string	The command that the e*Way will execute after a successful file transfer (for example, Delete, Rename, or Archive.
archiveDirectory	string	Specifies the local directory in which to archive the working files.
filename	string	The name of a local file.

Return Values

Undefined.

Throws

except-local-op

Location

local-post-transfer.monk

6.6 File Copy Transfer Functions

These functions execute file-based “copy” transfers. 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.

Note: *Many of the functions in this section are place-holders for user-supplied customizations. If you need to add functionality to these place-holder functions, be sure not to change the arguments required nor the type of value returned.*

The file copy transfer functions are

file-ext-connect on page 148

file-ext-shutdown on page 149

file-ext-connect

Syntax

```
(file-ext-connect)
```

Description

file-ext-connect opens a connection to the external system.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

None.

Location

file-ext-connect.monk

file-ext-shutdown

Syntax

```
(file-ext-shutdown)
```

Description

file-ext-shutdown closes the connection to an external system.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

None.

Location

file-ext-shutdown.monk

file-ext-verify

Syntax

```
(file-ext-verify)
```

Description

file-ext-verify verifies the connection to an external system.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

None.

Location

file-ext-verify.monk

file-fetch

Syntax

```
(file-fetch filename)
```

Description

file-fetch fetches a file from a remote system.

Parameters

Name	Type	Description
filename	string	The name of a file

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

except-transfer, plus the name of the file.

Location

file-fetch.monk

file-fetch-path

Syntax

```
(file-fetch-path remoteDirectory filename)
```

Description

file-fetch-path-list fetches a file from a specified location on a remote system.

Parameters

Name	Type	Description
remoteDirectory	string	The complete directory path on the remote system where the file to be fetched resides.
filename	string	The name of a file.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

except-transfer, plus the name of the file.

Location

file-fetch.monk

file-init

Syntax

```
(file-init)
```

Description

file-init initializes the Monk environment for file-based-transfer functions.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

None.

Location

file-init.monk

file-remote-path-list

Syntax

```
(file-remote-path-list remoteDirectory remoteFileRegexp)
```

Description

file-remote-path-list lists the files within a specified location on an external system.

Parameters

Name	Type	Description
remoteDirectory	string	The complete directory path on the remote system where the files to be listed reside. A regular expression is not accepted.
remoteFileRegexp	string	A regular expression that describes the files to be listed. (See **"Remote File Regexp" on page 42** and **"Using Special Characters" on page 53**.)

Return Values

List
Returns a list of files.

Throws

except-rmt-list

Location

file-remote-path-list.monk

file-rmt-list

Syntax

```
(file-rmt-list)
```

Description

file-rmt-list lists the files in the external source directory.

Parameters

None.

Return Values

List
Returns a list of files.

Throws

except-rmt-list

Location

file-rmt-list.monk

file-rmt-post-transfer

Syntax

```
(file-rmt-post-transfer filename)
```

Description

file-rmt-post-transfer performs post-transfer operations on the named file, depending on the setting of the **Remote Command After Transfer** configuration parameter.

Parameters

Name	Type	Description
filename	string	The name of a file

Return Values

Boolean

Returns **#t** (true) if the function evaluates the **Remote Command After Transfer** configuration parameter's to be **None**, or if the function succeeds. The exception **except-rmt-op** is thrown if the function fails, or if an unrecognized transfer option (other than none, archive, rename or delete) is selected. See **"Remote Command After Transfer" on page 44** for more information.

Throws

except-rmt-op.

Location

file-rmt-post-transfer.monk

file-send

Syntax

```
(file-send filename)
```

Description

file-send sends the specified file to the external system.

Parameters

Name	Type	Description
filename	string	The name of a file

Return Values

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

Throws

except-transfer, plus the name of the file.

Location

file-send.monk

file-send-path-file

Syntax

```
(file-send-path-file appendOverwrite localFilename remoteDirectory
                     remoteFilename)
```

Description

file-send-path-file sends the specified file to a specific directory on an external system.

Parameters

Name	Type	Description
appendOverwrite	string	Specifies whether to append the records in the file being transferred to the existing file on the external system, or to overwrite the existing file on the external system with the file being transferred.
localFilename	string	The name of the file being sent to the external system.
remoteDirectory	string	The path name at the remote location to which the file is to be sent.
remoteFilename	string	The name of the file on the external system that is being overwritten or appended.

Return Values

Boolean

Returns **#t** (true) if the transfer succeeds; otherwise, returns **#f** (false).

Throws

except-transfer, plus the name of the file.

Location

file-send-path-file.monk

file-startup

Syntax

```
(file-startup)
```

Description

file-startup performs startup functions specific to file-based transfers.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

None.

Location

file-startup.monk

file-validate-params

Syntax

```
(file-validate-params)
```

Description

file-validate-params validates the configuration parameters specific to file-based transfers.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

None.

Location

file-validate-params.monk

6.7 FTP Transfer Functions

The functions in this section control the FTP connection and perform basic operations such as send, list, and fetch. 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 FTP transfer functions are

ftp-do-connect on page 161	**ftp-init** on page 167
ftp-ext-connect on page 162	**ftp-rmt-list** on page 168
ftp-ext-shutdown on page 163	**ftp-rmt-post-transfer** on page 169
ftp-ext-verify on page 164	**ftp-send** on page 170
ftp-fetch on page 165	**ftp-startup** on page 171
ftp-heuristic-download on page 166	**ftp-validate-params** on page 172

ftp-do-connect

Syntax

```
(ftp-do-connect)
```

Description

ftp-do-connect is a helper function related to **ftp-ext-connect**, which actually makes the connection to the remote host.

Parameters

None.

Return Values

Undefined.

Throws

None.

Location

ftp-ext-connect.monk

ftp-ext-connect

Syntax

```
(ftp-ext-connect)
```

Description

ftp-ext-connect opens an FTP connection to an external system.

Parameters

None.

Return Values

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

Throws

except-connect

Location

ftp-ext-connect.monk

ftp-ext-shutdown

Syntax

```
(ftp-ext-shutdown)
```

Description

ftp-ext-shutdown closes the FTP connection to the external system.

Parameters

None.

Return Values

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

Throws

None.

Location

ftp-ext-shutdown.monk

ftp-ext-verify

Syntax

```
(ftp-ext-verify)
```

Description

ftp-ext-verify verifies that the FTP connection to the external system is still operating properly.

Parameters

None.

Return Values

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

Throws

None.

Location

ftp-ext-verify.monk

ftp-fetch

Syntax

```
(ftp-fetch *filename*)
```

Description

ftp-fetch retrieves the specified file from the external system.

Parameters

Name	Type	Description
filename	string	The name of a file

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

except-transfer, plus the file name.

Location

ftp-fetch.monk

ftp-heuristic-download

Syntax

```
(ftp-heuristic-download)
```

Description

ftp-heuristic-download downloads the file **FtpHeuristics.cfg** from the e*Gate Registry.

Parameters

None.

Return Values

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

Throws

None.

Location

ftp-init.monk

ftp-init

Syntax

```
(ftp-init)
```

Description

ftp-init initializes the Monk environment for FTP-transfer functions.

Parameters

None.

Return Values

Boolean
Returns **#t** (true) if the initialization operations succeed; otherwise, returns **#f** (false).

Throws

None.

Location

ftp-init.monk

ftp-rmt-list

Syntax

```
(ftp-rmt-list)
```

Description

ftp-rmt-list returns a list of files in the external source directory.

Parameters

None.

Return Values

List
Returns a list of files.

Throws

except-rmt-list

Location

ftp-rmt-list.monk

ftp-rmt-post-transfer

Syntax

```
(ftp-rmt-post-transfer filename)
```

Description

ftp-rmt-post-transfer performs post-transfer operations on the named file, depending on the setting of the **Remote Command After Transfer** configuration parameter.

Parameters

Name	Type	Description
filename	string	The name of a file

Return Values

Boolean

Returns **#t** (true) if the function evaluates the **Remote Command After Transfer** configuration parameter's to be "none" or if the function succeeds; **#f** (false) if an unrecognized transfer option (other than none, archive, rename or delete) is selected or if the function fails. See **"Remote Command After Transfer" on page 44** for more information.

Throws

except-rmt-op

Location

ftp-rmt-post-transfer.monk

ftp-send

Syntax

```
(ftp-send filename)
```

Description

ftp-send sends the specified file to the external system.

Parameters

Name	Type	Description
filename	string	The name of a file

Return Values

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

Throws

except-transfer, plus the file name.

Location

ftp-send.monk

ftp-startup

Syntax

```
(ftp-startup)
```

Description

ftp-startup performs startup functions necessary for FTP transfers, such as establishing the required handles.

Parameters

None.

Return Values

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

Throws

None.

Location

ftp-startup.monk

ftp-validate-params

Syntax

```
(ftp-validate-params)
```

Description

ftp-validate-params validates configuration parameters specific to FTP transfers.

Parameters

None.

Return Values

Undefined.

Throws

except-param

Location

ftp-validate-params.monk

6.8 Advanced FTP Functions

The functions in this section perform advanced FTP 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 advanced FTP functions are:

ftp-append-file on page 174

ftp-append-path on page 175

ftp-archive on page 176

ftp-archive-path on page 177

ftp-capture-data on page 178

ftp-change-dir on page 179

ftp-close on page 180

ftp-connect on page 181

ftp-create-handle on page 183

ftp-disconnect on page 184

ftp-delete on page 185

ftp-delete-path on page 186

ftp-fetch-path on page 187

ftp-append-file

Syntax

```
(ftp-append-file handle local_file remote_file)
```

Description

ftp-append-file sends a local file to the external host with the given external file name. This function appends to the target file, or creates a new file if the target file does not exist.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
local_file	string	The name of a file
remote_file	string	The name of a file

Return Values

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

Throws

ftp-append-file will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 508

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-append-path

Syntax

```
(ftp-append-path handle local_file remote_dir remote_file)
```

Description

ftp-append-path sends a local file to the external host with the specified external file name, to the specified directory. This function appends to the target file if it exists, or creates a new file. **ftp-append-path** is functionally identical to **ftp-append-file**, except that the FTP Heuristics database is used to generate a correct path name for the external file.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
local_file	string	The name of a file
remote_dir	string	The name of a directory
remote_file	string	The name of a file

Return Values

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

Throws

ftp-append-path will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-archive

Syntax

```
(ftp-archive handle filename directory)
```

Description

ftp-archive moves an external file to a different directory on the external host.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
filename	string	The name of a file
directory	string	The name of a directory

Return Values

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

Throws

ftp-archive will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Note: *ftp-archive is not supported on heuristics MVS GDG. See* **Operating System or File Type Selection** *on page 56. In addition, MVS does not allow partitioned data sets to be renamed to another partitioned data set.*

ftp-archive-path

Syntax

```
(ftp-archive-path handle old_dir filename new_dir)
```

Description

ftp-archive-path moves a file on the external system to a different directory on the external system. The FTP Heuristics database is used to generate the correct path for the external file.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
old_dir	string	The name of a directory
filename	string	The name of a file
new_dir	string	The name of a directory

Return Values

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

Throws

ftp-archive-path will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 506

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Note: *ftp-archive-path is not supported on heuristics MVS GDG. See* **Operating System or File Type Selection** *on page 56. In addition, MVS does not allow partitioned data sets to be renamed to another partitioned data set.*

ftp-capture-data

Syntax

```
(ftp-capture-data handle filename)
```

Description

ftp-capture-data reads the data from a data port previously opened with **ftp-open-data-port**, and captures said data to the file specified. If the file already exists, it is overwritten.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
filename	string	The name of a file

Return Values

Boolean

Returns **#t** (true) if the operation succeeds; otherwise, returns **#f** (false).

Throws

ftp-capture-data will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-change-dir

Syntax

```
(ftp-change-dir handle directory)
```

Description

ftp-change-dir changes to the specified directory on the external host.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
directory	string	The name of a directory

Return Values

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

Throws

ftp-change-dir will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 506

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 30.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-close

Syntax

```
(ftp-close handle)
```

Description

ftp-close closes the FTP connection on the specified handle.

Parameters

Name	Type	Description
handle	Handle	The FTP handle

Return Values

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

Throws

ftp-close will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-connect

Syntax

```
ftp-connect ftpHandle socksServerName socksServerPort SocksMethod
            SocksUserName Sockspassword ftpServerName ftpServerPort
            userName encryptedPassword
```

Description

ftp-connect makes a connection to a FTP server through a SOCKS host, and allows for a configurable FTP server port number. If the FTP server port is an empty string, the e*Way uses the default port number 21.

If SOCKS is not used, an empty string is passed for both the SOCKS server name and the SOCKS server port.

Parameters

Name	Type	Description
ftpHandle	string	The FTP handle.
socksServerName	string	A valid name for the SOCKS server.
socksServerPort	integer	The port number to use on the SOCKS server for connection.
socksMethod	string	Indicates the Authentication method, if any, for connecting to the SOCKS server.
socksUserName	string	The User Name to be used for authentication when connecting to the SOCKS server.
sockspassword	encrypted string	The encrypted password to be used for authentication when connecting to the SOCKS server.
ftpServerName	string	A valid name for the FTP server
userName	string	The User Name to be used for authentication when connecting to the FTP server.
encryptedPassword	encrypted string	The encrypted password to be used for authentication when connecting to the FTP server.

Return Values

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

Throws

ftp-connect will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

ftp-connect.monk

ftp-create-handle

Syntax

```
(ftp-create-handle host-type)
```

Description

ftp-create-handle creates a new FTP handle for the specified host type. The host type must be valid, and specified in the Ftp Heuristics configuration file.

You must supply the argument for this function; there is no default.

Parameters

Name	Type	Description
host-type	string	A valid host type

Return Values

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

Throws

ftp-create-handle will throw the following exceptions:

$Ftp-Exception-Catastrophic, E_STR 502.

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-disconnect

Syntax

```
(ftp-disconnect ftpHandle)
```

Description

ftp-disconnect closes the FTP connection on the specified handle.

Parameters

Name	Type	Description
ftpHandle	string	The FTP handle

Return Values

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

Throws

ftp-close will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

ftp-disconnect.monk

ftp-delete

Syntax

```
(ftp-delete handle filename)
```

Description

ftp-delete deletes a file from the external system.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
filename	string	The name of a file

Return Values

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

Throws

ftp-delete will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Note: *ftp-delete is not supported on heuristics MVS GDG. See* **Operating System or File Type Selection** *on page 56.*

ftp-delete-path

Syntax

```
(ftp-delete-path handle remote_dir remote_file)
```

Definition

ftp-delete-path deletes a file from a named directory on the external system. The FTP Heuristics database is used to generate a correct path to the external file' location.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
remote_dir	string	The name of a directory
remote_file	string	The name of a file

Return Values

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

Throws

ftp-delete-path will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Note: *ftp-delete-path is not supported on heuristics MVS GDG. See* **Operating System or File Type Selection** *on page 56.*

ftp-fetch-path

Syntax

```
(ftp-fetch-path ftphandle ftpMode remoteDirectory filename)
```

Description

ftp-fetch-path-list fetches a file, through a FTP connection, from a specified location on a remote system.

Parameters

Name	Type	Description
ftpHandle	string	The FTP handle.
ftpMode	string	The FTP mode.
remoteDirectory	string	The complete directory path on the remote system where the file to be fetched resides.
filename	string	The name of a file.

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

except-transfer, plus the name of the file.

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 508

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

ftp-fetch-path.monk

ftp-get-file

Syntax

```
(ftp-get-file handle remote_file local_file)
```

Description

ftp-get-file retrieves the specified file from the external host and stores it in the specified local file.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
remote_file	string	The name of a file
local_file	string	The name of a file

Return Values

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

Throws

ftp-get-file will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-get-last-response

Syntax

```
(ftp-get-last-response handle)
```

Description

ftp-get-last-response returns the full textual response of the last FTP transaction.

Parameters

Name	Type	Description
handle	Handle	The FTP handle

Return Values

String
Returns the external system’s response.

Throws

ftp-get-last-response will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-get-last-result-code

Syntax

```
(ftp-get-last-result-code handle)
```

Description

ftp-get-last-result-code returns the result code of the last FTP transaction. See RFC 959 for a description of the values that may be returned in this function.

Parameters

Name	Type	Description
handle	Handle	The FTP handle

Return Values

Integer
Returns the external system's response.

Throws

ftp-get-last-result-code will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-get-path

Syntax

```
(ftp-get-path handle remote_dir remote_file local_file)
```

Description

ftp-get-path retrieves a file from a named directory on the external system. This is functionally identical to **ftp-get-file**, except that the FTP Heuristics database is used to generate a correct path name for the external file.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
remote_dir	string	The name of a directory
remote_file	string	The name of a file
local_file	string	The name of a file

Return Values

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

Throws

ftp-get-path will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 508

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-handle?

Syntax

```
(ftp-handle? handle)
```

Description

ftp-handle? determines whether the specified handle is a valid FTP handle.

Parameters

Name	Type	Description
handle	Handle	An FTP handle

Return Values

Boolean
Returns **#t** (true) if the handle is valid; otherwise, returns **#f** (false).

Throws

ftp-handle? will throw the following exceptions:

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Additional Information

The fact that a file is of the same size on both occasions does not imply that it is stable. This function and **ftp-set-compare-time** are provided for compatibility purposes only.

ftp-list-files

Syntax

```
(ftp-list-files handle directory regexp_mask)
```

Description

ftp-list-files uses the FTP Heuristics to retrieve the list of the files, in the specified directory, that match the given regular expression.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
directory	string	The name of a directory
regexp_mask	string	A regular expression

Return Values

Returns one of the following values:

List
Returns a list of files.

Boolean
Returns **#f** (false) when it fails to find the list of files that match the given regular expression.

Throws

ftp-list-files will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 501.

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Examples

```
(define file-list (ftp-list-files “srcdir” “*.txt”))
```

ftp-list-raw

Syntax

```
(ftp-list-raw handle directory filename_regexp)
```

Description

ftp-list-raw performs a "LIST" command on the external system, using the specified directory and file name regular expression. The reply from the FTP server is returned as a list of lines, so that a Monk programmer can parse the output in any way that may be required.

Parameters

Name	Type	Description
handle	handle	The FTP handle
directory	string	The name of a directory
filename_regexp	string	A regular expression

Return Values

List
Returns a list of lines.

Throws

ftp-list-raw will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-login

Syntax

```
(ftp-login handle username encryptedpwd)
```

Description

ftp-login performs the FTP login sequence for the host previously opened on the current ftp handle.

Parameters

Name	Type	Description
handle	handle	The FTP handle
username	string	A valid username
encryptedpwd	string	The encrypted password corresponding to the specified username

Return Values

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

Throws

ftp-login will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 505

$Ftp-Exception-Invalid-Arg, E_STR 504

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Additional Information

The following Monk environment variables contain the user name and password specified in the e*Way Editor:

```
EXTERNAL_HOST_SETUP_ENCRYPTED_PASSWORD
EXTERNAL_HOST_SETUP_USER_NAME
```

See **"User Name" on page 41** and **"Encrypted Password" on page 41** for more information on these variables. If the **ftp-login** function is called within the Batch e*Way's Monk environment, you can obtain the required username and password information from those variables. For example,

```
(ftp-login handle EXTERNAL_HOST_SETUP_ENCRYPTED_PASSWORD
        EXTERNAL_HOST_SETUP_USER_NAME)
```

You may also use the **(encrypt-password)** function to generate an encrypted password. For example,

```
(ftp-login handle "Administrator"
        (encrypt-password "Administrator" "Admin-password"))
```

(encrypt-password) requires two string parameters (the user name and password), and returns the encrypted password as a string. The **(encrypt-password)** function is defined in the following file:

/monk_library/monkext/monkext.monk

You must load this file to use **(encrypt-password)**. To load the **monkext.monk** file within the e*Way's Monk environment, add the directory **/monk_library/monkext/** to the list of Auxiliary Library Directories. See **"Auxiliary Library Directories" on page 34** for more information.

ftp-make-dir

Syntax

```
(ftp-make-dir handle directory)
```

Description

ftp-make-dir creates a directory on the external system.

Parameters

Name	Type	Description
handle	handle	The FTP handle
directory	string	A valid directory name

Return Values

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

Throws

ftp-make-dir will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 506

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 30.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-open-data-port

Syntax

```
(ftp-open-data-port handle)
```

Description

ftp-open-data-port creates opens a TCP/IP port.

Parameters

Name	Type	Description
handle	handle	The FTP handle

Return Values

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

Throws

ftp-open-data-port will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-open-host

Syntax

```
(ftp-open-host handle hostname)
```

Description

ftp-open-host opens a command connection to the FTP port of the given host name.

Parameters

Name	Type	Description
handle	handle	The FTP handle
hostname	string	A valid hostname

Return Values

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

Throws

ftp-open-host will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 503.

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-open-host-through-SOCKS

Syntax

```
(ftp-open-host-through-SOCKS ftpHandle socksServerName
                              socksServerPort SocksMethod
                              SocksUserName Sockspassword
                              ftpServerName)
```

Description

ftp-open-host-through-SOCKS connects to the specified FTP Host through the SOCKS Host.

Parameters

Name	Type	Description
ftpHandle	string	The FTP handle.
socksServerName	string	A valid name for the SOCKS server.
socksServerPort	integer	The port number to use on the SOCKS server for connection.
socksMethod	string	Indicates the Authentication method, if any, for connecting to the SOCKS server.
socksUserName	string	The User Name to be used for authentication when connecting to the SOCKS server.
sockspassword	encrypted string	The encrypted password to be used for authentication when connecting to the SOCKS server.
ftpServerName	string	A valid name for the FTP server.

Return Values

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

Throws

ftp-open-host-through-SOCKS will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-put-file

Syntax

```
(ftp-put-file handle local_file remote_file)
```

Description

ftp-put-file sends the specified local file to the external host, saving it under the specified remote file name. A target file of the same name will be overwritten.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
local_file	string	The local file name
remote_file	string	The remote file name

Return Values

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

Throws

ftp-put-file will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 508

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-put-path

Syntax

```
(ftp-put-path handle local_file remote_dir remote_file)
```

Description

ftp-put-path sends a file from the local system to a named directory on the external system. This is functionally identical to **ftp-put-file**, except that the FTP Heuristics database is used to generate a correct path name for the external file.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
local_file	string	The local file name
remote_dir	string	The remote directory name
remote_file	string	The remote file name

Return Values

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

Throws

ftp-put-path will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Generic, E_STR 507

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-remote-path-list

Syntax

```
(ftp-remote-path-list ftpHandle remoteDirectory remoteFileRegexp)
```

Description

ftp-remote-path-list lists the files within a specified location on an external system, through a FTP connection.

Parameters

Name	Type	Description
ftpHandle	string	The FTP handle
remoteDirectory	string	The complete directory path on the remote system where the files to be listed resides.
remoteFileRegexp	string	A regular expression that describes files to be listed. (See **"Remote File Regexp" on page 42** and **"Using Special Characters" on page 53**.)

Return Values

List

Returns a list of files.

Throws

except-rmt-list

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

ftp-remote-path-list.monk

ftp-rename

Syntax

```
(ftp-rename handle old_name new_name)
```

Description

ftp-rename renames a file on the external host.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
old_name	string	The current file name
new_name	string	A valid file name

Return Values

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

Throws

ftp-rename will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Note: *Not all FTP daemons support this command.*
ftp-rename is not supported on heuristics MVS GDG. See **Operating System or File Type Selection** *on page 56. In addition, MVS does not allow partitioned data sets to be renamed to another partitioned data set.*

ftp-rename-path

Syntax

```
(ftp-rename-path handle remote_dir old_name new_name)
```

Description

ftp-rename-path renames a file on the external system. The directory in which the file is located is passed as a parameter. The FTP heuristics database is used to generate a correct path name for the external file.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
remote_dir	string	The remote directory name
old_name	string	The current file name
new_name	string	A valid file name

Return Values

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

Throws

ftp-rename-path will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Note: *ftp-rename-path is not supported on heuristics MVS GDG. See* **Operating System or File Type Selection** *on page 56. In addition, MVS does not allow partitioned data sets to be renamed to another partitioned data set.*

ftp-send-command

Syntax

```
(ftp-send-command handle command)
```

Description

ftp-send-command enables the developer to send any command to the external FTP server. The results of the command should be read with **ftp-get-last-result-code** and **ftp-get-last-response**.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
command	string	A valid FTP command

Return Values

Boolean
Returns **#t** (true) if the operation succeeds; otherwise, returns **#f** (false). This function does not return the results of the FTP command itself.

Throws

ftp-send-command will throw the following exceptions:

$Ftp-Exception-Generic, E_STR 510.

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-send-path-file

Syntax

```
(ftp-send-path-file ftpHandle ftpMode appendOverwrite localFilename
                    remoteDirectory remoteFilename)
```

Description

ftp-send-path-file sends the specified file to a specific directory on an external system through a FTP connection.

Parameters

Name	Type	Description
ftpHandle	string	The FTP handle.
ftpMode	string	The FTP mode.
appendOverwrite	string	Specifies whether to append the records in the file being transferred to the existing file on the external system, or to overwrite the existing file on the external system with the file being transferred.
localFilename	string	The name of the file being sent to the external system.
remoteDirectory	string	The path name at the remote location to which the file is to be sent.
remoteFilename	string	The name of the file on the external system that is being overwritten or appended.

Return Values

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

Throws

except-transfer, plus the name of the file.

$Ftp-Exception-Generic, E_STR 509

$Ftp-Exception-Invalid-Arg, E_STR 500

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

file-send-path-file.monk

ftp-send-reply-immediate

Syntax

```
(ftp-send-reply-immediate handle flag)
```

Description

ftp-send-reply-immediate sets a Boolean flag. When the flag is set to **#t**, this function prevents the FTP ***.dll** file from waiting for a reply from the command port before starting a data transfer. The default for this flag is **#f**.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
flag	Boolean	The value of the flag

Return Values

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

Throws

ftp-send-reply-immediate will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

Exception-InvalidArg, E_STR 29.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-set-compare-time

Syntax

```
(ftp-set-compare-time handle seconds)
```

Description

ftp-set-compare-time sets the time between file listings for size comparison to the supplied number of seconds. See **Additional Information** on page 192 for more information.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
seconds	integer	A non-zero positive integer

Return Values

Boolean
Returns **#t** (true) under all circumstances.

Throws

ftp-set-compare-time will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

Exception-InvalidArg, E_STR 29.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-set-mode

Syntax

```
(ftp-set-mode handle mode)
```

Description

ftp-set-mode sets the transfer mode to either **A** for ASCII, **E** for EBCDIC, or **I** for image (binary).

Parameters

Name	Type	Description
handle	Handle	The FTP handle
mode	character	A, E, or I

Return Values

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

Throws

ftp-set-port will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

Exception-InvalidArg, E_STR 30.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

Additional Information

The mode selected will produce different results, depending on the type of data transferred, and the types of systems involved. The table below illustrates the possible different configurations of systems, data, and modes, with the corresponding results.

Configuration	Mode	Results
Batch e*Way on ASCII machine retrieving data from an EBCDIC machine.	ASCII	Data converts to ASCII which can be read on ASCII machine.
	EBCDIC	Data converts to ASCII which can be read on ASCII machine.
	Image	Data remains in EBCDIC.
Batch e*Way on ASCII machine retrieving data from an ASCII machine.	ASCII	Data remains in ASCII.
	EBCDIC	Data converts to unreadable format.
	Image	Data will be in ASCII.

ftp-set-port

Syntax

```
(ftp-set-port handle port)
```

Description

ftp-set-port sets the FTP port number. The default port is 21 if this port is not set.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
port	integer	A positive integer

Return Values

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

Throws

ftp-set-port will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

Exception-InvalidArg, E_STR 29.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-set-SOCKS-host

Syntax

```
(ftp-set-SOCKS-host handle SOCKS-hostname)
```

Description

ftp-set-SOCKS-host sets the host name of the SOCKS server.

Note: *This function is for backwards compatibility only. If you are using SOCKS version 5, you should use* **ftp-open-host-through-SOCKS** *on page 200.*

Parameters

Name	Type	Description
handle	Handle	The FTP handle
SOCKS-hostname	string	A valid host name

Return Values

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

Throws

ftp-set-SOCKS-host will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

$Ftp-Exception-Invalid-Arg, E_STR 503.

Exception-InvalidArg, E_STR 39.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-set-SOCKS-port

Syntax

```
(ftp-set-SOCKS-port handle SOCKS-port)
```

Description

ftp-set-SOCKS-port sets the port number through which to connect to the SOCKS server. When this SOCKS port is set, the FTP server is connected through the SOCKS server.

Note: *This function is for backwards compatibility only. If you are using SOCKS version 5, you should use* **ftp-open-host-through-SOCKS** *on page 200.*

Parameters

Name	Type	Description
handle	Handle	The FTP handle
SOCKS-port	integer	A positive integer

Return Values

Boolean

Returns **#t** (true) if the operation succeeds; otherwise, returns **#f** (false).

Throws

ftp-set-SOCKS-port will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

Exception-InvalidArg, E_STR 29.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

ftp-set-timeout

Syntax

```
(ftp-set-timeout handle time)
```

Description

ftp-set-timeout sets the number of seconds to wait for a response from the external FTP host or that a data transfer can stall.

Parameters

Name	Type	Description
handle	Handle	The FTP handle
time	integer	A non-zero positive integer

Return Values

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

Throws

ftp-set-timeout will throw the following exceptions:

$Ftp-Exception-Invalid-Arg, E_STR 500.

Exception-InvalidArg, E_STR 29.

Exception-InvalidArg E_STR 12.

See the Table, **Advanced FTP Exceptions** on page 215, for details about these exceptions.

Location

stc_ewftp.dll

6.8.1 Advanced FTP Function Exceptions

The following table shows details of the exceptions which the advanced FTP functions may throw.

Table 9 Advanced FTP Exceptions

symbol	category	E_STR	string	Reason
$Ftp-Exception-Generic	-51	510	argument %d - \"%s\" - must be valid Command.	Command is empty string.
$Ftp-Exception-Generic	-51	509	argument %d - \"%s\" - must be valid File name.	The file name is an empty string.
$Ftp-Exception-Generic	-51	508	argument %d - \"%s\" - must be valid Local Path.	Remote path is an empty string.
$Ftp-Exception-Generic	-51	507	argument %d - \"%s\" - must be valid Local Path.	Local path is an empty string.
$Ftp-Exception-Generic	-51	506	argument %d - \"%s\" - must be valid Directory.	Directory path is an empty string.
$Ftp-Exception-Invalid-Arg	-52	505	argument %d - \"%s\" - must be valid User.	User name is an empty string.
$Ftp-Exception-Invalid-Arg	-52	504	"argument %d - \"%s\" - must be password."	Password is an empty string.
$Ftp-Exception-Invalid-Arg	-52	503	argument %d - \"%s\" - must be valid Host name.	Host name is an empty string.
$Ftp-Exception-Catastrophic	-52	502	"Failed to create new FTP session handle."	Failed to create FTP handle.
$Ftp-Exception-Invalid-Arg	-52	501	"argument %d - \"%s\" - must be valid \filter with length in range of 1-255."	File filter is an empty string.
$Ftp-Exception-Invalid-Arg	-52	500	argument %d must be a valid FTP handle.	FTP handle is invalid.
Exception-InvalidArg	-10	39	argument %u must be a string	The argument must be a string.
Exception-InvalidArg	-10	30	%s: argument %u must be a char.	Mode must be a character.

symbol	category	E_STR	string	Reason
Exception-InvalidArg	-10	29	argument %u must be an integer.	Timeout must be an integer.
Exception-InvalidArg	-10	12	requires %u argument(s).	Not enough input parameters.

6.9 File System Functions

This section describes functions that perform file-system operations. 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 file system functions are

fs-append-file on page 218

fs-copy-file on page 219

fs-delete-file on page 220

fs-list-files on page 221

fs-make-dir on page 222

fs-read-delim on page 223

fs-read-fixed on page 224

fs-rename-file on page 225

fs-append-file

Syntax

```
(fs-append-file source_file dest_file)
```

Description

fs-append-file appends the contents of the source file to the destination file. If the destination file does not exist, it is created.

Parameters

Name	Type	Description
source_file	string	The source file name
dest_file	string	A valid file name

Return Values

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

Throws

None.

Location

stc_monkfilesys.dll

fs-copy-file

Syntax

```
(fs-copy-file source_file dest_file)
```

Description

fs-copy-file copies the source file to the destination file. If the destination file does not exist, it is created.

Parameters

Name	Type	Description
source_file	string	The source file name
dest_file	string	A valid file name

Return Values

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

Throws

None.

Location

stc_monkfilesys.dll

fs-delete-file

Syntax

```
(fs-delete-file filename)
```

Description

fs-delete-file deletes the specified file.

Parameters

Name	Type	Description
filename	string	The name of the file to delete.

Return Values

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

Throws

None.

Location

stc_monkfilesys.dll

fs-list-files

Syntax

```
(fs-list-files directory regexp)
```

Description

fs-list-files lists all files in the specified directory. If a second parameter is entered, only files matching the specified regular expression are listed. Directories are excluded from the list. On Windows NT systems, files with the hidden, system or archive attributes will be listed.

Parameters

Name	Type	Description
directory	string	The directory containing files to list.
regexp	string	A regular expression (optional)

Return Values

List
Returns a list of files.

Throws

None.

Location

stc_monkfilesys.dll

fs-make-dir

Syntax

```
(fs-make-dir directory)
```

or

```
(fs-make-dir directory option)
```

Description

In the standard, single argument form, **fs-make-dir** creates the named directory, returning **#t** on success. If the directory already exists, or it is not possible to create the directory for some other reason, **#f** is returned.

In the alternate form, an optional Boolean value may be given as the second argument. If this is set to **#f**, then the behavior described above is observed. A value of **#t** indicates that all components of the directory given will be created. If any or all of these components exist, including the final one, then no error is generated and **#t** is returned.

Parameters

Name	Type	Description
directory	string	A valid directory name
option	Boolean	Optional Boolean argument

Return Values

Boolean
Returns **#t** (true) or **#f** (false), as described above.

Throws

None.

Location

stc_monkfilesys.dll

fs-read-delim

Syntax

```
(fs-read-delim port delimiter final_delim)
```

Description

fs-read-delim provides a fast method for reading delimiter records from an already opened file. The input port and the delimiter string are passed as arguments, and the next Event in the file is returned, minus the delimiter. If the final Event in the file is not terminated with the delimiter string, *it is not returned.*

To change this behavior, an additional Boolean value may be supplied. A value of **#t** will provide the same behavior as described above, while a value of **#f** indicates that there is no delimiter on the final record.

Parameters

Name	Type	Description
port	integer	A valid port number
delimiter	string	A record-delimiter string
final_delim	Boolean	Optional Boolean argument

Return Values

String
Returns the string indicating the delimiter records that have been read.

Throws

Exception if the requested delimiter records are not read.

Location

stc_monkfilesys.dll

fs-read-fixed

Syntax

```
(fs-read-fixed port bytes)
```

Description

fs-read-fixed attempts to read a specified number of bytes from an input port. If the final record in the file is less than the requested number of bytes, it is ignored.

Parameters

Name	Type	Description
port	integer	A valid port number
bytes	integer	A non-zero positive integer

Return Values

Returns one of the following values:

String
If the function successfully read the required number of bytes, returns a string of the specified length.

Boolean
Returns **#f** (false) if the required number of bytes cannot be read.

Throws

None.

Location

stc_monkfilesys.dll

fs-rename-file

Syntax

```
(fs-rename-file old_name new_name)
```

Description

fs-rename-file renames a file.

Parameters

Name	Type	Description
old_name	string	The current file name
new_name	string	A valid file name

Return Values

Boolean

Returns **#t** (true) if the operation succeeds; otherwise, returns **#f** (false).

Note: *When moving or renaming a file, the destination volume must be the same as the source volume.*

Throws

None.

Location

stc_monkfilesys.dll

Chapter 7

FTP Event Type Definition

This chapter describes the FTP Event Type Definition (ETD) for the Batch e*Way Intelligent Adapter, as well as the ETD's features.

7.1 FTP ETD: Introduction

The Batch e*Way includes a Java programming language-enabled ETD that allows you to perform FTP operations in a Java environment.

Note: *For this release, the FTP ETD is available on all e*Gate-supported platforms except the OS/390 system.*

The combination of a specific ETD, working with a specific e*Way Connection, with a set of configurable parameters for that e*Way Connection, defines the characteristics of the external interface. Through the FTP ETD and one or more e*Way Connections, you can create the Collaboration Rules to make the Batch e*Way behave in a specific way.

The FTP ETD enables the e*Gate system to exchange data with other network hosts, for the purpose of receiving and delivering Events stored in files. Because of network-mounted file systems or FTP servers on different platforms, the FTP ETD can send to, and receive from, machines running numerous operating systems. This ETD supports standard FTP commands according to RFC-959, for example:

APPE	NOOP	RNTO
CWD	PASS	SITE
DELE	QUIT	STOR
LIST	RETR	TYPE
MKD	RNFR	USER

Caution: *The FTP ETD payload uses a byte array. You must also use a byte array for the payload copy, specifically for binary transfers, but is a good idea in all cases. Failure to do so can cause loss of data.*

7.1.1 Components

The Java-enabled FTP ETD is comprised of the components listed in this section.

New features available only in Java Collaborations

- **ETD**: The FTP ETD can be used in a Collaboration Rule to operate with e*Way Connections.
- **e*Way Connection**: The Java-enabled Batch e*Way Connection provides access to the information necessary for connection to a specified external connection.
- **Configuration and GUI**: An e*Way Connection Properties dialog box is available to centrally configure properties; the e*Way Editor uses configuration files to define configuration parameters.

A complete list of installed files appears in **Table 1 on page 18**.

The FTP ETD must be configured and administered, using the e*Gate Enterprise Manager GUI.

7.1.2 Client Components

Any client components relevant to the Java-enabled FTP ETD have their own requirements. See the subject system's documentation for details.

7.2 FTP ETD: Overview

Essentially, the FTP ETD is a mirror image of the e*Way Connection and allows you to configure specific e*Way Connection parameters in the Java Collaboration controlling the FTP process. Once you have done this configuration, you do not have to define the same parameters in each relative e*Way Connection component that uses this Collaboration.

Note: **Chapter 8** *lists the e*Way Connection configuration parameters for the FTP ETD and explains each one.*

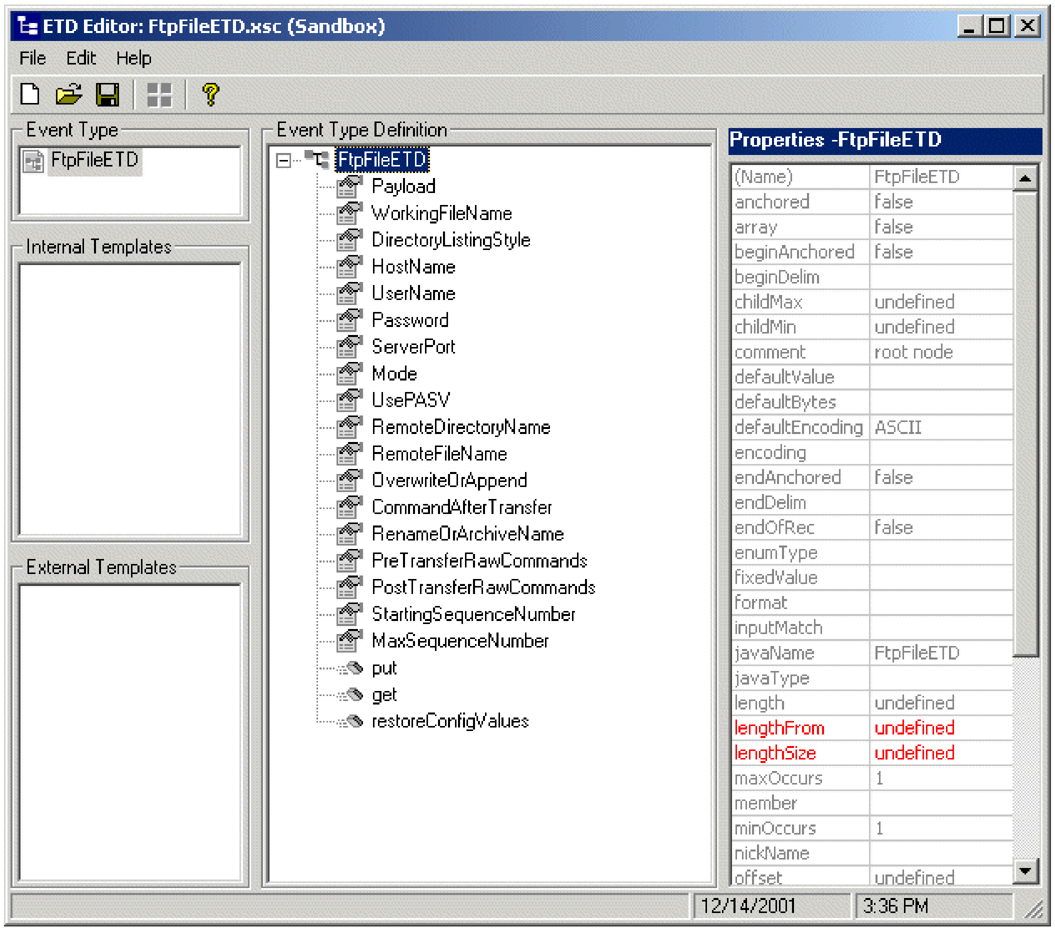
7.2.1 ETD Structure

The file name and installed directory location of the FTP ETD is:

\eGate\client\etd\batchclient\FtpFileETD.xsc

Figure 11 shows the FTP ETD as it appears in the ETD Editor's Main window.

Figure 11 FTP ETD Structure



FtpFileETD
- FtpFileETD
  - Payload
  - WorkingFileName
  - DirectoryListingStyle
  - HostName
  - UserName
  - Password
  - ServerPort
  - Mode
  - UsePASV
  - RemoteDirectoryName
  - RemoteFileName
  - OverwriteOrAppend
  - CommandAfterTransfer
  - RenameOrArchiveName
  - PreTransferRawCommands
  - PostTransferRawCommands
  - StartingSequenceNumber
  - MaxSequenceNumber
  - put
  - get
  - restoreConfigValues

(Name)	FtpFileETD
anchored	false
array	false
beginAnchored	false
beginDelim	
childMax	undefined
childMin	undefined
comment	root node
defaultValue	
defaultBytes	
defaultEncoding	ASCII
encoding	
endAnchored	false
endDelim	
endOfRec	false
enumType	
fixedValue	
format	
inputMatch	
javaName	FtpFileETD
javaType	
length	undefined
lengthFrom	undefined
lengthSize	undefined
maxOccurs	1
member	
minOccurs	1
nickName	
offset	undefined

Note that each field element in the ETD structure corresponds to one of the e*Way Connection's configuration parameters. See **Chapter 8** for an explanation of each of these parameters.

7.2.2 ETD Java Methods

In addition to the field elements shown in **Figure 11 on page 228**, the FTP ETD contains the following Java methods:

- **get()**: Retrieves the payload from the FTP server, that is, it retrieves the first matching file based on **Remote Directory Name** and **Remote File Name** to the payload and performs **Command After Transfer**. It also returns a Boolean **True** if the data is retrieved successfully or **False** if no data is available.
- **put()**: Places the payload on the FTP server, that is, it performs an **append** or **put** from the payload to the remote FTP server and performs **Command After Transfer**. It also returns a Boolean **True** if the data is sent successfully or **False** if the operation fails.
- r**estoreConfigValues()**: Restores all the values from the e*Way Connection to the appropriate values in the FTP ETD.

See **Chapter 9** for more information on each of these methods.

Note: **Chapter 8** *and* **Chapter 9** *explain how to configure and use the FTP ETD.*

Chapter 8

e*Way Connection Configuration

This chapter describes how to configure e*Way Connections for the FTP ETD.

8.1 Configuring e*Way Connections

Set up e*Way Connections using the e*Gate Enterprise Manager graphical user interface (GUI).

To create and configure e*Way Connections

1. In the Enterprise Manager's **Navigation** pane**,** select the **Component** tab.
2. Select the **e*Way Connections** folder.
3. On the palette, click on the icon to create a new **e*Way Connection**.

   The **New e*Way Connection Component** dialog box appears.
4. Enter a name for the **e*Way Connection**, then click **OK**.

   An icon for your new e*Way Connection appears in the Navigation pane.
5. Double-click on the new **e*Way Connection** icon.

   The **e*Way Connection Properties** dialog box appears.
6. From the **e*Way Connection Type** drop-down box, select (for the examples) **Batch**.
7. Enter **100** or **1000** for the **Event Type "get" interval** in the dialog box provided.
8. From the **e*Way Connection Configuration File**, click **New** to open the e*Way Editor GUI.

Note: *To use an existing file, click* ***Find****.*

9. Use the e*Way Editor to create a new configuration file for this e*Way Connection. Do this operation by selecting the appropriate configuration parameters available in the GUI.
10. When you are finished, close the e*Way Editor and save the new configuration file.

The rest of this chapter explains the FTP ETD e*Way Connection configuration parameters as follows:

- **"Connector" on page 231**
- **"FTP File" on page 232**

8.2 Configuration Parameters

This section explains the configuration parameters for the FTP ETD e*Way Connection.

8.2.1 Connector

The parameters in the Connector section allow the Collaboration engine to identify the e*Way Connection.

Type

Description

Specifies the type of connection.

Required Values

FTPFile. The value defaults to **FTPFile**.

Class

Description

Specifies the class name of the FTP file connector object.

Required Values

A valid package name. The default is **com.stc.eways.batch.FTPConnector**.

Property.Tag

Description

Identifies the data source. This parameter is required by the current **EBobConnectorFactory**.

Required Values

A valid data source package name.

8.2.2 FTP File

This section lists the following set of parameters:

Directory Listing Style

Description

Select the system that reflects the remote host. This parameter is used to determine the format in which the **LIST** command returns file listing information.

Required Values

From the list provided, select the name of the desired system.

Host Name

Description

The name of the external system that the e*Way connects to.

Required Values

Enter the name of the external host system, for example, **ftphost**.

User Name

Description

When a log-in to the external system is required, enter the log-in user name to be used.

Required Values

Enter the desired user name.

Password

Description

If a password is required in order to log in to the external system, enter the password that corresponds to the given user name.

Required Values

Enter the required password.

Mode

Description

This parameter describes the mode to transfer data to and from the FTP server.

Required Values

Enter one of the following modes:

- **ASCII**
- **BINARY**

Use PASV

Description

It causes the e*Way to enter the passive or active mode.

Required Values

Select either **Yes** or **No**. The default is **No**.

Server Port

Description

The port number to use on the FTP server when connecting to it.

Required Values

Enter the desired port number.

Remote Directory Name

Description

The directory (absolute path location) on the external system where files are retrieved or sent.

Required Values

Enter the desired directory name and path location.

Remote File Name

Description

For inbound (subscriber), it is the remote file name regular expression. For outbound (publisher), it is the remote file name.

For inbound, files in the remote directory that match the regular expression are retrieved to payload, through **get()**, for processing.

For outbound, this is the name of the file as it appears on the remote system for **put()**. Special characters for date and time and sequence numbering expansions may be used, which are expanded by the e*Way before the file is transmitted.

Required Values

Enter the appropriate remote file name, as specified previously. For example, for MVS GDG, this entry can be the version of the data set.

Additional Examples:

- **Remote Directory Name = 'STC.SAMPLE.GDGSET'**
- **Remote File Name = (0)** to indicate the current version

Overwrite Or Append

Description

Select the appropriate parameter, as follows:

- If **Append** is selected and the remote file already exists, then the payload is appended to the existing file.
- If **Overwrite** is selected, then the e*Way overwrites the existing file on the remote system.
- If a file with the same name does not exist, both **Append** and **Overwrite** create a new file on the external host.

This parameter is for outbound only.

Required Values

Select either **Append** or **Overwrite**, as directed previously.

Command After Transfer

Description

After a file has been successfully retrieved from or sent to the external system, the following actions can be performed on the remote copy: delete, rename, archive. Also, no action can be taken at all.

The rename and archive functions may not be available in all cases. In the case of FTP, they rely on the RNFR command being available on the remote FTP daemon.

When retrieving multiple files, use the **Rename** parameter with care. You set this value yourself, so, to use maximum caution, use name sequencing. There is no default.

Required Values

- **Delete**: Delete the file from the remote host.
- **Rename**: Rename the file.
- **Archive**: Move the file to another directory.
- **None**: Do nothing (leaves the file on the remote host intact).

Rename or Archive Name

Description

Depending on the value in the parameter **Command After Transfer**, this command either specifies the name of the file that the remote file is renamed to or the directory it is archived to (see **"Command After Transfer" on page 234**).

Required Values

Enter either the file or directory name, as explained previously.

Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used.

Note: *If you are entering a path name, use the forward slash (/) instead of the back slash (\) because the e*Way interprets the back slash as a special character and not a path separator. For example, use **c:/temp/dir** for that path location, not **c:\temp\dir**.*

Pre Transfer Raw Commands

Description

These are FTP raw commands needed *before* the file transfer command, for example, some SITE commands.

Note: *These commands are sent to the FTP server directly, so the commands must be FTP raw commands.*

Required Values

Enter the required FTP raw commands. Use semicolons (;) to separate the command set, for example: PWD;CWD;SITE (and so on).

Post Transfer Raw Commands

Description

These are FTP raw commands needed *after* the file transfer command, for example, some SITE commands.

Note: *These commands are sent to the FTP server directly, so the commands must be FTP raw commands.*

Required Values

Enter the required FTP raw commands. Use semicolons (;) to separate the command set, for example: PWD;CWD;SITE (and so on).

Starting Sequence Number

Description

Use this parameter when you have set up the remote file name to contain a sequence number. It tells the e*Way which value to start with in the absence of a sequence number from a previous run.

Also, when the **Max Sequence Number** is reached, the sequence number rolls over to the **Starting Sequence Number**.

This parameter is used for the name pattern %#.

Required Values

The value of the **Starting Sequence Number** *must* be less than the **Max Sequence Number**. The default value is 1.

Max Sequence Number

Description

Use this parameter when you have set up the remote file name to contain a sequence number. It tells the e*Way that when this value (the **Max Sequence Number**) is reached, to reset the sequence number to the **Starting Sequence Number**.

This parameter is used for the name pattern %#.

Required Values

The value of the **Max Sequence Number** *must* be greater than the **Starting Sequence Number**.

Chapter 9

Java Methods

This chapter explains the Java methods used by the FTP Event Type Definition (ETD).

9.1 FTP ETD Methods: Overview

Java methods have been added to make it easier to set information in the **FtpFileETD.xsc** ETD and get information from it. These methods are contained in the following class:

- **"FtpFileETD" on page 237**

9.2 FtpFileETD

The **FtpFileETD** class is the implementation of the FTP ETD. It is the core part of the FTP ETD feature.

The **FtpFileETD** class is defined as:

```
public class FtpFileETD
```

The **FtpFileETD** class extends **com.stc.jcsre.SimpleETDImpl.**

The **FtpFileETD** class methods include:

- **get** on page 239
- **getCommandAfterTransfer** on page 239
- **getDirectoryListingStyle** on page 240
- **getHostName** on page 240
- **getMaxSequenceNumber** on page 241
- **getMode** on page 241
- **getOverwriteOrAppend** on page 242
- **getPassword** on page 242
- **getPayload** on page 243

- **getPostTransferRawCommands** on page 243
- **getPreTransferRawCommands** on page 244
- **getRemoteDirectoryName** on page 244
- **getRemoteFileName** on page 245
- **getRenameOrArchiveName** on page 245
- **getServerPort** on page 246
- **getStartingSequenceNumber** on page 246
- **getUserName** on page 247
- **getWorkingFileName** on page 247
- **Initialize** on page 248
- **isTrace** on page 248
- **isUsePASV** on page 249
- **put** on page 249
- **reset** on page 250
- **restoreConfigValues** on page 250
- **setCommandAfterTransfer** on page 251
- **setDirectoryListingStyle** on page 252
- **setHostName** on page 252
- **setMaxSequenceNumber** on page 253
- **setMode** on page 253
- **setOverwriteOrAppend** on page 254
- **setPassword** on page 254
- **setPayload** on page 255
- **setPostTransferRawCommands** on page 255
- **setPreTransferRawCommands** on page 256
- **setRemoteDirectoryName** on page 256
- **setRemoteFileName** on page 257
- **setRenameOrArchiveName** on page 258
- **setServerPort** on page 258
- **setStartingSequenceNumber** on page 259
- **setTrace** on page 259
- **setUsePASV** on page 260
- **setUserName** on page 260
- **setWorkingFileName** on page 261
- **terminate** on page 261

get

Description

get retrieves a file from the FTP server and stores it in the payload (a byte array). It gets the first matching entry under **remoteDirectoryName** and **remoteFileName**. It is exposed in the ETD structure. This method encapsulates all necessary FTP operations.

Syntax

```
public boolean get()
```

Parameters

None.

Returns

Boolean

- **true** if it gets the file to the payload successfully
- **false** if no file is available to get

Throws

java.lang.Exception if some error occurs.

getCommandAfterTransfer

Description

getCommandAfterTransfer retrieves the action/command performed on a file after it has been successfully retrieved from or sent to the external system as follows:

- **Delete**: The file has been deleted from the remote host.
- **Rename**: The file has been renamed.
- **Archive**: The file has been removed to another directory.
- **None**: The file has been left on the remote host intact; no action was taken.

The rename and archive functions may not be available in all cases. In the case of FTP, they rely on the RNFR command being available on the remote FTP daemon.

The default value is **None**.

Syntax

```
public java.lang.String getCommandAfterTransfer()
```

Parameters

None.

Returns

java.lang.String

The command performed after transfer: **Delete, Rename, Archive**, or **None**.

Throws

None.

getDirectoryListingStyle

Description

getDirectoryListingStyle tells you the listing format the FTP server displays upon issuing a LIST command. **FtpHeuristics** defines all styles supported.

The default value is **UNIX**.

Syntax

```
public java.lang.String getDirectoryListingStyle()
```

Parameters

None.

Returns

java.lang.String

The directory listing style.

Throws

None.

getHostName

Description

getHostName retrieves the name of the FTP server that the e*Way connects to. It can be either an IP address or a host name.

The default value is **localhost**.

Syntax

```
public java.lang.String getHostName()
```

Parameters

None.

Returns

java.lang.String

The IP address or host name.

Throws

None.

getMaxSequenceNumber

Description

getMaxSequenceNumber is used for the name pattern %#.

The value of this parameter is used when you have set up the remote file name to contain a sequence number. It tells the e*Way that, when this value (**Max Sequence Number**) is reached, it must reset the sequence number to the value of **Starting Sequence Number**.

The value of **Max Sequence Number** *must* be greater than **Starting Sequence Number**.

Syntax

```
public long getMaxSequenceNumber()
```

Parameters

None.

Returns

Long

The value of **Max Sequence Number**.

Throws

None.

getMode

Description

getMode retrieves the name of the mode used to transfer data to and from the FTP server, **Binary** or **ASCII**.

The default value is **ASCII**.

Syntax

```
public java.lang.String getMode()
```

Parameters

None.

Returns

java.lang.String

The name of the mode, binary or ASCII.

Throws

None.

getOverwriteOrAppend

Description

getOverwriteOrAppend returns a value, **Overwrite** or **Append**, informing you of file status. If the remote file exists already, it is overwritten or appended. If a file with the same name does not exist, both **Append** and **Overwrite** create a new file on the external host, and the same return values still apply.

Note: *This method operates with outbound files only.*

The default value is **Overwrite**.

Syntax

```
public java.lang.String getOverwriteOrAppend()
```

Parameters

None.

Returns

java.lang.String

Overwrite or Append.

Throws

None.

getPassword

Description

getPassword retrieves the encrypted password that corresponds to the current logged-in user name (no default value).

Syntax

```
public java.lang.String getPassword()
```

Parameters

None.

Returns

java.lang.String

The desired password.

Throws

None.

getPayload

Description

getPayload retrieves the blob (byte array) used to store the raw content of the file; this byte array is called the payload.

Syntax

```
public byte[] getPayload()
```

Parameters

None.

Returns

Byte array

The payload (blob).

Throws

None.

getPostTransferRawCommands

Description

getPostTransferRawCommands allows you to inquire about which FTP raw Commands are used after the file transfer command. Some SITE commands use the semi-colon (;) to separate the command set, for example:

SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE PRI=5;SITE SEC=5

Note: *These commands are sent to the FTP server directly, so the commands must be FTP raw commands.*

Syntax

```
public java.lang.String getPostTransferRawCommands()
```

Parameters

None.

Returns

java.lang.String

The post-transfer raw commands.

Throws

None.

getPreTransferRawCommands

Description

getPreTransferRawCommands allows you to get the FTP raw Commands used before the file transfer command. Some SITE commands use the semi-colon (;) to separate the command set, for example:

SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE PRI=5;SITE SEC=5

Note: *These commands are sent to the FTP server directly, so the commands must be FTP raw commands.*

Syntax

```
public java.lang.String getPreTransferRawCommands()
```

Parameters

None.

Returns

java.lang.String

The pre-transfer raw commands.

Throws

None.

getRemoteDirectoryName

Description

getRemoteDirectoryName retrieves the name of the directory on the external system where files are sent or where they are taken from. For FTP transfers, this path is relative to the home directory of the user in the parameter **User Name**.

Note: *For the publisher (outbound), the directory is created if it does not exist.*

Syntax

```
public java.lang.String getRemoteDirectoryName()
```

Parameters

None.

Returns

java.lang.String

The remote directory name.

Throws

None.

getRemoteFileName

Description

getRemoteFileName allows you to retrieve the remote file name. For the subscriber (inbound), it is the remote file name regular expression, but for the publisher (outbound), it is the remote file name. The following conditions apply:

- For inbound, files in the remote directory, which match the regular expression, are retrieved to the payload (through **get()**) for processing.
- For outbound, the remote file name is the name of the file as it appears on the remote system, for **put()**. Special characters for date, time, and sequence numbering expansions can be used. These characters are expanded by the e*Way before the file is transmitted.

For MVS GDG, this name can be the version of the data set, for example:

- **Remote Directory Name = 'STC.SAMPLE.GDGSET'**
- **Remote File Name = (0)** to indicate the current version

Syntax

```
public java.lang.String getRemoteFileName()
```

Parameters

None.

Returns

java.lang.String

The remote file name.

Throws

None.

getRenameOrArchiveName

Description

getRenameOrArchiveName specifies either the name of the file the remote file is renamed to or the directory it is archived to, depending on the value specified in the configuration parameter **Command After Transfer**.

Note: *Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used, which may cause long file names because the characters are concatenated.*

Syntax

```
public java.lang.String getRenameOrArchiveName()
```

Parameters

None.

Returns

java.lang.String

The renamed file name or archive directory name.

Throws

None.

getServerPort

Description

getServerPort retrieves the number of the port to use on the FTP server when connecting to it.

The default value is 21.

Syntax

```
public int getServerPort()
```

Parameters

None.

Returns

Integer

The server port number.

Throws

None.

getStartingSequenceNumber

Description

getStartingSequenceNumber is used for the name pattern %#.

The value of this parameter is used when you have set up the remote file name to contain a sequence number. It tells the e*Way which value to start with in the absence of a sequence number from the previous run.

Also, when the value **Max Sequence Number** value is reached, the sequence number rolls over to the value of **Starting Sequence Number**.

The value of **Starting Sequence Number** *must* be less than **Max Sequence Number**.

The default value is 1.

Syntax

```
public long getStartingSequenceNumber()
```

Parameters

None.

Returns

Long

The value of **Starting Sequence Number**.

Throws

None.

getUserName

Description

getUserName retrieves the current user name that was used when logging in to the FTP server.

The default value is **anonymous**.

Syntax

```
public java.lang.String getUserName()
```

Parameters

None.

Returns

java.lang.String

The desired user name.

Throws

None.

getWorkingFileName

Description

getWorkingFileName allows you, through a regular expression or file name expansion, to get a real working file name after using the **get()** or **put()** methods.

Syntax

```
public java.lang.String getWorkingFileName()
```

Parameters

None.

Returns

java.lang.String

The desired working file name.

Throws

None.

Initialize

Description

Initialize is called by the Collaboration Service to initialize the object; it overrides **initialize** in class **com.stc.jcsre.SimpleETDImpl.**

Syntax

```
public void initialize(com.stc.common.collabService.JCollabController
    cntrCollab, java.lang.String key,int mode)
```

Parameters

Name	Type	Description
cntrCollab	String	The Java Collaboration controller object.
key	Integer	The instance name.
mode	String	The input/output mode.

Returns

Void.

Throws

- **com.stc.common.collabService.CollabConnException** for a connection exception
- **com.stc.common.collabService.CollabDataException** for a data exception

isTrace

Description

isTrace allows you to retrieve the trace flag status.

Syntax

```
public boolean isTrace()
```

Parameters

None.

Returns

Boolean

The trace flag status (**true** or **false**).

Throws

None.

isUsePASV

Description

isUsePASV allows to inquire whether the current data transfer status is passive, that is, using the passive mode.

Normally, when you connect to an FTP server, it establishes the data connection to your client. However, some FTP servers allow passive transfers, meaning that your client establishes the data connection.

By default, the passive mode is switched off, but it is a good idea that you use it for transfers to and from FTP sites that support it.

The passive mode can be required in the following situations:

- For users on networks behind some types of router-based fire walls
- Users on networks behind a gateway requiring passive transfers
- If transfers are erratic
- If you keep getting failed data channel errors

Syntax

```
public boolean isUsePASV()
```

Parameters

None.

Returns

Boolean

- **false** if the status of the current data transfer is not the passive (PASV) mode
- **true** if the status is passive

Throws

None.

put

Description

put retrieves a file from the payload (a byte array) then stores it to the FTP server. This method is exposed in the ETD structure and encapsulates all necessary FTP operations.

Syntax

```
public boolean put()
```

Parameters

None.

Returns

Boolean

- **true** if the file is transferred to FTP server successfully
- **false** if it is not

Throws

java.lang.Exception if some error occurs.

reset

Description

reset resets the data content of an ETD; it overrides **reset** in class **com.stc.jcsre.SimpleETDImpl**.

Syntax

```
public boolean reset()
```

Parameters

None.

Returns

Boolean

- **false** if the ETD does not have a meaningful implementation of **reset()**; you must correct the ETD’s implementation
- **true** if the reset clears the data content of the ETD

Throws

- **com.stc.common.collabService.CollabConnException** for a connection exception
- **com.stc.common.collabService.CollabDataException** for a data exception

restoreConfigValues

Description

restoreConfigValues allows you to restore all configuration properties from the e*Way Connection. It is exposed in the ETD structure.

Syntax

```
public boolean restoreConfigValues()
```

Parameters

None.

Returns

Boolean

- **true** if the properties are successfully restored
- **false** if they are not

Throws

java.lang.Exception if some error occurs.

setCommandAfterTransfer

Description

setCommandAfterTransfer allows you to set the action/command to be performed on a file after it has been successfully retrieved from or sent to the external system as follows:

- **Delete**: The file is deleted from the remote host.
- **Rename**: The file is renamed.
- **Archive**: The file is removed to another directory.
- **None**: The file is left on the remote host intact; no action is taken.

The rename and archive functions may not be available in all cases. In the case of FTP, they rely on the RNFR command being available on the remote FTP daemon.

The default value is **None**.

Syntax

```
public void setCommandAfterTransfer(java.lang.String
    newCommandAfterTransfer)
```

Parameters

Name	Type	Description
newCommandAfterTransfer	String	The command to be performed after transfer: Delete, Rename, Archive, or None.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setDirectoryListingStyle

Description

setDirectoryListingStyle allows you to set the listing format the FTP server is to display upon issuing a LIST command. **FtpHeuristics** defines all styles supported.

The default value is **UNIX**.

Syntax

```
public void setDirectoryListingStyle(java.lang.String
    newDirectoryListingStyle)
```

Parameters

Name	Type	Description
newDirectoryListingStyle	String	The directory listing style.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setHostName

Description

setHostName allows you to set the name of the FTP server the e*Way connects to. It can be either an IP address or a host name. The default value is **localhost**.

Syntax

```
public void setHostName(java.lang.String newHostName)
```

Parameters

Name	Type	Description
newHostName	String	The appropriate IP address or host name.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setMaxSequenceNumber

Description

setMaxSequenceNumber is used for the name pattern %#.

The value of this parameter is used when you have set up the remote file name to contain a sequence number. It tells the e*Way that when this value (**Max Sequence Number**) is reached, it must reset the sequence number to the value of **Starting Sequence Number**.

The value of **Max Sequence Number** *must* be greater than **Starting Sequence Number**.

Syntax

```
public void setMaxSequenceNumber(long newMaxSequenceNumber)
```

Parameters

Name	Type	Description
newMaxSequenceNumber	Long	The Max Sequence Number.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setMode

Description

setMode allows you to set the mode used to transfer data to and from the FTP server, **Binary** or **ASCII**.

The default value is **ASCII**.

Syntax

```
public void setMode(java.lang.String newMode)
```

Parameters

Name	Type	Description
newMode	String	The desired mode, Binary or ASCII.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setOverwriteOrAppend

Description

setOverwriteOrAppend allows you to overwrite or append a file if the remote file exists already. If a file with the same name does not exist, both **Append** and **Overwrite** create a new file on the external host.

Note: *This method operates with outbound files only.*

The default value is **Overwrite**.

Syntax

```
public void setOverwriteOrAppend(java.lang.String
    newOverwriteOrAppend)
```

Parameters

Name	Type	Description
newOverwriteOrAppend	String	Overwrite or Append.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setPassword

Description

setPassword allows you to set the encrypted password that corresponds to the current logged-in user name (no default value).

Syntax

```
public void setPassword(java.lang.String newPassword)
```

Parameters

Name	Type	Description
newPassword	String	The desired password.

Returns

Void.

Throws

None.

setPayload

Description

setPayload allows you to set the payload, that is the blob or byte array, used to store the raw content of the transferred file.

Syntax

```
public void setPayload(byte[] newPayload)
```

Parameters

Name	Type	Description
newPayload	Byte array	The desired payload (blob).

Returns

Void.

Throws

None.

setPostTransferRawCommands

Description

setPostTransferRawCommands allows you to set the FTP raw Commands that are used after the file transfer command. Some SITE commands use the semi-colon (;) to separate the command set, for example:

SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE PRI=5;SITE SEC=5

Note: *These commands are sent to the FTP server directly, so the commands must be FTP raw commands.*

Syntax

```
public void setPostTransferRawCommands(java.lang.String
    newPostTransferRawCommands)
```

Parameters

Name	Type	Description
newPostTransferRawCommands	String	The desired post-transfer raw commands.

Returns

Void.

Throws

None.

setPreTransferRawCommands

Description

setPreTransferRawCommands allows you to set the FTP raw Commands that are used before the file transfer command. Some SITE commands use the semi-colon (;) to separate the command set, for example:

SITE RECFM=FB;SITE LRECL=50;SITE BLOCKSIZE=32750;SITE TRACKS;SITE PRI=5;SITE SEC=5

Note: *These commands are sent to the FTP server directly, so the commands must be FTP raw commands.*

Syntax

```
public void setPreTransferRawCommands(java.lang.String
    newPreTransferRawCommands)
```

Parameters

Name	Type	Description
newPreTransferRawCommands	String	The desired pre-transfer raw commands.

Returns

Void.

Throws

None.

setRemoteDirectoryName

Description

setRemoteDirectoryName allows you to set the directory on the external system where files are to be sent or retrieved from. For FTP transfers, this path is relative to the home directory of the user in the configuration parameter **User Name**.

Note: *For the publisher (outbound), the directory is created if it does not exist.*

Syntax

```
public void setRemoteDirectoryName(java.lang.String
    newRemoteDirectoryName)
```

Parameters

Name	Type	Description
newRemoteDirectoryName	String	The name of the remote directory.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setRemoteFileName

Description

setRemoteFileName allows you to set the remote file name. For the subscriber (inbound), it is the remote file name regular expression, but for the publisher (outbound), it is the remote file name. The following conditions apply:

- For inbound, files in the remote directory, which match the regular expression, are retrieved to the payload (through **get()**) for processing.
- For outbound, the remote file name is the name of the file as it appears on the remote system, for **put()**. Special characters for date, time, and sequence numbering expansions can be used. These characters are expanded by the e*Way before the file is transmitted.

For MVS GDG, this name can be the version of the data set, for example:

- **Remote Directory Name = 'STC.SAMPLE.GDGSET'**
- **Remote File Name = (0)** to indicate the current version

Syntax

```
public void setRemoteFileName(java.lang.String newRemoteFileName)
```

Parameters

Name	Type	Description
newRemoteFileName	String	The desired remote file name.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setRenameOrArchiveName

Description

setRenameOrArchiveName allows you either to specify the name of the file that the remote file is renamed to or the directory it is archived to, depending on the value in the configuration parameter **Command After Transfer**.

Note: *Special characters are allowed. The expansion of any special characters is carried out each time this parameter is used.*

Syntax

```
public void setRenameOrArchiveName(java.lang.String
    newRenameOrArchiveName)
```

Parameters

Name	Type	Description
newRenameOrArchiveName		The specified file name or directory name.

Returns

Void.

Throws

None.

setServerPort

Description

setServerPort allows you to specify the port number to use on the FTP server when connecting to it.

The default value is 21.

Syntax

```
public void setServerPort(int newServerPort)
```

Parameters

Name	Type	Description
newServerPort	Integer	The server port number.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setStartingSequenceNumber

Description

setStartingSequenceNumber is used for the name pattern %#.

The value of this parameter is used when you have set up the remote file name to contain a sequence number. It tells the e*Way which value to start with in the absence of a sequence number from the previous run.

Also, when the **Max Sequence Number** value is reached, the sequence number rolls over to the value of **Starting Sequence Number**.

The value of the **Starting Sequence Number** *must* be less than **Max Sequence Number**.

The default value is 1.

Syntax

```
public void setStartingSequenceNumber(long newStartingSequenceNumber)
```

Parameters

Name	Type	Description
newStartingSequenceNumber	Long	The Starting Sequence Number.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setTrace

Description

setTrace allows you to set the trace flag.

Syntax

```
public void setTrace(boolean newTrace)
```

Parameters

Name	Type	Description
newTrace	Boolean	The Trace flag (true or false).

Returns

Void.

Throws

None.

setUsePASV

Description

setUsePASV Normally, when you connect to an FTP site, the site establishes the data connection to your PC. However, some FTP sites allow passive transfers. This means that your PC establishes the data connection. By default, passive mode is turned off; we recommend that you use it for transfers to and from FTP sites that support it.

The passive mode may be required in the following instances:

- For users on networks behind some types of router-based fire walls
- Users on networks behind a gateway requiring passive transfers
- If transfers are erratic
- If you keep getting failed data channel errors

Syntax

```
public void setUsePASV(boolean newUsePASV)
```

Parameters

Name	Type	Description
newUsePASV		The Use PASV flag.

Returns

Void.

Throws

None.

setUserName

Description

setUserName allows you to create a user name to use when logging in to the FTP server.

The default value is **anonymous**.

Syntax

```
public void setUserName(java.lang.String newUserName)
```

Parameters

Name	Type	Description
newUserName	String	The desired user name.

Returns

Void.

Throws

java.lang.Exception if some error occurs.

setWorkingFileName

Description

setWorkingFileName allows you to set the working file name. This method is not called by the client. It is ignored and updated by the **get()** and **put()** methods.

Syntax

```
public void setWorkingFileName(java.lang.String newWorkingFileName)
```

Parameters

Name	Type	Description
newWorkingFileName	String	The working file name.

Returns

Void.

Throws

None.

terminate

Description

terminate terminates the ETD. It overrides **terminate** in the class **com.stc.jcsre.SimpleETDImpl**.

Syntax

```
public void terminate()
```

Parameters

None.

Returns

Void.

Throws

com.stc.common.collabService.CollabConnException when there is an external connection problem.

Appendix A

Document Type Definitions

This appendix provides Document Type Definitions (DTDs) for the XML Messages used in Dynamic Configuration. The **payload** element in each DTD contains a new attribute, **location**, which can have two values: **base64InSitu** or **localDir**.

The **base64InSitu** value is the default, which implies that the data is Base64-encoded, and that it is located in the body of the **payload** element.

If the **location** attribute is **localDir**, the Batch e*Way assumes that the payload data is contained in a file in a local directory on the Participating Host. The local directory is specified by a value (a the directory name) stored in the **payload** element. If you do not want to transport large files through the Intelligent Queues, for the sole purpose of sending the files to an external location, using the **localDir** attribute is recommended.

A.1 Send or Receive XML Messages

The DTD below provides details of the XML Message that can be used for Send orders, or Receive orders.

```
<!-- Copyright (C) 2000, SeeBeyond Technology Corporation, All rights
reserved. -->
<!-- batch eway order record format. -->
<!ELEMENT batch_eWay_order (command, (order_record)+, payload?)>
<!ELEMENT command (#PCDATA)>
<!ATTLIST command
    Enumeration (send | receive) "send"
>
<!ELEMENT order_record (external_host_setup?, (subscribe_to_external
| publish_to_external)?, FTP?, SOCKS?)>
<!ELEMENT external_host_setup (host_type?, external_host_name?,
user_name?, encrypted_password?, file_transfer_method?, return_tag?)>
<!ELEMENT host_type (#PCDATA)>
<!ELEMENT external_host_name (#PCDATA)>
<!ELEMENT user_name (#PCDATA)>
<!ELEMENT encrypted_password (#PCDATA)>
<!ELEMENT file_transfer_method (#PCDATA)>
<!ATTLIST file_transfer_method
    Enumeration (ftp | copy) "ftp"
>
<!ELEMENT return_tag (#PCDATA)>
<!ELEMENT subscribe_to_external (remote_directory_name?,
remote_file_regexp?, remote_command_after_transfer?,
remote_rename_or_archive_name?, local_command_after_transfer?,
local_archive_directory?)>
<!ELEMENT remote_directory_name (#PCDATA)>
```

```
<!ELEMENT remote_file_regexp (#PCDATA)>
<!ELEMENT remote_command_after_transfer (#PCDATA)>
<!ATTLIST remote_command_after_transfer
    Enumeration (archive | delete | none | rename) "delete"
>
<!ELEMENT remote_rename_or_archive_name (#PCDATA)>
<!ELEMENT local_command_after_transfer (#PCDATA)>
<!ATTLIST local_command_after_transfer
    Enumeration (archive | delete) "delete"
>
<!ELEMENT local_archive_directory (#PCDATA)>
<!ELEMENT publish_to_external (remote_directory_name?,
remote_file_name?, append_or_overwrite_when_transferring_files?,
remote_command_after_transfer?, remote_rename_or_archive_name?,
local_command_after_transfer?, local_archive_directory?)>
<!ELEMENT remote_file_name (#PCDATA)>
<!ELEMENT append_or_overwrite_when_transferring_files (#PCDATA)>
<!ATTLIST append_or_overwrite_when_transferring_files
    Enumeration (append | overwrite) "append"
>
<!ELEMENT FTP (server_port, mode, Pretransfer_Commands,
Posttransfer_Commands)>
<!ELEMENT server_port (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT Pretransfer_Commands (#PCDATA)>
<!ELEMENT Posttransfer_Commands (#PCDATA)>
<!ELEMENT SOCKS (server_host_name, server_port, method, user_name,
encrypted_password)>
<!ELEMENT server_host_name (#PCDATA)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT payload (#PCDATA)>
<!ATTLIST payload
    Location (base64InSitu | localDir) #IMPLIED
>
```

A.2 Error Messages

The DTD below is used for the Error Reporting XML Message.

```
<!-- Copyright (C) 2000, SeeBeyond Technology Corporation, All rights
reserved. -->
<!-- batch eway error record format. -->
<!ELEMENT batch_eWay_error (command, (return_tag | order_record)?,
error_record, payload?)>
<!ELEMENT command (#PCDATA)>
<!ATTLIST command
    Enumeration (send | receive) "send"
>
<!ELEMENT order_record (external_host_setup?, (subscribe_to_external
| publish_to_external)?, FTP?, SOCKS?)>
<!ELEMENT external_host_setup (host_type?, external_host_name?,
user_name?, encrypted_password?, file_transfer_method?, return_tag?)>
<!ELEMENT host_type (#PCDATA)>
<!ELEMENT external_host_name (#PCDATA)>
<!ELEMENT user_name (#PCDATA)>
<!ELEMENT encrypted_password (#PCDATA)>
<!ELEMENT file_transfer_method (#PCDATA)>
<!ATTLIST file_transfer_method
    Enumeration (ftp | copy) "ftp"
>
```

```
<!ELEMENT return_tag (#PCDATA)>
<!ELEMENT subscribe_to_external (remote_directory_name?,
remote_file_regexp?, remote_command_after_transfer?,
remote_rename_or_archive_name?, local_command_after_transfer?,
local_archive_directory?)>
<!ELEMENT remote_directory_name (#PCDATA)>
<!ELEMENT remote_file_regexp (#PCDATA)>
<!ELEMENT remote_command_after_transfer (#PCDATA)>
<!ATTLIST remote_command_after_transfer
    Enumeration (archive | delete | none | rename) "delete"
>
<!ELEMENT remote_rename_or_archive_name (#PCDATA)>
<!ELEMENT local_command_after_transfer (#PCDATA)>
<!ATTLIST local_command_after_transfer
    Enumeration (archive | delete) "delete"
>
<!ELEMENT local_archive_directory (#PCDATA)>
<!ELEMENT publish_to_external (remote_directory_name?,
remote_file_name?, append_or_overwrite_when_transferring_files?,
remote_command_after_transfer?, remote_rename_or_archive_name?,
local_command_after_transfer?, local_archive_directory?)>
<!ELEMENT remote_file_name (#PCDATA)>
<!ELEMENT append_or_overwrite_when_transferring_files (#PCDATA)>
<!ATTLIST append_or_overwrite_when_transferring_files
    Enumeration (append | overwrite) "append"
>
<!ELEMENT FTP (server_port, mode, Pretransfer_Commands,
Posttransfer_Commands)>
<!ELEMENT server_port (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT Pretransfer_Commands (#PCDATA)>
<!ELEMENT Posttransfer_Commands (#PCDATA)>
<!ELEMENT SOCKS (server_host_name, server_port, method, user_name,
encrypted_password)>
<!ELEMENT server_host_name (#PCDATA)>
<!ELEMENT method (#PCDATA)>
<!ELEMENT payload (#PCDATA)>
<!ATTLIST payload
    Location (base64InSitu | localDir) #IMPLIED
>
<!ELEMENT error_record (error_code, error_text, last_action)>
<!ELEMENT error_code (#PCDATA)>
<!ELEMENT error_text (#PCDATA)>
<!ELEMENT last_action (#PCDATA)>
```

A.3 Data Message

The DTD file below provides a data structure, includes a data payload, and is used for transporting data to Batch e*Way. See **“Enable Message Configuration” on page 71**.

```
<!-- Copyright (C) 2000, SeeBeyond Technology Corporation, All rights
reserved. -->
<!-- batch eway data record format. -->
<!ELEMENT batch_eWay_data (command,
                           (return_tag|order_record)?,
                           payload) >
<!ELEMENT command (#PCDATA) >
<!ATTLIST command Enumeration (send|receive) "send" >
<!ELEMENT order_record (external_host_setup?,
                        (subscribe_to_external|publish_to_external)?,
                        FTP?,
                        SOCKS?) >
<!ELEMENT external_host_setup (host_type?,
                               external_host_name?,
                               user_name?,
                               encrypted_password?,
                               file_transfer_method?,
                               return_tag?) >
<!ELEMENT host_type (#PCDATA) >
<!ELEMENT external_host_name (#PCDATA) >
<!ELEMENT user_name (#PCDATA) >
<!ELEMENT encrypted_password (#PCDATA) >
<!ELEMENT file_transfer_method (#PCDATA) >
<!ATTLIST file_transfer_method Enumeration (ftp|copy) "ftp" >
<!ELEMENT return_tag (#PCDATA) >
<!ELEMENT subscribe_to_external (remote_directory_name?,
                                 remote_file_regexp?,
                                 remote_command_after_transfer?,
                                 remote_rename_or_archive_name?,
                                 local_command_after_transfer?,
                                 local_archive_directory?) >
<!ELEMENT remote_directory_name (#PCDATA) >
<!ELEMENT remote_file_regexp (#PCDATA) >
<!ELEMENT remote_command_after_transfer (#PCDATA) >
<!ATTLIST remote_command_after_transfer Enumeration
(archive|delete|none|rename) "delete" >
<!ELEMENT remote_rename_or_archive_name (#PCDATA) >
<!ELEMENT local_command_after_transfer (#PCDATA) >
<!ATTLIST local_command_after_transfer Enumeration (archive|delete)
"delete" >
<!ELEMENT local_archive_directory (#PCDATA) >
<!ELEMENT publish_to_external (remote_directory_name?,
                               remote_file_name?,

append_or_overwrite_when_transferring_files?,
                               remote_command_after_transfer?,
                               remote_rename_or_archive_name?,
                               local_command_after_transfer?,
                               local_archive_directory?) >
<!ELEMENT remote_file_name (#PCDATA) >
<!ELEMENT append_or_overwrite_when_transferring_files (#PCDATA) >
<!ATTLIST append_or_overwrite_when_transferring_files Enumeration
(append|overwrite) "append" >
<!ELEMENT FTP (server_port,
               mode,
               Pretransfer_Commands,
               Posttransfer_Commands) >
<!ELEMENT server_port (#PCDATA) >
<!ELEMENT mode (#PCDATA) >
<!ELEMENT Pretransfer_Commands (#PCDATA) >
<!ELEMENT Posttransfer_Commands (#PCDATA) >
<!ELEMENT SOCKS
(server_host_name,server_port,method,user_name,encrypted_password) >
<!ELEMENT server_host_name (#PCDATA) >
<!ELEMENT method (#PCDATA) >
<!ELEMENT payload (#PCDATA) >
```

Index