SeeBeyond ICAN Suite

e*Way Intelligent Adapter for IMS User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Monk Version

Version 20050504132249.

Contents

Chapter 5

Chapter 1

Introduction

This user's guide explains how to install and configure SeeBeyond™ Technology Corporation's (SeeBeyond™) e*Way Intelligent Adapter for IMS.

1.1 Overview

The IMS e*Way enables e*Gate Integrator to access IMS Transaction Manager mainframe applications through IBM's IMS Connect.

The implementation of the IMS e*Way is in accordance with the IBM documents IMS TCP/IP OTMA Connection User's Guide and Reference, Version 2.1.3, and IMS Connect Guide and Reference, Version 1. These documents describe the OTMA protocol and contain important prerequisite information for the configuration of IMS Connect on the mainframe.

The e*Way includes a utility, stcIMSmfs2ssc, to create input and output Event Type Definitions from IMS Message Format Service (MFS) files.

The e*Way contains (within ims-out.monk) an example of how a non-conversational scenario is managed; that is, a simple send / response scenario. Conversational transactions require more complex programming in SeeBeyond's Monk language.

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, also to have a working knowledge of Windows and UNIX operations and administration, to have an in-depth understanding of IMS transactions and the OTMA protocol, and to be thoroughly familiar with Windows-style graphical user interface (GUI) operations.

1.1.2. Components

The IMS e*Way comprises the following:

- **stcewgenericmonk.exe**, the executable component
- Configuration files that the e*Way Editor uses to define configuration parameters
- Monk function scripts, discussed in **IMS e*Way Functions** on page 47

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

1.2 Supported Operating Systems

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

- Windows 2000, Windows XP, and Windows Server 2003
- HP-UX 11.0 and 11i (PA-RISC)
- IBM AIX 5.1L
- IBM z/OS V1.3 and V1.4
- Sun Solaris 8
- Japanese Windows 2000, Windows XP, and Windows Server 2003
- Japanese HP-UX 11.0 and 11i (PA-RISC)
- Japanese Sun Solaris 8
- Korean Sun Solaris 8

1.3 System Requirements

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

- An e*Gate Participating Host.
- A TCP/IP network connection.
- Additional disk space for e*Way executable, configuration, library, and script files. The disk space is required on both the Participating and the Registry Host. Additional disk space is required to process and queue the data that this e*Way processes; the amount necessary varies based on the type and size of the data being processed, and any external applications performing the processing.

1.3.1. Client Components

The client components of IMS have their own requirements; see the subject system's documentation for details.

1.4 External System Requirements

Note: *This document uses the term RACF when referring to RACF or an equivalent product.*

Hardware Requirements

- Host processor capable of running the versions of operating system, IMS, TCP/IP, and RACF detailed below.

Software Requirements

- IMS Connect version 2.1 or 2.2.

Software Requirements for IMS Connect

- IMS Version 7, 8, and 9.1, with the required maintenance APARs applied for each version. See the *IMS Connect Guide and Reference*, for more information about IMS and IMS Connect coexistence.
- TCP/IP (Version 3.2, or Version 3.4 or later), APAR PQ13154, and APAR PQ38814.
- Resource Access Control Facility (RACF) Version 1.9.2 or later, or equivalent product.

Important: *IMS Connect does not support TCP/IP Version 3.3*

Chapter 2

Installation

This chapter explains the procedures for installing the IMS e*Way.

2.1 Windows Installation

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 IMS e*Way on a Windows system

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

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

11 From the **Select an e*Way Template** list, select **stcewims** and click **OK**.

12 The e*Way Editor launches. Make any necessary changes, then save the configuration file.

13 Return to the e*Way's property sheet. Click **OK** to close the property 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** chapter of the **e*Gate Integrator User's Guide**.*

2.2 UNIX

2.2.1. Pre-installation

You must have root privileges to install this e*Way.

2.2.2. Installation Procedure

To install the IMS e*Way on a UNIX system

1 Log in as root on the workstation containing the CD-ROM drive, and insert the CD-ROM into the drive.

2 If necessary, mount the CD-ROM drive.

3 At the shell prompt, type

cd /cdrom

4 Start the installation script by typing

setup.sh

5 A menu of options appears. Select the **e*Gate Add-on Applications** option. Then, follow any additional on-screen directions.

Caution: *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 path" setting.***

6 After installation is complete, exit the installation utility and launch the Enterprise Manager.

7 In the Navigator/Component pane, 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 Under **Configuration file**, click **New**.

12 From the **Select an e*Way template** list, select **stcewims** and click **OK**.

13 The e*Way Editor launches. Make any necessary changes, then save the configuration file.

14 Return to the e*Way's property sheet. Click **OK** to close the property 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, 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*** *chapter of the* ***e*Gate Integrator User's Guide****.*

2.3 Files/Directories Created by the Installation

The system installs files within the **egate\client** tree on the Participating Host and committed to the **default** schema on the Registry Host. **Table 1 on page 12** shows files installed within the e*Gate directory tree by the installation process. The path shown is relative to the **\client** directory.

Table 1 Files Created by the Installation

e*Gate Directory	File(s)
/bin/	stc_monkims.dll stcIMSmfs2ssc.exe stccococo.exe
/configs/stcewgenericmonk/	stcewims.def
/monk_library/	ims.gui ewims/ims-ack.monk ewims/ims-connect.monk ewims/ims-exchange.monk ewims/ims-funcs.monk ewims/ims-init.monk ewims/ims-nak.monk ewims/ims-notify.monk ewims/ims-out.monk ewims/ims-shutdown.monk ewims/ims-startup.monk ewims/ims-verify.monk
/msg/	ims.properties ims_ja_JP.properties ims_ko_KR.properties

Chapter 3

Configuration

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

3.1 e*Way Configuration Parameters

Set the IMS e*Way configuration parameters, using the e*Way Editor.

To set or change e*Way configuration parameters

1. In the Enterprise Manager's Components tab, select the e*Way you want to configure and display its properties.
2. 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.
3. Under **Configuration File**, click **New** to create a new file, **Find** to select an existing configuration file, or **Edit** to edit the currently selected file.

Note: *When creating a new e*Way, you must also select the* ***stcewims*** *template file from the* ***e*Way Template Selection*** *list.*

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

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

- General Settings
- Communication Setup
- Monk Configuration
- IMS

3.1.1. General Settings

The General Settings control basic operational parameters.

Journal File Name

Description

Specifies the name of the journal file.

Required Values

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

Additional Information

An Event is journaled for the following conditions:

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

Max Resends Per Message

Description

Specifies the number of times the e*Way attempts to resend messages to the external system after receiving an error.

Required Values

An integer between 1 and 1,000,000. The default is 5.

Max Failed Messages

Description

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

Required Values

An integer between 1 and 1,000,000. 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 is queued to the e*Way's configured queue. See **"Exchange Data with External Function" on page 28** for more information. See **"Schedule-driven Data Exchange Functions" on page 22** for more information about how the e*Way uses this function.

Required Values

YES or **NO**. The default value, **YES**, specifies that error messages can 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.

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

Start Exchange Data Schedule

Description

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

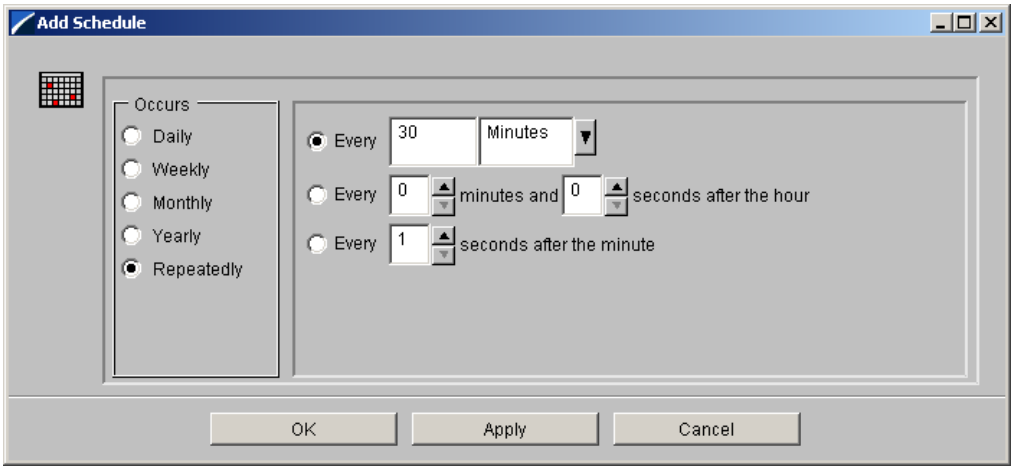
Required Values

One of the following values:

- One or more specific dates/times
- A single repeating interval, such as yearly, weekly, monthly, daily or repeatedly (every n seconds, minutes, hours, days, or weeks). Since months vary in length, "Months" is not provided as a unit under "Repeatedly."

  Since months do not all contain equal numbers of days, be sure not to provide boundaries that would cause an invalid date selection (i.e. the 30th of every month would not include February).

Figure 1 Add Schedule Dialog Box



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 terminates execution when the schedule attempts to start.

Additional Information

When the schedule starts, the e*Way determines whether it is waiting to send positive acknowledgement 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 acknowledgment condition is pending, and the connection is active, the e*Way immediately executes the **Exchange Data with External** function. Thereafter, the **Exchange Data with External** function is called according to the **Exchange Data Interval** parameter until the **Stop Exchange Data Schedule** time is reached.

See **"Exchange Data with External Function" on page 28**, **"Exchange Data Interval" on page 16**, and **"Stop Exchange Data Schedule" on page 16** for more information.

Stop Exchange Data Schedule

Description

Establishes the schedule to stop data exchange.

Required Values

One of the following values:

- One or more specific dates/times
- A single repeating interval, such as yearly, weekly, monthly, daily or repeatedly (every n seconds, minutes, hours, days, or weeks). Since months vary in length, "Months" is not provided as a unit under "Repeatedly."(See **Figure 1 on page 15**)

  Since months do not all contain equal numbers of days, be sure not to provide boundaries that would cause an invalid date selection (i.e. the 30th of every month would not include February).

Exchange Data Interval

Description

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

Required Values

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

Additional Information

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

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

See **"Down Timeout" on page 17** and **"Stop Exchange Data Schedule" on page 16** for more information about the data-exchange schedule.

Down Timeout

Description

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

Required Values

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

Up Timeout

Description

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

Required Values

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

Resend Timeout

Description

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

Required Values

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

Zero Wait Between Successful Exchanges

Description

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

Required Values

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

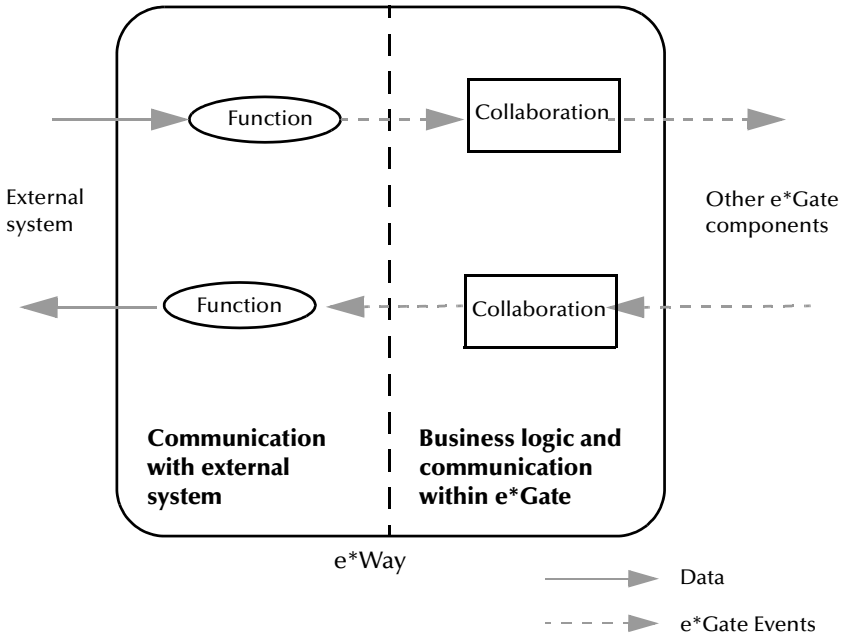
See **"Exchange Data with External Function" on page 28** for more information.

3.1.3. Monk Configuration

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

Conceptually, an e*Way is divided into two halves. One half of the e*Way (shown on the left in **Figure 2 on page 18**) handles communications with the external system. The other half manages the related internal business logic, that is, the Collaborations that process data and subscribe or publish to other e*Gate components.

Figure 2 e*Way internal architecture



The communications half of the e*Way uses Monk functions to start and stop scheduled operations, to exchange data with the external system, to package data as e*Gate Events and to send those Events to Collaborations, and to manage the connection between the e*Way and the external system.

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

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

Operational Details

The Monk functions in the communications half of the e*Way fall into the groups shown in **Table 2 on page 19**. A series of figures, following the table, illustrates the interaction and operation of these functions.

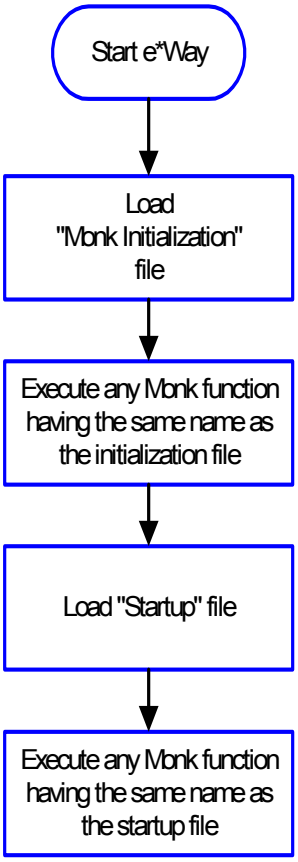
Table 2 Monk Functions, Communications Half

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

Initialization Functions

Figure 3 below illustrates how the e*Way executes its initialization functions.

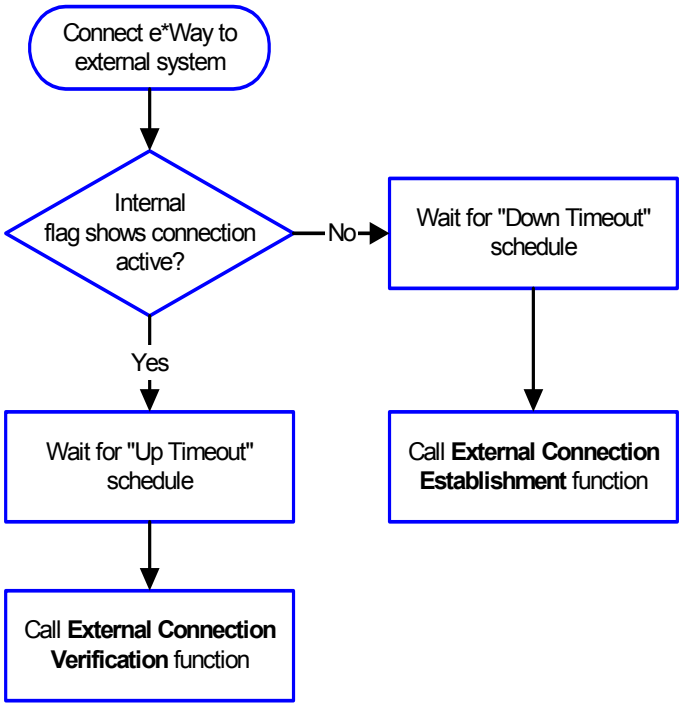
Figure 3 Initialization Functions



Connection Functions

Figure 4 below illustrates how the e*Way executes the connection establishment and verification functions.

Figure 4 Connection Establishment and Verification Functions

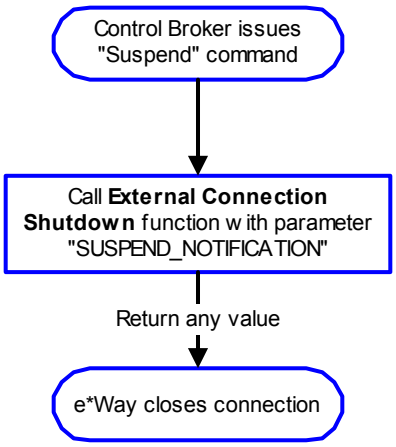


Note: *The e*Way selects the connection function based on an internal "up/down" flag rather than a poll to the external system. See* **Figure 6 on page 23** *and* **Figure 8 on page 25** *for examples of how different functions use this flag.*

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

Figure 5 on page 22 illustrates how the e*Way executes its **connection shutdown** function.

Figure 5 Connection Shutdown Function



Schedule-driven Data Exchange Functions

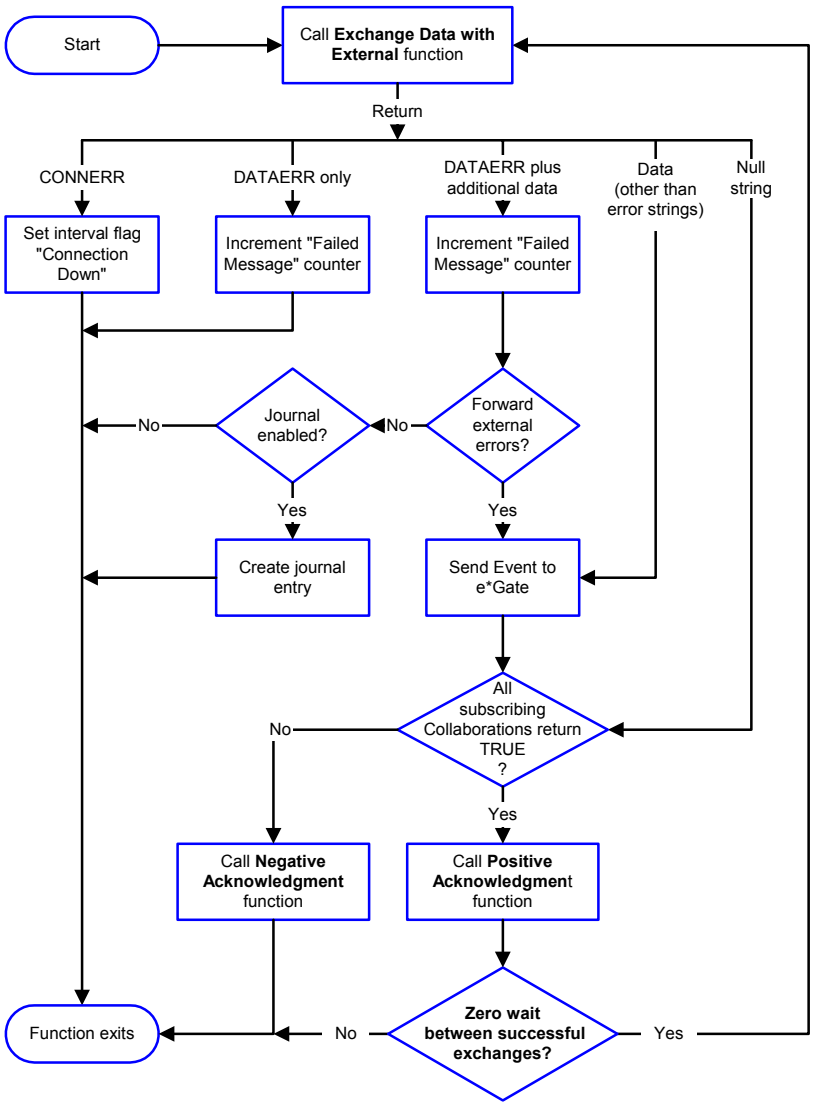
Figure 6 on page 23 illustrates how the e*Way performs schedule-driven data exchange, using the **Exchange Data with External Function**. The **Positive Acknowledgement Function** and **Negative Acknowledgement Function** are also called during this process.

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.

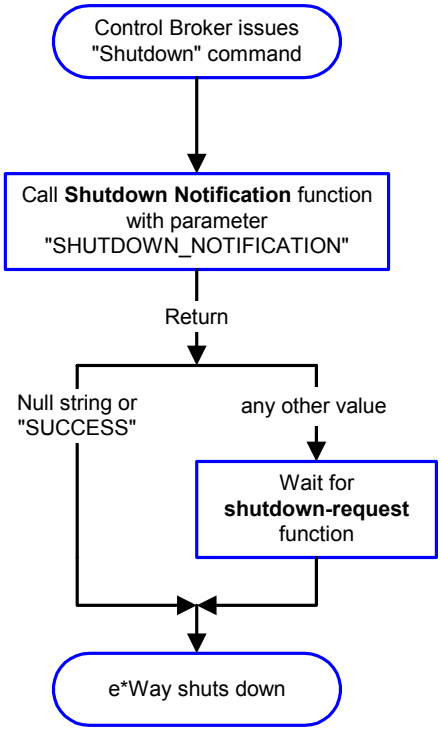
Figure 6 Schedule-driven Data Exchange Functions



Shutdown Functions

Figure 7 on page 24 illustrates how the e*Way implements the **shutdown request** function.

Figure 7 Shutdown Functions

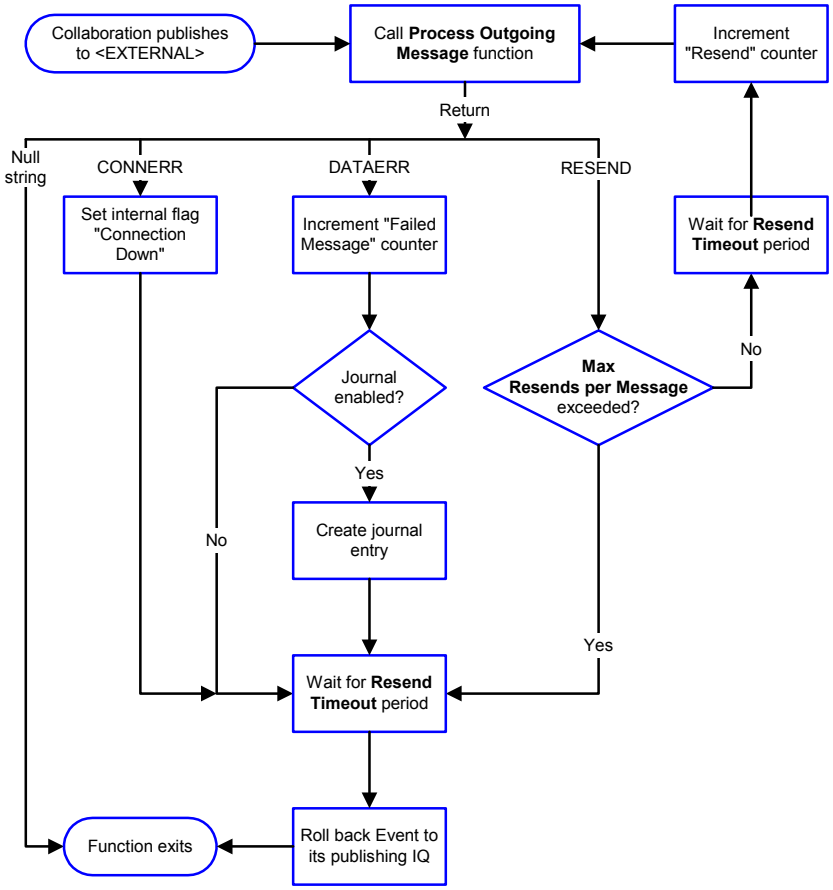


Event-driven Data Exchange Functions

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 8 on page 25** illustrates event-driven data exchange using the **Process Outgoing Message** function.

Figure 8 Event-driven Data-exchange Functions



How to Specify Function Names or File Names

Parameters requiring 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** is searched after the default load paths.

Required Values

A path name, 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 type the directory names rather than selecting them with the **Find File** button. Directory names must be separated with semicolons, and you can mix absolute paths with relative e*Gate paths, for example:

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

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

Auxiliary Library Directories

Description

Specifies a path to auxiliary library directories. Any**.monk** files found within those directories are automatically loaded into the e*Way's Monk environment.

Required Values

A path name, or a series of paths separated by semicolons. The default is monk_library/ewims. This parameter is optional and may be left blank.

Additional Information

To specify multiple directories, manually type the directory names rather than selecting them with the **Find File** button. Directory names must be separated with semicolons, and you can mix absolute paths with relative e*Gate paths, for example:

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

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

Monk Environment Initialization File

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

Required Values

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

Additional Information

Any environment-initialization functions called by this file accept no input and must return a string. The e*Way loads this file and tries to invoke a function of the same base name as the file name (for example, for a file named **my-init.monk**, the e*Way attempts 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 3 on page 20**).

Startup Function

Description

Specifies a Monk function that the e*Way loads and invokes upon startup or whenever the e*Way's configuration is reloaded. The system uses this function to initialize the external system before data exchange starts.

Required Values

The name of a Monk function or the name of a file (optionally including path information) containing a Monk function. This parameter is optional and may be left blank. The default is ims-startup.

Additional Information

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

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

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

The e*Way loads this file and tries to invoke a function of the same base name as the file name (see **Figure 3 on page 20**). For example, for a file named **my-startup.monk**, the e*Way would attempt to execute the function **my-startup**.

Process Outgoing Message Function

Description

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

Required Values

The name of a Monk function or the name of a file (optionally including path information) containing a Monk function. The default is ims-out. ***This parameter cannot be left 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 values:

- 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 Event data itself.
- If a string other than the following is returned, the e*Way creates an entry in the log file indicating that an attempt has been made to access an unsupported function.

See **Figure 8 on page 25** for more details.

Note: *If you wish to use* **event-send-to-egate** *to enqueue failed Events in a separate Intelligent Queue (IQ), the e*Way must have an inbound Collaboration (with appropriate IQs) configured to process those Events. See* **event-send-to-egate** ***on page 53*** *for more information.*

Exchange Data with External Function

Description

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

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. This parameter is optional and may be left blank. The default is ims-exchange.

Additional Information

The function accepts no input and must return a string (see **Figure 6 on page 23** for more details):

- Null string — Indicates that the data exchange was completed successfully. No information is sent into the e*Gate system.
- "CONNERR" — Indicates that a problem with the connection to the external system has occurred.

- "DATAERR" — Indicates that a problem with the data itself has occurred. The e*Way handles the string "DATAERR" and "DATAERR" plus additional data differently; see **Figure 6 on page 23** for more details.
- Any other string — The contents of the string are packaged as an inbound Event. The e*Way must have at least one Collaboration configured suitably to process the inbound Event, as well as any required IQs.

This function is initially triggered by the **Start Exchange Data** schedule or manually by the Monk function **start-schedule**. After the function has returned true and the data received by this function has been 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 immediately calls the **Exchange Data with External** function again; otherwise, the e*Way does not call the function until the next scheduled **Start Exchange Data Schedule** time or the schedule is manually invoked using the Monk function **start-schedule** (see **start-schedule** on page 48 for more information).

External Connection Establishment Function

Description

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

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. The default is ims-connect. ***This field cannot be left blank.***

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 **Down Timeout** parameter, and is *only* called according to this schedule.

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

External Connection Verification Function

Description

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

Required Values

The name of a Monk function. This function is optional; if no **External Connection Verification** function is specified, the e*Way executes the **External Connection Establishment** function in its place. The default is ims-verify

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 previous section) is called when the e*Way has determined that its connection to the external system is down.

External Connection Shutdown Function

Description

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

Required Values

The name of a Monk function. The default is ims-shutdown. This parameter is optional.

Additional Information

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

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

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

Positive Acknowledgment Function

Description

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

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. The default is ims-ack. 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 as follows:

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

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

Negative Acknowledgment Function

Description

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

Required Values

The name of a Monk function, or the name of a file (optionally including path information) containing a Monk function. This parameter is required if the **Exchange Data with External** function is defined. The default is ims-nak.

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 as follows:

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

This function is only called during the processing of inbound Events. After the **Exchange Data with External** function returns a string that is transformed into an inbound Event, the Event is handed off to one or more Collaborations for further processing.

If the Event's processing is not completed successfully by *all* the Collaborations to which it was sent, the e*Way executes the **Negative Acknowledgment** function (otherwise, the e*Way executes the **Positive Acknowledgment** function).

Shutdown Command Notification Function

Description

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

Required Values

The name of a Monk function. The default is ims-notify.

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 does not proceed until the Monk function **shutdown-request** is executed (see **shutdown-request** on page 54).

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. IMS

The parameters in this section set up the IMS-specific information required by the e*Way.

Host

Description

The host name or dotted IP address of the system on which IMS Connect is running.

Port

Description

The TCP/IP port number on which IMS Connect is running.

Required Values

1 through 65535.

RACF ID

Description

The Resource Access Control Facility (RACF) user ID to log into the Host system.

Required Values

A valid RACF user ID (up to eight characters). The RACF ID must be all upper case.

RACF Password

Description

A valid password (up to eight characters) for the above user ID. The RACF Password must be all upper case.

RACF Group

Description

The RACF group name of the user specified in RACF ID.

User exit

Description

An eight character string specifying the identifier of the IMS Connect user message exit to be invoked after the complete message has been received. User message exits can be used to handle EBCDIC / ASCII conversion, among other things. IBM supplies several user message exits with IMS Connect. Custom exits may be written (in assembly language) if desired.

For example, specify "*SAMPLE*" to invoke the IBM-supplied exit HWSSMPL0. Some IBM-supplied exits are:

- *IRMREQ* for HWSIMSO0
- *SAMPLE* for HWSSMPL0
- *HWSJAV* for HWSJAVA0

Client ID

Description

Unique eight-character ID to identify this instance of the client to IMS Connect.

MOD name

Description

Specifies whether a MFS Message Output Descriptor (MOD) name should be returned as part of the output.

Required Values

Y or N. When Y is specified, a Request Mod Message (RMM) will be returned as the first structure of the output message. This structure will have an ID of *REQMOD* followed by the MFS MOD name. For details, see IMS Connect Guide and Reference, page 59.

Commit mode

Description

The Commit mode to use. This parameter, in combination with Sync Level (see below), determines transaction flow. For details, see IMS Connect Guide and Reference, Chapter 8, "Protocols."

Required Values

Either S (Send then commit) or C (Commit then send).

LTERM

Description

A valid name or blanks.

Sync level

Description

Specifies the synchronization level to use for this transaction. Used in combination with Commit Mode (see above).

Required Values

Either 0 or 1 as follows:

- 0 — NONE (Commit mode 0)
- 1 — CONFIRM (Commit mode 1)

Trancode

Description

The ID of the IMS transaction to be invoked.

Datastore

Description

The IMS database name that is the destination for this transaction.

3.2 Environment Configuration

To support the operation of this e*Way, *no* changes are necessary

- In the Participating Host's operating environment
- In the e*Gate system

Note: *Changes to Monk files can be made using the Collaboration Rules Editor (available from within the 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 are not 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 in the* ***e*Gate Integrator System Administration and Operations Guide****.*

3.3 External Configuration Requirements

IMS Connect needs to be properly configured on the mainframe, with the desired user exits specified in the HWSCFGxx parmlib member. To verify proper configuration, reply VIEWHWS to the outstanding *HWS READY* message on the system console. Status should be ACTIVE, as in the example below.

```
IEE600I REPLY TO 10 IS;VIEWHWS
*15 HWSC0000I *HWS READY*  HWS1
 HWSC0001I   HWS ID=HWS1        Racf=N
 HWSC0001I     Maxsoc=50  Timeout=6000
 HWSC0001I    Datastore=IVP1     Status=ACTIVE
 HWSC0001I     Group=STCGRP01 Member=STCXCF
 HWSC0001I     Target Member=STCIMS
 HWSC0001I    Port=7777     Status=ACTIVE
 HWSC0001I     No active Clients
EADY*  HWS1
```

Chapter 4

Implementation

This chapter explains how to construct input and output Event Type Definitions (ETDs) using the MFS and Cobol Copybook file converters.

4.1 Implementation Notes

The default IMS configuration provides the framework for an outbound e*Way. That is, this e*Way responds to the arrival of an Event, which is assumed to be in a format suitable for the configured IMS transaction as follows:

- A connection is then made to the IMS TOC in accordance with the parameters specified for the e*Way.
- The data is "wrapped" in the OTMA protocol and transmitted to the IMS TOC. The response from the specified IMS transaction becomes the outbound Event for the e*Way.

The function controlling the connection and transmit/receive strategy is **ims-out.monk**. Note that the parameters passed to the receive method of the Extension DLL (see **receive** on page 62) are hard-coded in this Monk script.

This operation is intentional, and SeeBeyond recommends that you do not modify these values when using the default framework for an outbound e*Way. Use other parameter values (as explained on page 62) with great care.

4.2 The IMS MFS Converter Tool

The IMS MFS file converter (**stcIMSmfs2ssc.exe**) parses MFS files and creates ETDs (***.ssc** files) from the IMS control statements they contain. More than one MFS file can be converted at the same time when the converter is run from the ETD Editor.

In addition, the ***.ssc** files contain transaction codes so that a Monk execute script can step through the states of a transaction, using only the initial input Event(s) and the ***.ssc** file structure. This operation also happens with the conversational transactions that comprise several exchanges in both directions.

See **"Sample Input File — invsamp.txt" on page 42** for a sample of an input MFS file.

Note: *You must process the MFS definitions you input to the* ***stcIMSmfs2ssc*** *converter, using the MFS language utility to ensure they are valid MFS definitions. Invalid definitions cause the converter to produce an error message and close.*

4.2.1. Running the File Converter

You may run the **stcIMSmfs2ssc** file converter either from the ETD Editor window, using the Build feature, or from the command line.

Using the ETD Editor's Build Tool

The ETD Editor's Build feature automatically creates an ETD file from specified input files. Use this procedure to create an ETD based upon the MFS file(s) your installation requires.

To create an ETD using the ETD Editor

1. Launch the ETD Editor.
2. On the ETD Editor's Toolbar, click **Build**.

   The **Build an Event Type Definition** dialog box appears.
3. In the **File Name** text box, type the name of the ETD file you wish to build. *Do not specify any file extension.* The ETD Editor supplies an ***.ssc** extension for you.

Note: *This name is given to the first* ****.ssc*** *file created. If you select more than one MFS file in the* ***Converter Wizard*** *dialog box, this name is used for the first file to be converted (regardless of the input file name). However, for additional files, the file names of the input MFS files are used to create the* ****.ssc*** *file names.*

4. Click **Next**.

   A new dialog box appears.
5. Leave the **Input file** selection box blank. This file name must be entered later.
6. Under **Build From**, select **Library Converter**.
7. Under **Select a Library Converter**, select **IMS MFS File Converter**.
8. Click **Finish**.

   The **IMS MFS File Converter Wizard** dialog box opens. This dialog box allows you to select one or more MFS files to be converted.
9. When the **IMS MFS File Converter Wizard** dialog box opens, type the following information:

   - For **IMS Host Name or LOCALHOST**, type the host name of the IMS Host or **LOCALHOST**.
   - For **User Name**, type your user name to log in to your local Window box.
   - For **Password**, type your password to log into your local Window box, and then click **Next**.

- For **Fully qualified MFS dataset name, or full pathname if LOCALHOST**, type the full path of the directory where the MFS files are located in your local Window box.
- For **Dataset member mask, or file mask LOCALHOST**, type ***.*** then click **Next**. A new dialog box appears.
- Select an MFS file to be converted from **MFS files found in specified dataset**, then click **Add**. If more than one file is to be converted, select and click Add as necessary. When selection is complete, click **Finish**.

Using the Command Line

To run the **IMS MFS** file converter as a batch utility, from the command line, type:

```
stcIMSmfs2ssc -i <input MFS file> -o <output structure file>
[-e <error file>] [-v]
```

The output structure file name should be a full path name and should end in ***.ssc**. The following table explains these command options:

Command	Type	Description
-i	String	A mandatory parameter.
input MFS file	Path	The path and name of the file to be used as the input (mandatory).
-o	String	A mandatory parameter.
output structure file	Path	The path and name of the ETD to be created (mandatory).
-e	String	Errors are written to the screen unless you use the -e flag (optional), when they are written to <error file>.
error file	String	The name of the error file (optional).
-v	String	Causes a warning to be printed for all statements and parameters that are ignored (optional).

When run from the e*Gate GUI, the specified ETD is given an ***.ssc** extension automatically. In principle, the name given on the command line is free-form but the use of the ***.ssc** extension is recommended.

Note: *This feature does not commit the ***.ssc** file to the e*Gate registry. You must either commit the file using the **stcregutil** command or run the converter from the ETD Editor, which allows you to commit the ***.ssc** file.*

Control Statement Syntax

The **stcIMSmfs2ssc** converter only supports MFS statements in which

- The label (if present) begins in column 1.
- The operation begins in column 10.
- The operand begins in column 16.

Summary

The **stcIMSmfs2ssc** converter performs one of the following actions on each MFS statement or parameter:

- Handles it
  The statement or parameter is reflected in the ***.ssc** file.
- Ignores it
- Produces an error message and exits

The error message and exit occurs for statements and parameters that are not supported by the **stcIMSmfs2ssc** converter but that should be reflected in the ***.ssc** file. Under these circumstances, no portion of the MFS file will be converted by **stcIMSmfs2ssc**.

The **stcIMSmfs2ssc** converter processes statements as follows:

- **MSG Statements** — Handled with some exceptions, as shown in the table below.

Parameter	Handled?	Description
TYPE	Handles	The TYPE (INPUT or OUTPUT[a]) is added to the node name. Generated sub nodes are marked as optional for **OUTPUT** messages to allow messages with omitted fields or segments to be accepted.
SOR	Ignores	N/A
OPT	OPT=1 and OPT=2 are handled; OPT=3 errors and exit.	TYPE=INPUT — Messages are fixed format. All fields or segments are converted and marked as non-optional (that is, all fields in the message must be filled in before it is sent). TYPE=OUTPUT — Messages are fixed format. Fields at the end of a segment may be omitted, or entire segments may be empty.
NXT	Handled	NXT=XXX — The value _NXT_XXX is added to the node name.
PAGE	Ignores	N/A
FILL	Ignores	N/A

a. In MFS files, INPUT and OUTPUT are relative to IMS.

- **FMT Statements**—Everything between the FMT and FMTEND statements is ignored.
- **PDB Statements**—Everything between the PDB and PDB END statements is ignored.
- **TABLE Statements**—Everything between the TABLE and TABLEEND statements is ignored.

- **Compilation Statements**—Sets are ignored or become errors, with some exceptions as follows:
  - **Handles**—ALPHA, END (**stcIMSmfs2ssc** stops parsing the MFS file).
  - **Ignores**—EJECT, PRINT, SPACE, STACK, TITLE and UNSTACK.
  - **Errors and Exits**—COPY, EQU and RESCAN.
- **Definition Statements** — The following sections explain details of how the **stcIMSmfs2ssc** converter handles these statements.

LPAGE Statement

The **stcIMSmfs2ssc** converter ignores it.

PASSWORD Statement

The **stcIMSmfs2ssc** converter ignores it.

SEG Statement

The **stcIMSmfs2ssc** converter handles it. A SEG subnode is created that contains all the MFLD statements before the next SEG or MSGEND statement.

Parameter	Handled?	Description
EXIT	Errors and exits	N/A
GRAPHIC	Errors and exits	N/A

DO Statement

The **stcIMSmfs2ssc** converter handles it. A subnode is created that contains all the MFLDs before the DOEND statement.

Parameter	Handled?	Description
count	Handles	MSG TYPE=INPUT — The minimum repetition and maximum repetition of the node is set to the count value. MSG TYPE=OUTPUT — the minimum repetition is zero (to allow for omitted fields), and the maximum repetition is the count value.
SUF	Ignores	N/A

MFLD Statement

The **stcIMSmfs2ssc** converter handles it. A subnode may be created (depending on the **dfldname/literal** parameter) as follows:

- Minimum repetition is 0 for MSG TYPE=OUTPUT, otherwise 1.
- Maximum repetition is 1.

Parameter	Handled?	Description
label	Handles	The label is added to the node name. It is also used for the tag field in the ***.ssc** file after it has been padded to 8 characters with x40.
dfldname	Handles	The dfldname is appended to the node name.
'literal'	Handles	This is assumed to be the transaction code for MSG TYPE=INPUT. TRAN_CODE is appended to the node name, and the literal value is put in the default data field.
(dfldname, 'literal')	MSG TYPE = INPUT — Handles	dfldname is appended to the node name and the literal value is put in the default-data field.
	MSG TYPE = OUTPUT — Ignores	No node is created.
"Nothing" and MSG TYPE = OUTPUT	Handles	A node called MFLD_PADDING is created. **Note:** Nothing means none of dfldname, 'literal', (dfldname,'literal'), (dfldname,system-literal), or (,SCA). An example would be MFLD, LTH=10 that results in a 10-byte padding field.
(dfldname, system-literal)	Ignores	No node is created.
(,SCA)	Ignores	No node is created.
LTH=nn	Handles	The value nn is put in the length field of the ***.ssc** file.
LTH=(pp,nn)	Handles	A padding node is generated. The ***.ssc** file length field is (pp - 1). Then a data node is generated; the value nn is put in the length field of the ***.ssc** file.
JUST	Ignores	N/A
ATTR	Handles	Generates a padding node, MFLD_ATTR_nnn, if required and a data node with the length field adjusted to account for the padding field.
FILL	Ignores	N/A
EXIT	Errors and exits	N/A

To use the *.ssc files

1. In the IMS parameters section, set the **Mod** parameter to **Y**.

   This action ensures that the **Mod** name is returned in the output. See **"MOD name" on page 33**.

2. Use the sample file **ims-out.monk**.

This action puts the **Mod** name as a tag at the start of the returned message; and puts the tag found at SEG_OUT_001 in the ***.ssc** file at the start of every data segment in the returned message.

3 The **stcIMSmfs2ssc** converter generates ***.ssc** files with the segment tag **StCmAgIcMaGiCsTc.**

In the unlikely event of this being possible data, you may edit the generated ***.ssc** to use a different tag for all nodes starting with SEG_OUT_.

Sample Input File — invsamp.txt

```
PRINT ON,NOGEN
DI2DF    FMT
         DEV   TYPE=(3270,2),                                    X
               FEAT=IGNORE,                                      X
               SYSMSG=ERRMSG,                                    X
               DSCA=X'00A0'
         DIV   TYPE=INOUT                                         00100001
         DPAGE CURSOR=((3,12)),                                  X00110001
               FILL=PT                                           00120001
         DFLD  'DISPLAY INVOICE SUMMARY',                        X00130001
               POS=(1,2),                                        X00140001
               ATTR=(HI,PROT)                                    00150001
         DFLD  'DATE:',                                          X00160001
               POS=(1,65),                                       X00170001
               ATTR=(HI,PROT)                                    00180001
CURDATE  DFLD  POS=(1,72),                                       X00190001
               LTH=8,                                            X00200001
               ATTR=(HI,PROT)                                    00210001
         DFLD  'INVOICE:',                                       X00220001
               POS=(3,2),                                        X00230001
               ATTR=(HI,PROT)                                    00240001
INVNO    DFLD  POS=(3,12),                                       X00250001
               LTH=6,                                            X00260001
               ATTR=(NUM)                                        00270001
         DFLD  'DATE:',                                          X00280001
               POS=(3,21),                                       X00290001
               ATTR=(HI,PROT)                                    00300001
INVDATE  DFLD  POS=(3,28),                                       X00310001
               LTH=8,                                            X00320001
               ATTR=(PROT)                                       00330001
         DFLD  'SUBTOTAL:',                                      X00340001
               POS=(3,41),                                       X00350001
               ATTR=(HI,PROT)                                    00360001
SUBTOTAL DFLD  POS=(3,55),                                       X00370001
               LTH=9,                                            X00380001
               ATTR=(PROT)                                       00390001
         DFLD  'DISCOUNT:',                                      X00400001
               POS=(4,41),                                       X00410001
               ATTR=(HI,PROT)                                    00420001
DISCOUNT DFLD  POS=(4,55),                                       X00430001
               LTH=9,                                            X00440001
               ATTR=(PROT)                                       00450001
         DFLD  'SALES TAX:',                                     X00460001
               POS=(5,41),                                       X00470001
               ATTR=(HI,PROT)                                    00480001
SALESTAX DFLD  POS=(5,55),                                       X00490001
               LTH=9,                                            X00500001
               ATTR=(PROT)                                       00510001
         DFLD  'FREIGHT:',                                       X00520001
               POS=(6,41),                                       X00530001
               ATTR=(HI,PROT)                                    00540001
FREIGHT  DFLD  POS=(6,55),                                       X00550001
               LTH=9,                                            X00560001
               ATTR=(PROT)                                       00570001
         DFLD  '---------',                                      X00580001
               POS=(7,55),                                       X00590001
               ATTR=(HI,PROT)                                    00600001
         DFLD  'BILLING:',                                       X00610001
               POS=(8,41),                                       X00620001
               ATTR=(HI,PROT)                                    00630001
BILLING  DFLD  POS=(8,55),                                       X00640001
               LTH=9,                                            X00650001
               ATTR=(PROT)                                       00660001
         DFLD  'PMTS/ADJS',                                      X00670001
               POS=(9,41),                                       X00680001
               ATTR=(HI,PROT)                                    00690001
PMTSADJS DFLD  POS=(9,55),                                       X00700001
               LTH=9                                             X00710001
               ATTR=(PROT)                                       00720001
         DFLD  '---------',                                      X00730001
               POS=(10,55),                                      X00740001
               ATTR=(HI,PROT)                                    00750001
         DFLD  'DUE:',                                           X00760001
               POS=(11,41),                                      X00770001
```

```
               ATTR=(HI,PROT)                                          00780001
DUE       DFLD  POS=(11,55),                                          X00790001
               LTH=9,                                                  X00800001
               ATTR=(PROT)                                             00810001
ERRMSG    DFLD  POS=(23,2),                                           X00820001
               LTH=79,                                                 X00830001
               ATTR=(PROT)                                             00840001
          FMTEND                                                       00850001
*************************************************************************00860000
DI2I      MSG   TYPE=INPUT,                                           X00870001
               SOR=(DI2DF,IGNORE),                                     X00880001
               NXT=DI2O                                                00890001
          SEG                                                          00900001
          MFLD  'DI2     ',                                           X00910001
               LTH=8                                                   00920001
          MFLD  INVNO,                                                X00930001
               LTH=6,                                                  X00940001
               JUST=R,                                                 X00950001
               FILL=C'0'                                               00960001
          MSGEND                                                       00970001
*************************************************************************00980000
DI2O      MSG   TYPE=OUTPUT,                                          X00990001
               SOR=(DI2DF,IGNORE),                                     X01000001
               NXT=DI2I                                                01010001
          SEG                                                          01020001
          MFLD  (CURDATE,DATE2)                                        01030001
          MFLD  INVNO,                                                X01040001
               LTH=6                                                   01050001
          MFLD  INVDATE,                                              X01060001
               LTH=8                                                   01070001
          MFLD  SUBTOTAL,                                             X01080001
               LTH=9                                                   01090001
          MFLD  DISCOUNT,                                             X01100001
               LTH=9                                                   01110001
          MFLD  SALESTAX,                                             X01120001
               LTH=9                                                   01130001
          MFLD  FREIGHT,                                              X01140001
               LTH=9                                                   01150001
          MFLD  BILLING,                                              X01160001
               LTH=9                                                   01170001
          MFLD  PMTSADJS,                                             X01180001
               LTH=9                                                   01190001
          MFLD  DUE,                                                  X01200001
               LTH=9                                                   01210001
          MFLD  ERRMSG,                                               X01220001
               LTH=9                                                   01230001
          MSGEND                                                       01240001
*************************************************************************01250000
          END                                                          01260000
```

4.3 Cobol Copybook Converter

The Cobol Copybook Converter is a build tool that takes a Cobol copybook specification file as input and creates an ETD file, that is, an ***.ssc** file.

4.3.1. Tuning the Cobol Copybook Converter

This feature has the following basic properties:

- You may run this command via the ETD Editor or from the command line.
- The system presents the copybook specification to the Cobol Copybook Converter in a flat file. The converter feature uses the 01 segment of the Cobol copybook as the root node of the ETD.

Note that IMS FDs typically contain three reserved items at the beginning of the FD, namely one COMP field for the segment length, a second COMP field, which is fixed with a value of zero and a third element of CHAR(8) in which the transaction name is placed.

These three items must be deleted either from the Copybook file prior to processing or from the resulting ETD postprocessing as they are handled as part of the implementation.

4.3.2. Using the Cobol Copybook Converter

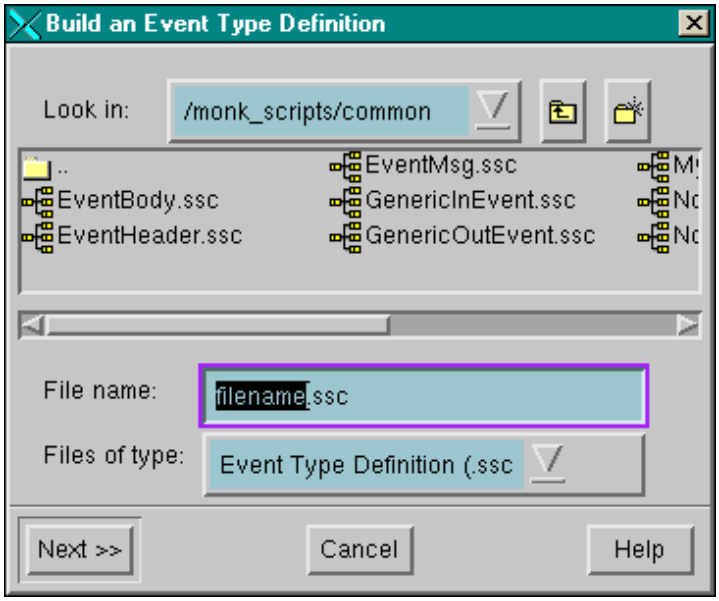
You may access the Cobol Copybook Converter feature via the ETD Editor (see page 37). Use the procedure below to create an ETD based on the data your installation requires.

To create an ETD file using the ETD Editor's Build Tool

1. Launch the ETD Editor.
2. On the ETD Editor Toolbar, click **Build**.

   The first **Build an Event Type Definition** dialog box window appears (see Figure 9 below).

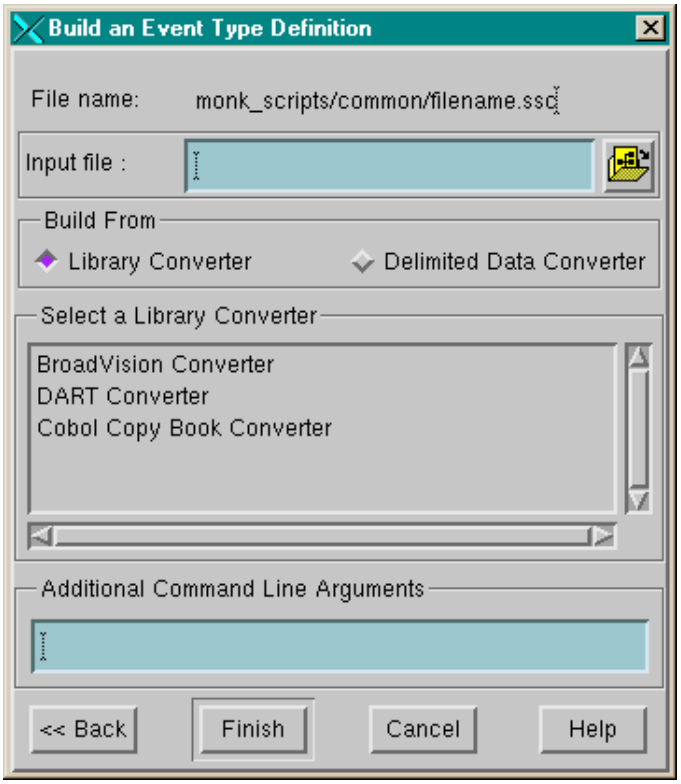
Figure 9 Build an Event Type Definition Dialog Box (Initial)



3. In the **File Name** text box, type the name of the ETD file you wish to build. *Do not specify any file extension*. The ETD Editor supplies the ***.ssc** extension for you.
4. Under **Files of Type,** leave **Event Type Definition** (.ssc) selected.
5. Click **Next**.

   The second **Build an Event Type Definition** dialog box window appears.

Figure 10 Build an Event Type Definition Dialog Box (Second)



6 For **Input File** type the name and path of the file to be converted.

7 Under **Build From**, select **Library Converter**.

8 Under **Select a Library Converter**, select **Cobol Copy Book Converter**.

9 Click **Finish**.

The dialog box closes, and the new ETD appears in the Workspace pane of the ETD Editor window. The conversion process is automatic.

10 Click **Save** to save the new ETD.

Using the command line

To run the Cobol Copybook Converter from the command line, as a batch utility, type

```
stccococo [-I] [-d] -i <copybook_file> <etd>.ssc
```

The following table explains these command options:

Command	Type	Description
-I	String	An optional parameter that indicates ICL data origin.
-d	String	An optional parameter that indicates DBCS input data mode.
-nls	String	An optional parameter that allows the conversion of Cobol files written in Japanese; currently the only string supported is SJIS (Japanese character set). Use of any other (or no) parameter defaults to English/ASCII.
-i	String	A mandatory parameter.
copybook_file	Path	The relative path of the file to be used as the input (mandatory).
etd	String	The name of the ETD file you wish to create (mandatory).

Note: *If you type* ***stccococo -h*** *at the command line, seeking help, you get the following message:*

USAGE: stccococo [-I] [-d] [-c control [-s section]] [-nls <Character Set>] -i copybook_file <etd>.ssc

The ***-c*** *and* ***-s*** *switches are for future use only and do not operate in the current system at this time.*

Chapter 5

IMS e*Way Functions

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

- **Basic Functions** on page 47
- **IMS Connect Functions** on page 55

Note: *The functions explained 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.*

5.1 Basic Functions

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

start-schedule on page 48

stop-schedule on page 49

send-external-up *on page 50*

send-external-down *on page 51*

get-logical-name on page 52

event-send-to-egate *on page 53*

shutdown-request on page 54

start-schedule

Syntax

```
(start-schedule)
```

Description

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

Parameters

None.

Return Values

None.

Throws

None.

stop-schedule

Syntax

```
(stop-schedule)
```

Description

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

Parameters

None.

Return Values

None.

Throws

None.

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.

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.

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.

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.

shutdown-request

Syntax

```
(shutdown-request)
```

Description

shutdown request requests the e*Way to perform the shutdown procedure when there is no outstanding incoming/outgoing event. When the e*Way is ready to act on the shutdown request, it invokes the **Shutdown Command Notification** function (see**"Shutdown Command Notification Function" on page 31**). Once this function is called, the 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.

5.2 IMS Connect Functions

The IMS e*Way has been developed using SeeBeyond's Monk Extension Kit. This mechanism requires that the supporting ***.dll** file be loaded using the **load-interface** procedure.

The entry-point for this ***.dll** file is **init**. Normally, users do not need to be concerned with the loading of this ***.dll** file because it is handled in the **ims-init.monk** script as per the default setting in the graphical user interface (GUI) control file.

In this script, a Monk global object named HIMS is defined and assigned a handle that refers to the ***.dll** file. From then on, the code may use either an explicit invoke procedure or an implied invoke as follows:

```
(invoke HIMS "close")
```

or

```
(HIMS "close")
```

These two procedure calls are equivalent and the implied style is used throughout the distributed code. Throughout the distributed code the Monk global variable HIMS is assumed to have been properly initialized.

For more information on the Monk programming language, see the *Monk Developer's Reference.*

This section describes the functions in the IMS Connect Monk extension. The functions are

init

Syntax

```
init
```

Description

init handles the initialization function.

Return Values

Monk Extension *.dll Handle
Returns a handle for use by the other functions.

Throws

None.

ack

Syntax

```
(HIMS "ack" acknak)
```

Description

ack specifies the acknowledgment function.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function.
acknak	Boolean	▪ #t for ACK ▪ #F for NAK

Return Values

Monk Vector

A Monk integer type, containing one element as follows:

- A value of zero indicates no errors have occurred.
- A negative number has one of the following meanings:

No.	Description
-1	Unable to load version 1.1 of the Winsock ***.dll**
-2	Unable to load Winsock ***.dll**
-3	A send error occurred at the socket level
-4	Unable to resolve hostname
-5	The specified port is invalid
-6	Cannot connect to the specified host/port
-7	The host has issued an unexpected shutdown on the connected socket
-8	[Not implemented]
-9	An error occurred during socket select
-10	Unable to acquire socket handle
-11	An unexpected error occurred during recv() processing
-12	Out of memory

Throws

A Monk exception is raised if any parameters are of the wrong type.

connect

Syntax

```
(HIMS "connect" hostname port)
```

Description

connect establishes a TCP/IP connection to the IMS TOC process.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function.
hostname	String	The system running IMS TOC; can be a name or a dotted IP address.
port	Integer	The port number on which IMS TOC is listening (1 through 65535).

Return Values

Monk Vector

A Monk integer type, containing one element as follows:

- A value of zero indicates no errors have occurred.
- A negative number indicates an error (see the table under **"Return Values" on page 57**).

Throws

A Monk exception is raised if any parameters are of the wrong type.

set_exit_params

Syntax

```
(HIMS "set-exit-params" userexit clientID mod-name commit-mode sync-
level command-type trancode datastore lterm id group password)
```

Description

set_exit_params stores a number of parameters in the handle, for future use.

Only one set of parameters may be defined per connection at a time; multiple **set_exit_params** calls cause the previous values to be replaced.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function. See **init** on page 56.
userexit	String	The user exit to invoke, for example "*SAMPLE*" for HWSSMPL0. These exits handle preprocessing of TCP/IP messages and postprocessing of output from IMS. For details see *IMS Connect Guide and Reference*.
clientID	String	Unique eight-character ID to identify this instance of the client to IMS Connect.
mod-name	Boolean	Describes the Boolean type response: **#t** if a *REQMOD* response is required in the output; **#f** if not
commit-mode	Character	The commit mode to use: ▪ **#\S** for send-then-commit ▪ **#\C** for commit-then-send. Requires “sync” mode to be #t.
sync-level	Boolean	The synchronization level to use for this transaction: ▪ **#t** for confirm ▪ **#f** for none
command-type	Character	One of the following values: ▪ Submit (#\space) for normal transaction input. ▪ ACK (#\A) ▪ NAK (#\N) ▪ Deallocate (#\D) ▪ Send-only (#\S) ▪ Resume-Tpipe (#\R)
trancode	String	The ID of the IMS transaction to be invoked.
datastore	String	The IMS database that is the destination.
lterm	String	Logical terminal override, or blanks.
id	String	The Resource Access Control Facility (RACF) user ID used for login.
group	String	The RACF group name of this user.

Name	Type	Description
password	String	The RACF user ticket or password for the RACF ID.

Return Values

Monk Vector

A Monk integer type, containing one element as follows:

- A value of zero indicates no errors have occurred.
- A positive number indicates the position of the first invalid parameter (for example, a character not in the known set or a number out of range).
- A negative number indicates an error (see the table under **"Return Values" on page 57**).

Throws

A Monk exception is raised if any parameters are of the wrong type.

send

Syntax

```
(HIMS "send" msg)
```

Description

send is used to deliver an input message. It returns immediately without waiting for a response. The **receive** function (see **receive** on page 62) must be used to fetch any response.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function. See **init** on page 56.
msg	String	The data to be sent to IMS.

Return Values

Monk Vector

A Monk integer type, containing one element as follows:

- A value of zero indicates no errors have occurred.
- A negative number indicates an error has occurred.

Throws

A Monk exception is raised if any parameters are of the wrong type.

receive

Syntax

```
(HIMS "receive" timeout merge)
```

Description

receive retrieves the response to the **send** function. See **send** on page 61.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function. See **init** on page 56.
timeout	Integer	The number of seconds to wait for a response or one of the following values: ▪ 0 for nonblocking poll ▪ -1 for a blocking call
merge	Boolean	Indicates whether the segmented data returned from IMS should be concatenated to one Monk string (#t) or returned as separate entities (#f).

Return Values

Monk Vector

Where polling of the host is being carried out, the normal return values (in the first element of the vector) are

- 1 - There is data available.
- 2 - There is no data available.

Where a timeout value is specified and the timer elapses prior to data arriving, the return value is 3.

Normal results from **receive** result in a string object being returned in the first element of the vector. If an IMS error occurs, this string contains the reserved value "*REQSTS*" in which case the vector contains an integer return code in element 2 and an integer reason code in element 3. Where IMS data segments are not being merged, then the vector contains as many string objects as there are segments in the response.

If a mod name has been requested (see **set_exit_params** on page 59), this is treated as a data segment and is returned as such.

Throws

A Monk exception is raised if any parameters are of the wrong type.

close

Syntax

```
(HIMS "close")
```

Description

close shuts down all the resources associated with the HIMS object. It also disconnects from the host if a connection is currently open.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function. See **init** on page 56.

Return Values

Monk Vector

A Monk integer type, containing one element as follows:

- A value of zero indicates no errors have occurred.
- A negative number indicates an error (see the table under **"Return Values" on page 57**).

Throws

A Monk exception is raised if any parameters are of the wrong type.

trace

Syntax

```
(HIMS "trace" filename)
```

Description

trace closes all the resourses associated with the HIMS object. It also disconnects from the host if a connection is currently open.

Parameters

Name	Type	Description
HIMS	Handle	The handle returned from the **init** function. See **init** on page 56.
filename	String	The name of a file in which network transmissions are dumped as a readable Hex/ASCII data; sends an empty string to disable tracing. Note that the specified file is overwritten if it already exists and if permissions allow; otherwise, it is created, subject to permissions.

Return Values

Monk Vector

A Monk integer type, containing one element as follows:

- A value of zero indicates no errors have occurred.
- A negative number indicates an error (see the table under **"Return Values" on page 57**).

Throws

A Monk exception is raised if any parameters are of the wrong type.

Index

U

Z