

***SeeBeyond ICAN Suite***

# **e\*Way Intelligent Adapter for PeopleSoft HTTP User's Guide**

*Release 5.0.5 for Schema Run-time Environment (SRE)*

*Monk Version*



The information contained in this document is subject to change and is updated periodically to reflect changes to the applicable software. Although every effort has been made to ensure the accuracy of this document, SeeBeyond Technology Corporation (SeeBeyond) assumes no responsibility for any errors that may appear herein. The software described in this document is furnished under a License Agreement and may be used or copied only in accordance with the terms of such License Agreement. Printing, copying, or reproducing this document in any fashion is prohibited except in accordance with the License Agreement. The contents of this document are designated as being confidential and proprietary; are considered to be trade secrets of SeeBeyond; and may be used only in accordance with the License Agreement, as protected and enforceable by law. SeeBeyond assumes no responsibility for the use or reliability of its software on platforms that are not supported by SeeBeyond.

SeeBeyond, e\*Gate, e\*Way, and e\*Xchange are the registered trademarks of SeeBeyond Technology Corporation in the United States and/or select foreign countries. The SeeBeyond logo, SeeBeyond Integrated Composite Application Network Suite, eGate, eWay, eInsight, eVision, eXchange, eView, eIndex, eTL, ePortal, eBAM, and e\*Insight are trademarks of SeeBeyond Technology Corporation. The absence of a trademark from this list does not constitute a waiver of SeeBeyond Technology Corporation's intellectual property rights concerning that trademark. This document may contain references to other company, brand, and product names. These company, brand, and product names are used herein for identification purposes only and may be the trademarks of their respective owners.

© 2005 SeeBeyond Technology Corporation. All Rights Reserved. This work is protected as an unpublished work under the copyright laws.

**This work is confidential and proprietary information of SeeBeyond and must be maintained in strict confidence.**

Version 20050406030159.

# Contents

---

<b>Preface</b>	<b>8</b>
Intended Reader	8
Organization	8
Nomenclature	9
Online Use	9
Writing Conventions	9
Additional Documentation	10
<hr/>	
<b>Chapter 1</b>	
<b>Introduction</b>	<b>11</b>
Overview	11
e*Way Components	12
Supported Operating Systems	13
<hr/>	
<b>Chapter 2</b>	
<b>Installation</b>	<b>14</b>
System Requirements	14
External System Requirements	15
External System Configuration	15
Installing the e*Way	16
Windows Systems	16
Installation Procedure	16
Subdirectories and Files	18
Environment Configuration	19
UNIX Systems	20
Installation Procedure	20
Subdirectories and Files	20
Environment Configuration	21
Installing the MUX Handler Classes	22

Windows Systems	22
UNIX Systems	24

## Chapter 3

<b>System Implementation</b>	<b>26</b>
<b>Overview</b>	<b>26</b>
Implementation Sequence	27
Viewing e*Gate Components	27
<b>Creating a Schema</b>	<b>28</b>
<b>Creating Event Types</b>	<b>29</b>
<b>Generating Event Type Definitions</b>	<b>29</b>
Generating and Publishing an XML Test Message	30
Extracting and Viewing the XML Test Message	36
Generating a DTD for the XML File	41
Generating an ETD from the DTD	44
<b>Assigning ETDs to Event Types</b>	<b>46</b>
<b>Defining Collaborations</b>	<b>47</b>
<b>Creating Intelligent Queues</b>	<b>48</b>
<b>Using the e*Way</b>	<b>49</b>
Publishing to PeopleSoft	49
XML Messages	49
Compressing the XML Message	50
Subscribing to PeopleSoft	52

## Chapter 4

<b>PeopleSoft 8 Setup</b>	<b>53</b>
<b>Overview</b>	<b>53</b>
<b>Configuring for Publication</b>	<b>54</b>
Creating PeopleSoft 8 Message Node for the e*Way	54
Activating the Message Definition for Publication	58
PeopleSoft 8 Message Definitions List	58
Defining Message Channel Routing Rules	61
Configuring the Message Channel	61
Defining Routing Directions for Message Nodes	65
Adding the PeopleSoft 8 Subscription Handler	67
<b>Configuring for Subscription</b>	<b>70</b>
Creating a MUX e*Way Message Node	70
Activating the Message Definition for Subscription	70
Defining the Message Channel Routing Rules	70
Adding the SeeBeyond MUX Subscription Handler	70

---

Chapter 5

<b>Setup Procedures</b>	<b>74</b>
<b>Overview</b>	<b>74</b>
<b>Setting Up the e*Way</b>	<b>75</b>
Defining e*Way	75
Modifying e*Way Properties	76
Configuring the e*Way	77
Using the e*Way Editor	78
Section and Parameter Controls	79
Parameter Configuration Controls	79
Command-line Configuration	80
Getting Help	80
Changing the User Name	81
Setting Startup Options or Schedules	81
Activating or Modifying Logging Options	83
Activating or Modifying Monitoring Thresholds	84
<b>Troubleshooting the e*Way</b>	<b>85</b>
Configuration Problems	85
System-related Problems	86

---

Chapter 6

<b>Operational Overview</b>	<b>87</b>
<b>e*Way Architecture</b>	<b>87</b>
<b>Basic e*Way Processes</b>	<b>89</b>
Initialization Process	90
Connect to External Process	91
Data Exchange Process	92
Disconnect from External Process	95
Shutdown Process	95

---

Chapter 7

<b>Configuration Parameters</b>	<b>96</b>
<b>Overview</b>	<b>96</b>
<b>General Settings</b>	<b>97</b>
Journal File Name	97
Max Resends Per Message	97
Max Failed Messages	97
Forward External Errors	98
<b>Communication Setup</b>	<b>99</b>
Exchange Data Interval	99
Zero Wait Between Successful Exchanges	99
Start Exchange Data Schedule	100
Stop Exchange Data Schedule	101
Down Timeout	101

Up Timeout	101
Resend Timeout	101
<b>Monk Configuration</b>	<b>102</b>
Specifying Function or File Names	102
Specifying Multiple Directories	102
Load Path	102
Additional Path	102
Auxiliary Library Directories	103
Monk Environment Initialization File	103
Startup Function	104
Process Outgoing Message Function	104
Exchange Data with External Function	105
External Connection Establishment Function	106
External Connection Verification Function	106
External Connection Shutdown Function	107
Positive Acknowledgment Function	107
Negative Acknowledgment Function	108
Shutdown Command Notification Function	109
<b>HTTP Configuration</b>	<b>110</b>
Request	110
Timeout	110
URL	110
User Name	110
Encrypted Password	111
Agent	111
Content-type	111
Request-content	111
Accept-type	112
<b>HTTP Proxy Configuration</b>	<b>113</b>
Use Proxy Server	113
User Name	113
Encrypted Password	113
Server Address	114
Port Number	114
<b>HTTP SSL Configuration</b>	<b>115</b>
Use SSL	115
Trusted CA Certificates Directory	115
Use Client Certificate Map	115
Client Certificate Map File	116
<b>PeopleSoft Application Messaging</b>	<b>117</b>
Base64 Deflate	117
Request Version	117
To Node	117
From Node	118
Encrypted Password	118
Channel	118
Subject	119
Message Version	119
Subject Detail	119
Publication ID	119
Subchannel	120
Originating Node	120
Publisher	121
Publication Process	121
Default Version	121

---

**Chapter 8**

<b>API Functions</b>	<b>122</b>
<b>Overview</b>	<b>122</b>
<b>Helper Functions</b>	<b>123</b>
psoft8-helper-ping	123
psoft8-helper-ping-response-isok	123
psoft8-helper-print-exception	124
psoft8-helper-pub-response-isok	124
psoft8-helper-wrap-xml	125
<b>Standard e*Way Functions</b>	<b>126</b>
psoft8-ack	126
psoft8-connect	127
psoft8-exchange	127
psoft8-init	128
psoft8-nack	128
psoft8-notify	129
psoft8-outgoing	129
psoft8-shutdown	130
psoft8-startup	131
psoft8-verify	131
<b>Generic e*Way Functions</b>	<b>133</b>
event-commit-to-egate	133
event-rollback-to-egate	134
event-send-to-egate	134
event-send-to-egate-ignore-shutdown	135
event-send-to-egate-no-commit	135
get-logical-name	136
insert-exchange-data-event	136
send-external-up	137
send-external-down	137
shutdown-request	138
start-schedule	138
stop-schedule	139
waiting-to-shutdown	139

---

**Appendix A**

<b>MUX Subscription Handler</b>	<b>140</b>
<b>Java Classes</b>	<b>140</b>
Entry.class	140
MuxHandlerConstants.class	140
MuxHandlerEntry.class	141
AdministerMuxHandler.class	141
AdministerMuxHandlerAddMode.class	141
AdministerMuxHandlerDeleteMode.class	141
AdministerMuxHandlerEditMode.class	141
AdministerMuxHandlerError.class	141
MuxPublicationHandler.class	141
MuxHandler.class	142

<b>Index</b>	<b>143</b>
--------------	------------

# Preface

This Preface contains information regarding the User's Guide itself.

---

## P.1 Intended Reader

The reader of this guide is presumed to be a developer or system administrator with responsibility for maintaining the SeeBeyond™ e\*Gate™ Integrator system, and have a working knowledge of:

- Operation and administration of the appropriate operating systems (see [Supported Operating Systems](#) on page 13)
- Windows-style GUI operations
- PeopleSoft concepts and operations
- Integrating PeopleSoft applications with external systems

---

## P.2 Organization

This User's Guide is organized into two parts. The first part, consisting of Chapters 1-5, introduces the e\*Way and describes the procedures for installing and setting up the e\*Way, and implementing a working system incorporating the e\*Way. This part should be of particular interest to a System Administrator or other user charged with the task of getting the system up and running.

The second part, consisting of Chapters 6-8, describes the details of e\*Way operation and configuration, including descriptions of the API functions. This part should be of particular interest to a Developer involved in customizing the e\*Way for a specific purpose. Information contained in this part that is necessary for the initial setup of the e\*Way is cross-referenced in the first part of the guide, at the appropriate points in the procedures.



---

## P.3 Nomenclature

Note that for purposes of brevity, the e\*Way Intelligent Adapter for PeopleSoft (HTTP) is frequently referred to as the PeopleSoft HTTP e\*Way, or simply the e\*Way.

---

## P.4 Online Use

This User's Guide is provided in Adobe Acrobat's Portable Document Format (PDF). As such, it can be printed out on any printer or viewed online. When viewing online, you can take advantage of the extensive hyperlinking imbedded in the document to navigate quickly throughout the Guide.

Hyperlinking is available in:

- The Table of Contents
- The Index
- Within the chapter text, indicated by **blue print**

Existence of a hyperlink *hotspot* is indicated when the hand cursor points to the text. Note that the hotspots in the Index are the *page numbers*, not the topics themselves. Returning to the spot you hyperlinked from is accomplished by right-clicking the mouse and selecting **Go To Previous View** on the resulting menu.

---

## P.5 Writing Conventions

The writing conventions listed in this section are observed throughout this document.

### Monospaced (Courier) Font

Computer code and text to be typed at the command line are set in Courier as shown below.

```
Configuration for BOB_Promotion
java -jar ValidationBuilder.jar
```

Variables within a command line are set in Courier italic as shown below.

```
stcregutl -rh host-name -un user-name -up password -sf
```

### Bold Sans-serif Font

- User Input: Click **Apply** to save, or **OK** to save and close.
- File Names and Paths: In the **Open** field, type **D:\setup\setup.exe**.
- Parameter, Function, and Command Names: The default parameter **localhost** is normally only used for testing; the Monk function **iq-put** places an Event into an IQ.

---

## P.6 Additional Documentation

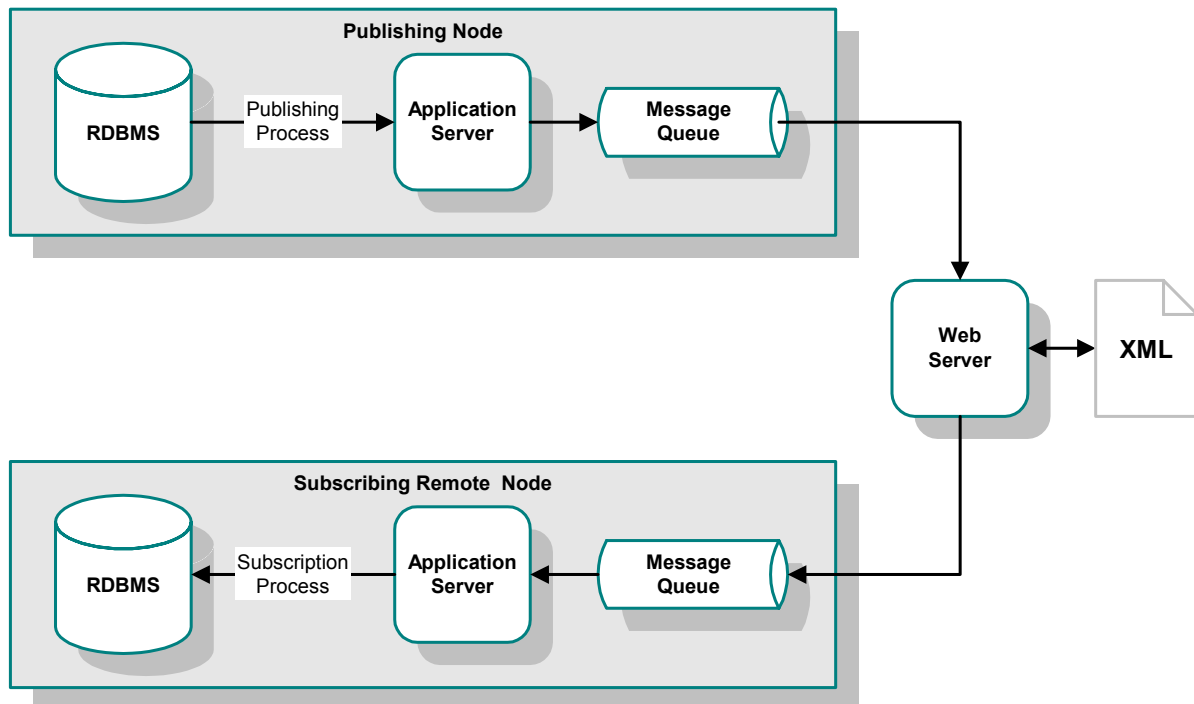
- Many of the procedures included in this User's Guide are described in greater detail in the *e\*Gate Integrator User's Guide*.
- For descriptions of the HTTP e\*Way, see the *HTTP e\*Way Intelligent Adapter User's Guide*.
- For descriptions of the e\*Gate API Kit, see the *e\*Gate API Kit User's Guide*.
- For detailed information regarding the DTD Builder or other components of the XML Toolkit, see the *XML Toolkit User's Guide*.
- For information on Application Messaging and PeopleSoft 8 Integration Technology, please refer to the *PeopleSoft 8 PeopleTools* documentation.
- For additional information on Message Definitions, see the *PeopleSoft 8 EIP Catalog*.

# Introduction

## 1.1 Overview

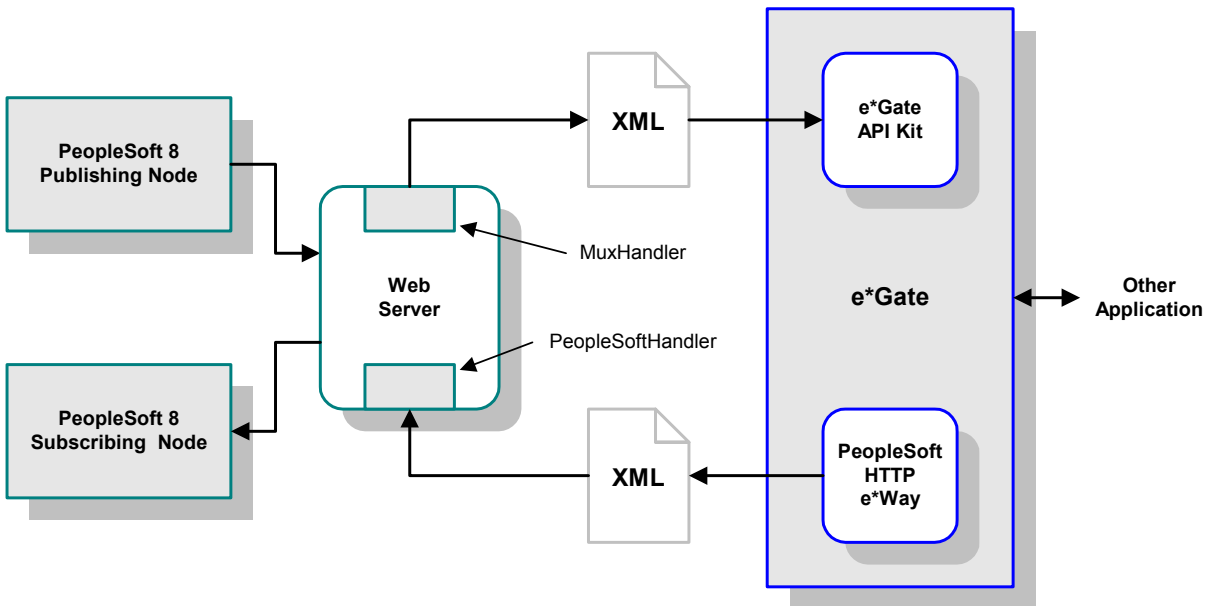
The PeopleSoft HTTP e\*Way provides a means of sending data to PeopleSoft 8 by using PeopleSoft 8 Application Messaging technology. The Application Messaging model allows for publication and subscription of XML messages using HTTP. This clean and flexible solution allows for implementation to be accomplished at the business level by means of XML messages. Figure 1 diagrams the PeopleSoft Application Messaging architecture.

**Figure 1** PeopleSoft Application Messaging Architecture



Bidirectional data exchange with PeopleSoft 8 requires the use of the PeopleSoft HTTP e\*Way, the e\*Gate API Kit, and SeeBeyond's customized PeopleSoft 8 MUX subscription handler classes. The PeopleSoft HTTP e\*Way is used to publish data to PeopleSoft 8, and the API Kit is used in conjunction with the SeeBeyond MUX subscription handler classes to receive data from PeopleSoft 8. Figure 2 illustrates the integration architecture.

Figure 2 PeopleSoft 8 - e\*Gate Integration



## 1.2 e\*Way Components

The Monk-enabled PeopleSoft HTTP e\*Way incorporates the following components:

- Executable files:
  - ♦ `stcewgenericmonk.exe` (installed with e\*Gate Integrator), for e\*Gate-to-PeopleSoft operation
  - ♦ `stcewipmp.exe`, for PeopleSoft-to-e\*Gate operation
- Default configuration file:
  - ♦ `stcewpsft8.def`
- Supporting Monk functions and scripts, and Event Type Definitions

For a list of installed files, see [Chapter 2](#).

---

## 1.3 Supported Operating Systems

The e\*Way Intelligent Adapter for PeopleSoft HTTP currently supports the following combinations of operating systems and PeopleSoft system components.

- Windows 2000 and Windows Server 2003
- IBM AIX 5.1L
- Sun Solaris 8
- Japanese Windows 2000 and Windows Server 2003
- Japanese Sun Solaris 8

**Note:** *The e\*Gate Schema Designer runs only on Windows.*

# Installation

This chapter describes the requirements and procedures for installing the e\*Way software. Procedures for implementing a working system, incorporating instances of the e\*Way, are described in [Chapter 3](#).

**Note:** Please read the *readme.txt* file located in the *addons\ewpsoft8* directory on the installation CD-ROM for important information regarding this installation.

---

## 2.1 System Requirements

To use the e\*Way Intelligent Adapter for PeopleSoft HTTP, you need the following:

- 1 An e\*Gate Participating Host.
- 2 A TCP/IP network connection.
- 3 Sufficient free disk space to accommodate e\*Way files:
  - ♦ Approximately 4 MB on Windows systems
  - ♦ Approximately 34 MB on Solaris systems
  - ♦ Approximately 28 MB on AIX systems

**Note:** 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.

---

## 2.2 External System Requirements

The e\*Way Intelligent Adapter for PeopleSoft HTTP requires the following external system components:

- PeopleSoft 8 with PeopleTools 8.13
- PeopleSoft 8.4 with PeopleTools 8.42
- For IBM AIX systems, a back-end Oracle 8.1.6 RDBMS

### 2.2.1 External System Configuration

To publish XML messages to, or subscribe to XML messages from PeopleSoft 8, Message Nodes, Messages, Message Channels, and Subscription Handlers must be created and configured within the PeopleSoft 8 environment. See [Chapter 4](#) for procedural details.

## 2.3 Installing the e\*Way

### 2.3.1 Windows Systems

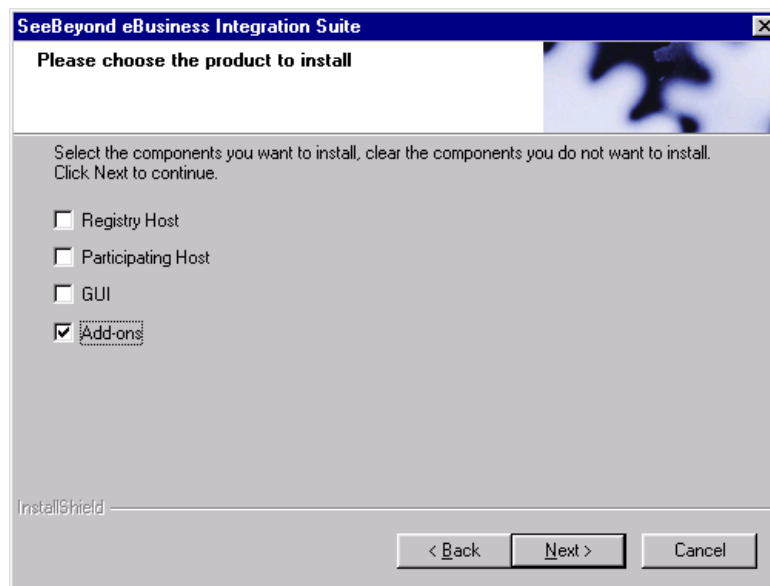
#### Installation Procedure

**Note:** *The installation utility detects and suggests the appropriate installation directory. Use this directory unless advised otherwise by SeeBeyond.*

#### To Install the e\*Way on a Microsoft Windows System

- 1 Log in as an Administrator on the workstation on which you want to install the e\*Way (you must have Administrator privileges to install this e\*Way).
- 2 Exit all Windows programs and disable any anti-virus applications before running the setup program.
- 3 Insert the e\*Way installation CD-ROM into the CD-ROM drive.
- 4 Launch the setup program.
  - A If the CD-ROM drive's Autorun feature is enabled, the setup program should launch automatically. Follow the on-screen instructions until the **Choose Product** dialog box appears (see Figure 3). Check **Add-ons**, then click **Next**.

**Figure 3** Choose Product Dialog



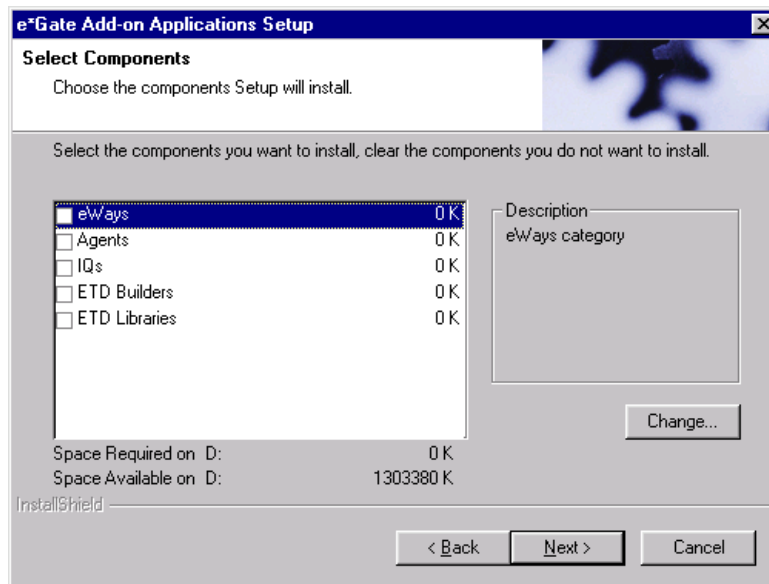
- B If the setup program does not launch automatically, use the Windows Explorer or the Control Panel's **Add/Remove Applications** feature to launch the following file on the CD-ROM drive (bypassing the **Choose Product** dialog):

setup\addons\setup.exe



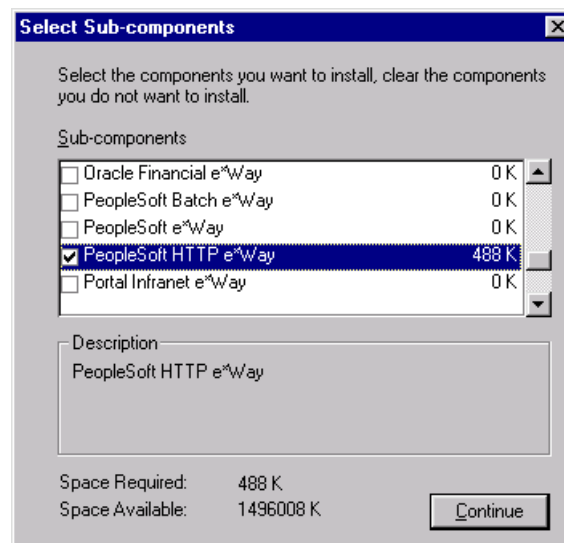
- 5 Follow the on-screen instructions until the **Select Components** dialog box appears (see Figure 4). Highlight—but do not check—**eWays** and then click **Change**.

**Figure 4** Select Components Dialog



- 6 When the Select Sub-components dialog box appears (see Figure 5), check the **PeopleSoft HTTP e\*Way**.

**Figure 5** Select Sub-components Dialog



- 7 Click **Continue**, and the Select Components dialog box reappears.
- 8 Click **Next** and continue with the installation.
- 9 After the e\*Way has been installed, you need to install the SeeBeyond MUX Subscription Handler. See [Windows Systems](#) on page 22.

## Subdirectories and Files

**Note:** *Installing the e\*Way Intelligent Adapter for PeopleSoft HTTP installs both Java and Monk versions. Only the files used by the Monk version are listed in this section.*

By default, the InstallShield installer installs the following file within the `\eGate\Server\registry\repository\default` tree on the Registry Host.

**Table 1** Registry Host Only

Subdirectories	Files
\	stcewsoft8.ctl

By default, the InstallShield installer also creates the following subdirectories and installs the following files within the `\eGate\client` tree on the Participating Host, and the `\eGate\Server\registry\repository\default` tree on the Registry Host.

**Table 2** Participating Host & Registry Host

Subdirectories	Files
<code>\configs\stcewgenericmonk\</code>	stcewsoft8.def
<code>\monk_library\</code>	ewpsoft8.gui
<code>\monk_library\ewpsoft8\</code>	psoft8-ack.monk psoft8-connect.monk psoft8-exchange.monk psoft8-helper-ping-response-isok.tsc psoft8-helper-ping.monk psoft8-helper-print-exception.tsc psoft8-helper-pub-response-isok.tsc psoft8-helper-wrap-xml.monk psoft8-init.monk psoft8-nack.monk psoft8-notify.monk psoft8-outgoing.monk psoft8-shutdown.monk psoft8-startup.monk psoft8-verify.monk
<code>\monk_scripts\common\</code>	psoft8-header-pubmsg.ssc psoft8-ping-response.ssc psoft8-pub-response.ssc psoft8-reply-exception.ssc
<code>\pkicerts\client\</code>	certmap.txt
<code>\pkicerts\trusedcas\</code>	GTECyberTrustGlobalRoot.cer MicrosoftRootAuthority.cer SecureServerCertificationAuthority.cer ThawtePremiumServerCA.cer ThawteServerCA.cer verisign_class3.cer

***Note:** Installing the e\*Way Intelligent Adapter for PeopleSoft HTTP also installs the HTTP e\*Way and the e\*Gate API Kit. See the respective User's Guides for information on components installed with those e\*Ways.*

## Environment Configuration

No changes are required to the Participating Host's operating environment to support this e\*Way.

## 2.3.2 UNIX Systems

### Installation Procedure

**Note:** *You are not required to have root privileges to install this e\*Way. Log on under the user name that you wish to own the e\*Way files. Be sure that this user has sufficient privilege to create files in the e\*Gate directory tree.*

#### To install the e\*Way on a UNIX system

- 1 Log onto the workstation containing the CD-ROM drive and, if necessary, mount the drive.
- 2 Insert the e\*Way installation CD-ROM into the CD-ROM drive.
- 3 At the shell prompt, type  
`cd /cdrom`
- 4 Start the installation script by typing:  
`setup.sh`
- 5 A menu appears, containing several options. Select the **Install e\*Way** option, and follow any additional on-screen directions.

**Note:** *The installation utility detects and suggests the appropriate installation directory. Use this directory unless advised otherwise by SeeBeyond. Note also that **no spaces** should appear in the installation path name.*

- 6 After the e\*Way has been installed, you need to install the SeeBeyond MUX Subscription Handler. See **UNIX Systems** on page 24.

### Subdirectories and Files

**Note:** *Installing the e\*Way Intelligent Adapter for PeopleSoft HTTP installs both Java and Monk versions. Only the files used by the Monk version are listed in this section.*

The preceding installation procedure installs the following file only within the `/eGate/Server/registry/repository/default` tree on the Registry Host.

**Table 3** Registry Host Only

Subdirectories	Files
/	stcewpsft8.ctl

The preceding installation procedure also creates the following subdirectories and installs the following files within the `/eGate/client` tree on the Participating Host, and the `/eGate/Server/registry/repository/default` tree on the Registry Host.

**Table 4** Participating Host & Registry Host

Subdirectories	Files
/configs/stcewgenericmonk/	stcewsoft8.def
/monk_library/	ewpsoft8.gui
/monk_library/ewpsoft8/	psoft8-ack.monk psoft8-connect.monk psoft8-exchange.monk psoft8-helper-ping-response-isok.tsc psoft8-helper-ping.monk psoft8-helper-print-exception.tsc psoft8-helper-pub-response-isok.tsc psoft8-helper-wrap-xml.monk psoft8-init.monk psoft8-nack.monk psoft8-notify.monk psoft8-outgoing.monk psoft8-shutdown.monk psoft8-startup.monk psoft8-verify.monk
/monk_scripts/common/	psoft8-header-pubmsg.ssc psoft8-ping-response.ssc psoft8-pub-response.ssc psoft8-reply-exception.ssc
/pkicerts/client/	certmap.txt
/pkicerts/trustedcas/	GTECyberTrustGlobalRoot.cer MicrosoftRootAuthority.cer SecureServerCertificationAuthority.cer ThawtePremiumServerCA.cer ThawteServerCA.cer verisign_class3.cer

**Note:** *Installing the e\*Way Intelligent Adapter for PeopleSoft HTTP also installs the HTTP e\*Way and the e\*Gate API Kit. See the respective User's Guides for information on components installed with those e\*Ways.*

## Environment Configuration

No changes are required to the Participating Host's operating environment to support this e\*Way.

## 2.4 Installing the MUX Handler Classes

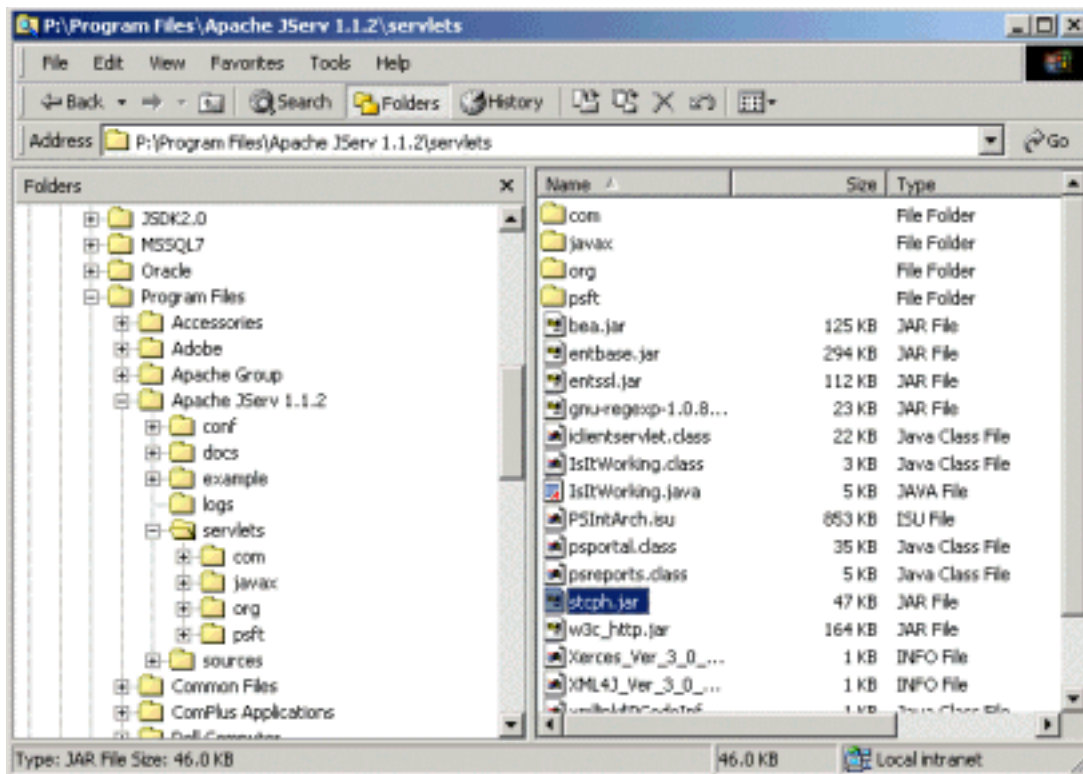
The SeeBeyond MUX Subscription Handler consists of ten java classes, which are contained in the `stcph.jar` file. This file is available when the e\*Gate participating host is installed. The class descriptions are given in [Appendix A](#).

### 2.4.1 Windows Systems

- 1 Stop the Apache Web server (for PeopleSoft 8.13, it is Apache JServ 1.1.2)
- 2 Stop the PeopleSoft 8 Application Server for the appropriate Domain.
- 3 Copy the `stcph.jar` file from the e\*Gate Installation CD to the `servlets` subdirectory under the servlet engine installation directory.

Figure 6, which reflects an installation of PeopleSoft 8.13 on Windows 2000, shows the location where the `jar` file must be copied.

**Figure 6** servlets Directory - Windows



- 4 Add the path to the `stcph.jar` file in the `CLASSPATH` for the servlet engine.

For PeopleSoft 8.13, which uses the Apache JServ 1.3 servlet engine, there is a configuration file called `jserv.properties` which resides in the `\conf` subdirectory of the servlet engine installation directory.

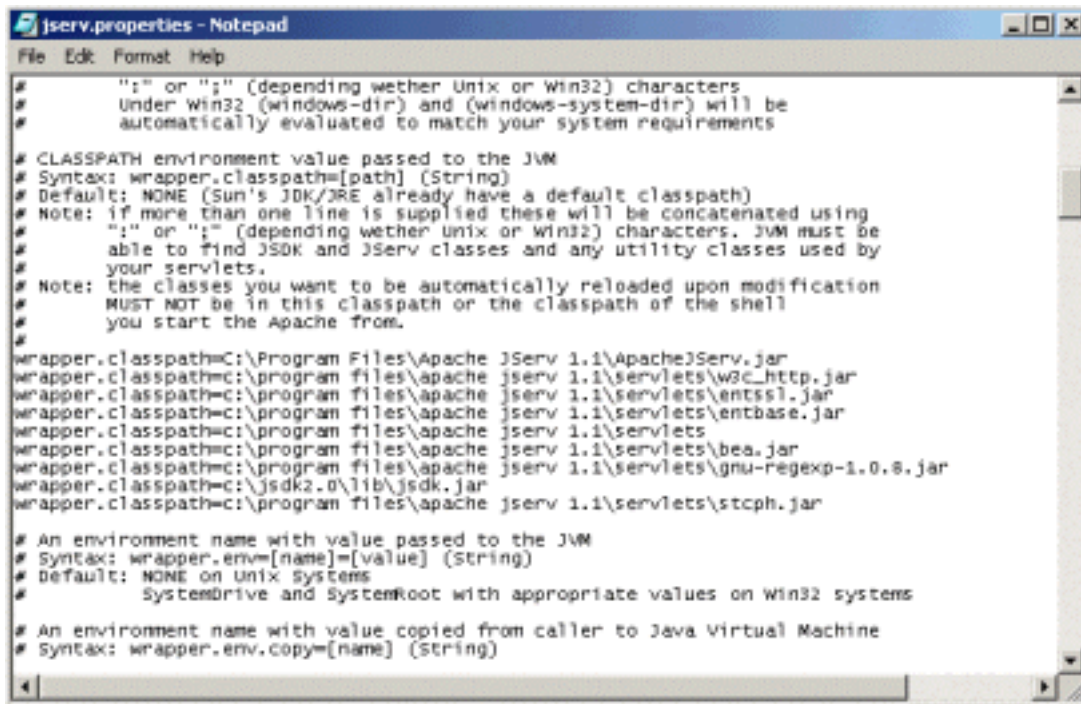
The following line must be added to the section called `CLASSPATH`:

```
wrapper.classpath=<servlet_engine_dir>\servlets\stcph.jar
```

where `servlet_engine_dir` is replaced with the name of the servlet engine installation directory (in the example, `Apache JServ 1.1.2`).

- 5 Add the entry to the `jserv.properties` file for the appropriate platform, as shown in Figure 7.

**Figure 7** jserv.properties File



```
File Edit Format Help
# ":" or ";" (depending wether Unix or Win32) characters
# Under Win32 (windows-dir) and (windows-system-dir) will be
# automatically evaluated to match your system requirements
# CLASSPATH environment value passed to the JVM
# Syntax: wrapper.classpath=[path] (String)
# Default: NONE (Sun's JDK/JRE already have a default classpath)
# Note: if more than one line is supplied these will be concatenated using
# ":" or ";" (depending wether unix or win32) characters. JVM must be
# able to find JSJK and JServ classes and any utility classes used by
# your servlets.
# Note: the classes you want to be automatically reloaded upon modification
# MUST NOT be in this classpath or the classpath of the shell
# you start the Apache from.
#
wrapper.classpath=C:\Program Files\Apache JServ 1.1\ApacheJServ.jar
wrapper.classpath=c:\program files\apache jserv 1.1\servlets\w3c_http.jar
wrapper.classpath=c:\program files\apache jserv 1.1\servlets\entss1.jar
wrapper.classpath=c:\program files\apache jserv 1.1\servlets\entbase.jar
wrapper.classpath=c:\program files\apache jserv 1.1\servlets
wrapper.classpath=c:\program files\apache jserv 1.1\servlets\bea.jar
wrapper.classpath=c:\program files\apache jserv 1.1\servlets\gnu-regexp-1.0.8.jar
wrapper.classpath=c:\jdk2.0\lib\jdk.jar
wrapper.classpath=c:\program files\apache jserv 1.1\servlets\stcph.jar
# An environment name with value passed to the JVM
# Syntax: wrapper.env=[name]=[value] (String)
# Default: NONE on unix systems
# SystemDrive and SystemRoot with appropriate values on Win32 systems
# An environment name with value copied from caller to Java Virtual Machine
# Syntax: wrapper.env.copy=[name] (String)
```

- 6 Start (boot) the Application Server for the appropriate domain.
- 7 Start the Apache Web server.

## 2.4.2 UNIX Systems

**Note:** You must have `jdk 1.2` or later installed on the host to run the `jar` command.

- 1 Stop the Web server.
- 2 Stop the PeopleSoft 8 Application Server for the appropriate Domain.
- 3 Copy the `stcph.jar` file from the e\*Gate Participating Host to the directory where servlets must reside. The location depends upon the Web server you are using.
  - A For Apache, copy the `stcph.jar` file to the `servlets` subdirectory under the `webserv` directory in the PeopleSoft Domain installation; for example:

**/psoft/FDM80/webserv/servlets**

(as shown in Figure 8).

**Figure 8** servlets Directory - Apache

```

total 1976
drwxr-sr-x  7 psoft  sys      1024 Jun 25 16:47 .
drwxr-sr-x 15 psoft  sys      512 Jun 19 15:16 ..
-rw-r--r--  1 psoft  sys     1134 Oct 08 2000 Hello.class
-rw-r--r--  1 psoft  sys     1303 Oct 08 2000 Hello.java
-rw-r--r--  1 psoft  sys     2119 Oct 08 2000 IsItWorking.class
-rw-r--r--  1 psoft  sys     4542 Oct 08 2000 IsItWorking.java
drwxr-sr-x  2 psoft  sys      512 Jun 25 16:47 META-INF
-rwxr-xr-x  1 psoft  sys       7 Feb 28 20:15 XML4J_Ver_3_0_1.info
-rwxr-xr-x  1 psoft  sys       7 Feb 28 20:15 Xerces_Ver_3_0_1.info
-rwxr-xr-x  1 psoft  sys    127587 Feb 28 16:46 bea.jar
drwxr-sr-x  6 psoft  sys      512 Jun 25 16:47 com
-rwxr-xr-x  1 psoft  sys    300083 Feb 28 16:26 entbase.jar
-rwxr-xr-x  1 psoft  sys    113748 Feb 28 16:26 entssl.jar
-rwxr-xr-x  1 psoft  sys    23153 Feb 28 16:46 gnu-regexp-1.0.8.jar
-rwxr-xr-x  1 psoft  sys    22165 Feb 28 17:02 iclientervlet.class
drwxr-sr-x  4 psoft  sys      512 Jun 19 15:18 javax
-r--r--r--  1 psoft  sys    76453 Oct 08 2000 jsdk.jar
drwxr-sr-x  6 psoft  sys      512 Feb 28 20:15 org
drwxr-sr-x  3 psoft  sys      512 Feb 28 20:15 psft
-rwxr-xr-x  1 psoft  sys    35724 Feb 28 17:13 psportal.class
-rwxr-xr-x  1 psoft  sys     5020 Feb 28 17:02 psreports.class
-rw-r----- 1 psoft  sys     47191 Jun 25 16:41 stcph.jar
-rwxr-xr-x  1 psoft  sys    167275 Feb 28 16:26 w3c_http.jar
-rwxr-xr-x  1 psoft  sys      539 Feb 28 17:03 xlink$PCoInfo.class
-rwxr-xr-x  1 psoft  sys     10037 Feb 28 17:03 xlink.class
$ pwd
/psoft/FDM80/webserv/servlets
$

```

- A For WebLogic, copy the `stcph.jar` file to the `servletclasses` subdirectory under the `weblogic/myserver` directory in the PeopleSoft Domain installation; for example:

**/do1/psoft/fdm80/weblogic/myserver/servletclasses**

(as shown in Figure 9).

- 4 Extract the contents of the `stcph.jar` file.
 

Use the command:

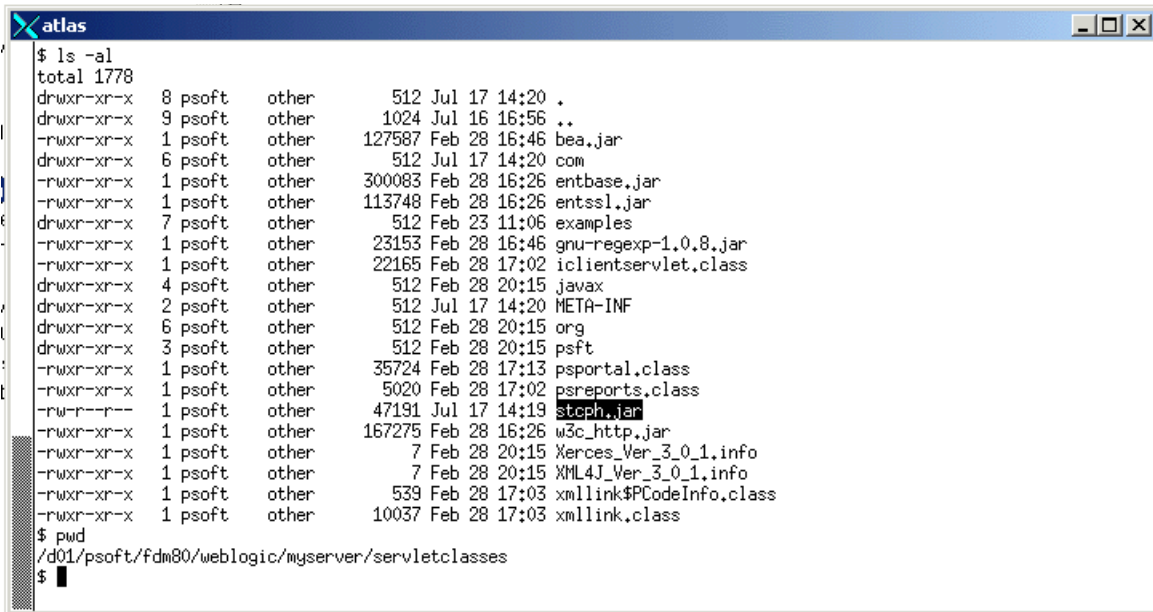
```
jar -tf stcph.jar
```

to extract the classes.
- 5 Start (boot) the Application Server for the appropriate domain.



- 6 Start the Web server.

**Figure 9** servlets Directory - WebLogic



```
atlas
$ ls -al
total 1778
drwxr-xr-x  8 psoft  other      512 Jul 17 14:20 .
drwxr-xr-x  9 psoft  other     1024 Jul 16 16:56 ..
-rwxr-xr-x  1 psoft  other    127587 Feb 28 16:46 bea.jar
drwxr-xr-x  6 psoft  other      512 Jul 17 14:20 com
-rwxr-xr-x  1 psoft  other    300083 Feb 28 16:26 entbase.jar
-rwxr-xr-x  1 psoft  other    113748 Feb 28 16:26 entssl.jar
drwxr-xr-x  7 psoft  other      512 Feb 23 11:06 examples
-rwxr-xr-x  1 psoft  other    23153 Feb 28 16:46 gnu-regexp-1.0.8.jar
-rwxr-xr-x  1 psoft  other    22165 Feb 28 17:02 iclientervlet.class
drwxr-xr-x  4 psoft  other      512 Feb 28 20:15 javax
drwxr-xr-x  2 psoft  other      512 Jul 17 14:20 META-INF
drwxr-xr-x  6 psoft  other      512 Feb 28 20:15 org
drwxr-xr-x  3 psoft  other      512 Feb 28 20:15 psft
-rwxr-xr-x  1 psoft  other    35724 Feb 28 17:13 psportal.class
-rwxr-xr-x  1 psoft  other    5020 Feb 28 17:02 psreports.class
-rw-r--r--  1 psoft  other    47191 Jul 17 14:19 stoph.jar
-rwxr-xr-x  1 psoft  other   167275 Feb 28 16:26 w3c_http.jar
-rwxr-xr-x  1 psoft  other      7 Feb 28 20:15 Xerces_Ver_3_0_1.info
-rwxr-xr-x  1 psoft  other      7 Feb 28 20:15 XML4J_Ver_3_0_1.info
-rwxr-xr-x  1 psoft  other    539 Feb 28 17:03 xlink$PCCodeInfo.class
-rwxr-xr-x  1 psoft  other   10037 Feb 28 17:03 xlink.class
$ pwd
/d01/psoft/fdm80/weblogic/myserver/servletclasses
$
```

# System Implementation

In this chapter we summarize the procedures required for implementing a working system incorporating the e\*Way Intelligent Adapter for PeopleSoft HTTP. Please see the *e\*Gate Integrator User's Guide* for additional details.

---

## 3.1 Overview

This e\*Way provides a specialized transport component for incorporation into an operational Schema. The schema also contains Collaborations, linking different data or Event types, and Intelligent Queues. Typically, other e\*Way types also are used as components of the Schema.

Topics included in this chapter include:

[Creating a Schema](#) on page 28

[Creating Event Types](#) on page 29

[Generating Event Type Definitions](#) on page 29

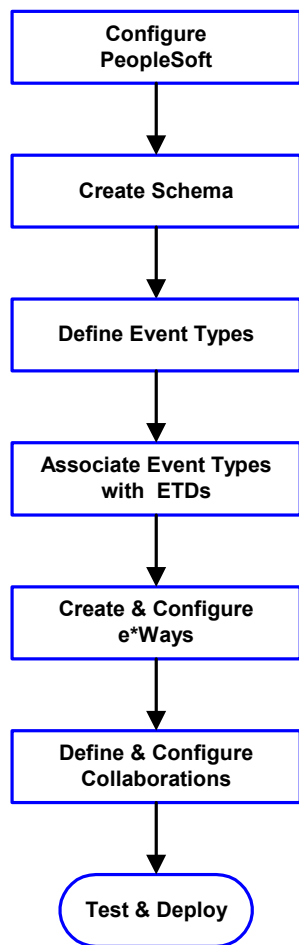
[Assigning ETDs to Event Types](#) on page 46

[Defining Collaborations](#) on page 47

[Creating Intelligent Queues](#) on page 48

[Using the e\\*Way](#) on page 49

### 3.1.1 Implementation Sequence



- 1 The first step is to configure the PeopleSoft application to interact properly with the e\*Way (see [Chapter 4](#)).
- 2 The second step is to create a new Schema—the subsequent steps apply only to this Schema (see [Creating a Schema](#) on page 28).
- 3 The third step is to define the Event Types you are transporting and processing within the Schema (see [Creating Event Types](#) on page 29).
- 4 Next, you need to associate the Event Types created in the previous step with Event Type Definitions (ETDs) derived from the applicable Business Rules (see [Generating Event Type Definitions](#) on page 29).
- 5 The fifth step is to create and configure the required e\*Ways (see [Chapter 5](#)).
- 6 Next is to define and configure the Collaborations linking the Event Types from step 3 (see [Defining Collaborations](#) on page 47).
- 7 Finally, you must test your Schema. Once you have verified that it is working correctly, you may deploy it to your production environment.

### 3.1.2 Viewing e\*Gate Components

Use the Navigator and Editor panes of the e\*Gate Schema Designer to view the various e\*Gate components. Note that you may only view components of a single schema at one time, and that all operations apply only to the current schema. All procedures in this chapter should be performed while displaying the **Components** Navigator pane. See the *e\*Gate Integrator User's Guide* for a detailed description of the features and use of the Schema Designer.

## 3.2 Creating a Schema

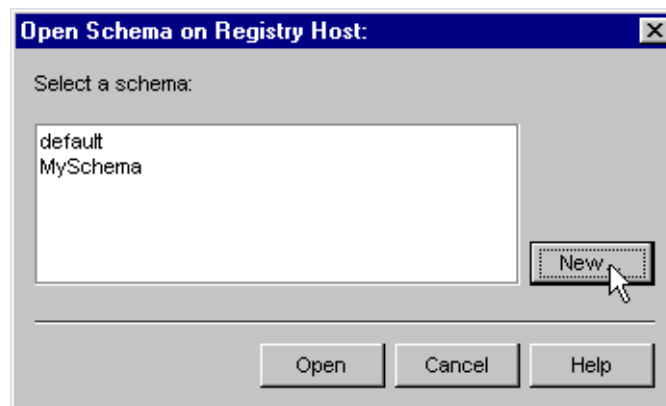
A schema is the structure that defines e\*Gate system parameters and the relationships between components within the e\*Gate system. Schemas can span multiple hosts.

Because all setup and configuration operations take place within an e\*Gate schema, a new schema must be created, or an existing one must be started before using the system. Schemas store all their configuration parameters in the e\*Gate Registry.

To select or create a schema

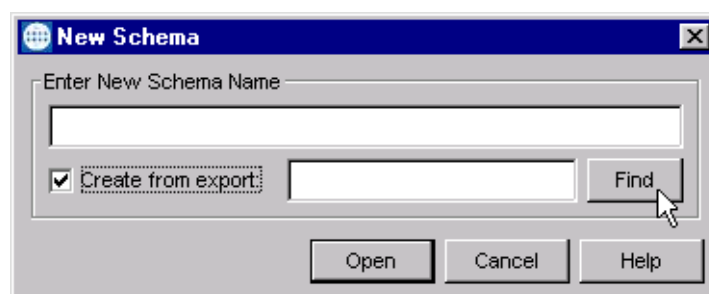
- 1 Invoke the **Open Schema** dialog box and **Open** an existing schema or click **New** to create a new schema.

**Figure 10** Open Schema Dialog



- 2 Clicking **New** invokes the **New Schema** dialog box (Figure 11).

**Figure 11** New Schema Dialog




- 3 Enter a new schema name and click **Open**.
- 4 The e\*Gate Schema Designer then opens under your new schema name.
- 5 From the **Options** menu, click on **Default Editor** and select **Monk**.
- 6 Select the **Components** tab, found at the bottom of the Navigator pane of the e\*Gate Schema Designer window.
- 7 You are now ready to begin creating the necessary components for this new schema.

---

## 3.3 Creating Event Types

Within e\*Gate, messages and/or packages of data are defined as Events. Each Event must be categorized into a specific Event Type within the schema.

### To define the Event Types

- 1 In the e\*Gate Schema Designer's Navigator pane, select the **Event Types** folder.
- 2 On the Palette, click the **New Event Type** button .
- 3 In the **New Event Type Component** box, enter the name for the input Event Type and click **Apply**. Use this method to create all required Event Types, for example:
  - ♦ **InboundEvent**
  - ♦ **ValidEvent**
  - ♦ **InvalidEvent**
- 4 After you have created the final Event Type, click **OK**.

---

## 3.4 Generating Event Type Definitions

As the name implies, an Event Type Definition (ETD) defines the structure of the Event Types employed in your Schema. Any one ETD can be associated with more than one Event Type within the schema. In the PeopleSoft HTTP e\*Way, ETDs are created semi-automatically using the DTD Builder, once a DTD has been generated.

As of release 8.13, PeopleSoft does not offer a DTD generation utility, nor does it include any sample DTDs. However, the PeopleSoft 8.13 Application Designer can be extended using third-party software to generate DTDs, which then can be converted to ETDs using SeeBeyond software. This chapter describes a "workaround" procedure for PeopleSoft 8.13 and earlier versions.

**Note:** *Before attempting the procedure described in this section, you should be familiar with PeopleSoft8, People Tools, Application Designer, Application Messaging architecture, and Third Party Integration technology.*

The workaround procedure involves several sequential steps, which are described under the following headings:

- 1 **Generating and Publishing an XML Test Message** on page 30.
- 2 **Extracting and Viewing the XML Test Message** on page 36.
- 3 **Generating a DTD for the XML File** on page 41.

**Note:** *The procedure described may not work for all Message Definitions, and you need to know the data constraints for a particular Message Definition in order to correctly populate the message with sample data. You should also be familiar with XML messaging and working with DTDs.*

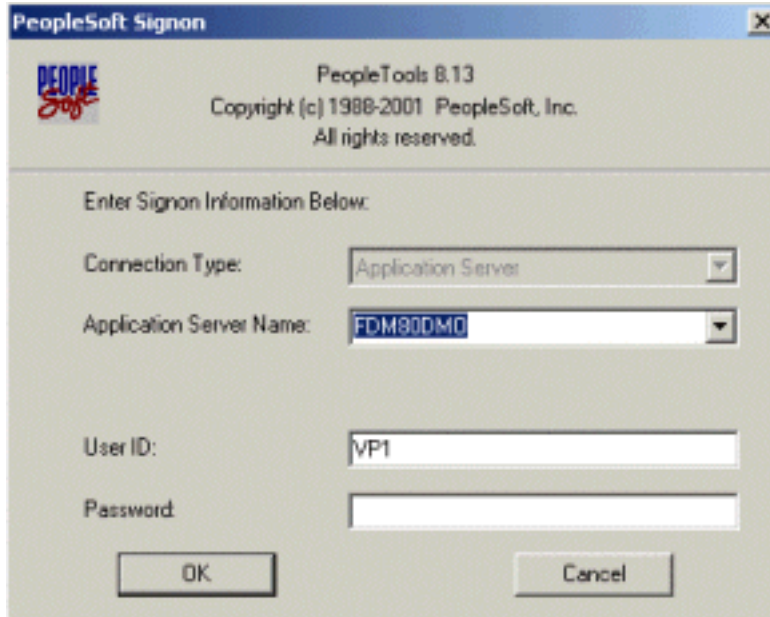
### 3.4.1 Generating and Publishing an XML Test Message

The first step is to use the PeopleSoft 8 Application Designer to generate a PeopleSoft 8 XML test message based on a particular Message Definition.

#### To Generate a PeopleSoft 8 XML Message

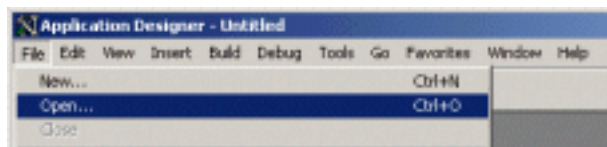
- 1 Sign onto People Tools.

**Figure 12** People Tools Signon



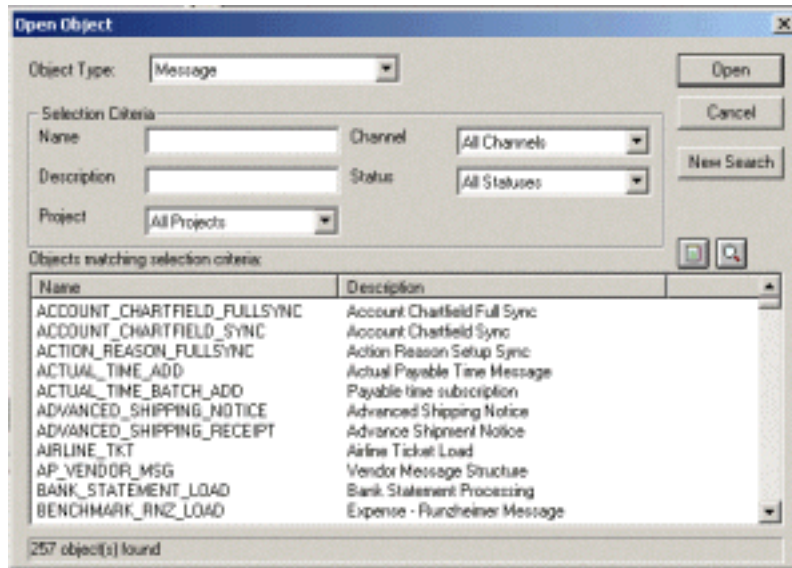
- 2 Log into the Application Designer

**Figure 13** File Menu - Open



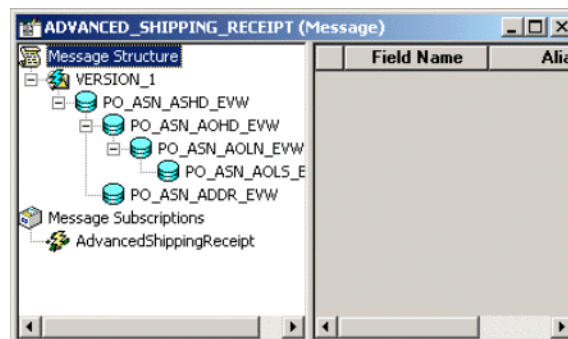
- 3 In the Application Designer, select **File** from the Menu and **Open** from the drop-down menu. The Open Object window appears (see Figure 14).

**Figure 14** Open Object Window - Object Type Message

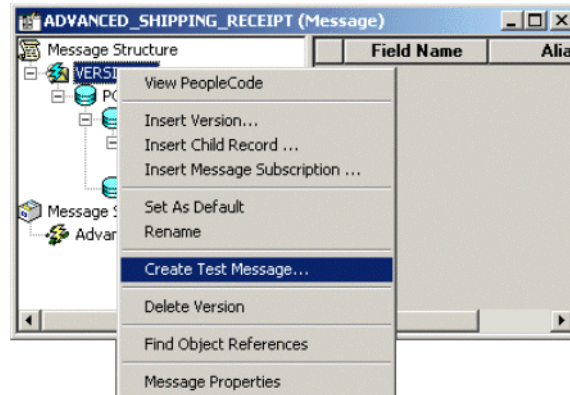


- 4 Select **Message** for the Object Type, and then click **Open**. A list of all available Message Definitions on the system appears in the bottom pane.
- 5 Find the desired Message Definition and double-click the selection, for example: **ADVANCED\_SHIPPING\_RECEIPT**. The Message window appears with **Message Structure** highlighted (see Figure 15).

**Figure 15** ADVANCED\_SHIPPING\_RECEIPT Details

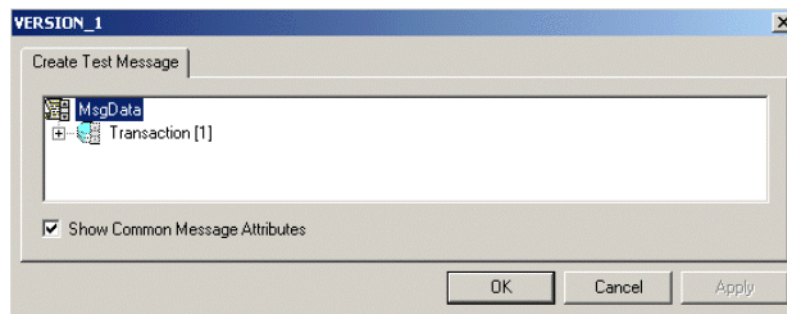


**Figure 16** Message Window and Menu



- 6 Highlight (left-click) the **Version\_1** entry within the Message window (partially hidden by overlapping menu).
- 7 Right-click the **Version\_1** entry to invoke the pop-up menu.
- 8 Select **Create Test Message** from the menu. The **Version\_1** window appears showing the records contained in the message **ADVANCED\_SHIPPING\_RECEIPT**. (see Figure 17).

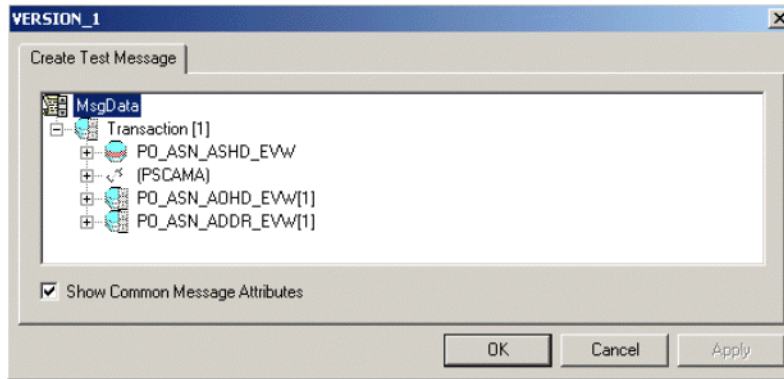
**Figure 17** Version 1 - Create Test Message (1)



- 9 Expand the **Transaction** record by clicking on the (+) symbol. This reveals all sub-records within the transaction record (see Figure 18).

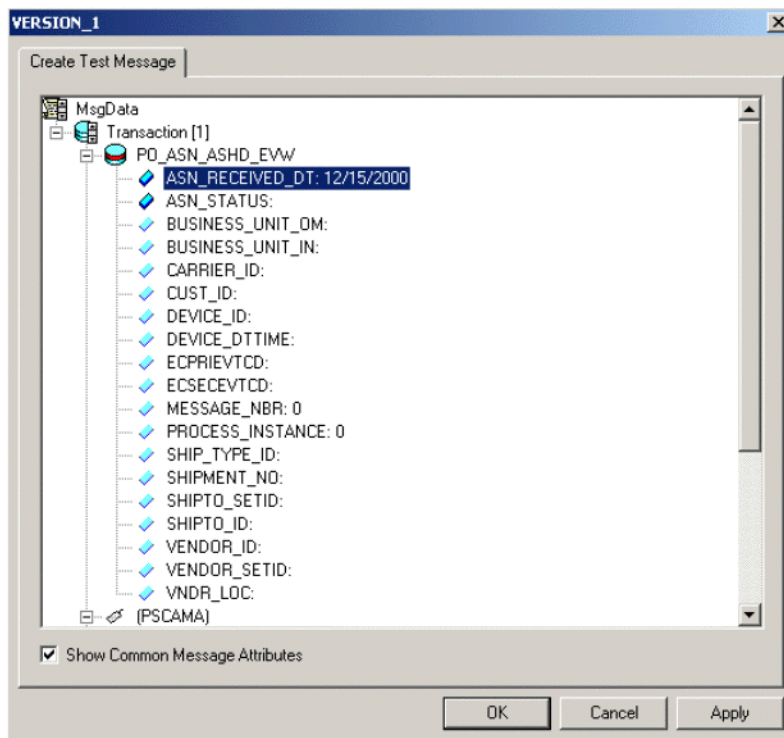


Figure 18 Version 1 - Create Test Message (2)



Records can nest to more than one level. Any record preceded by a (+) sign can be opened to verify the contents (see Figure 19).

Figure 19 Version 1 - Create Test Message (3)

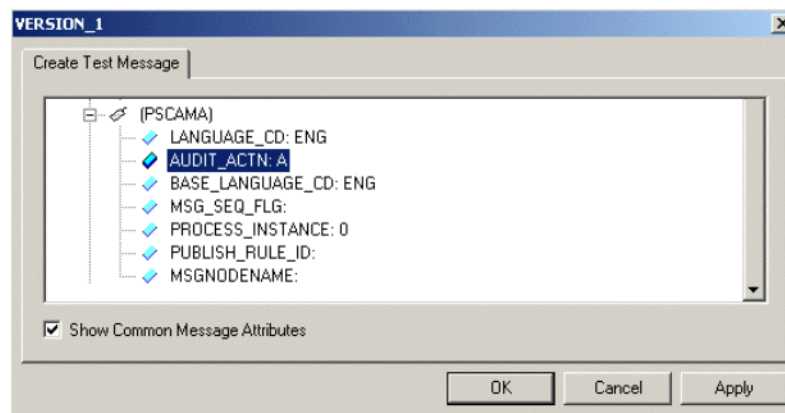


For purposes of this example, only the fields **ASN\_RECEIVED\_DT: 12/15/2000** and **ASN\_STATUS:** have data contained within them.

**Note:** You need to know the data constraints and types for each field before proceeding (the following information is specific to PeopleSoft 8).

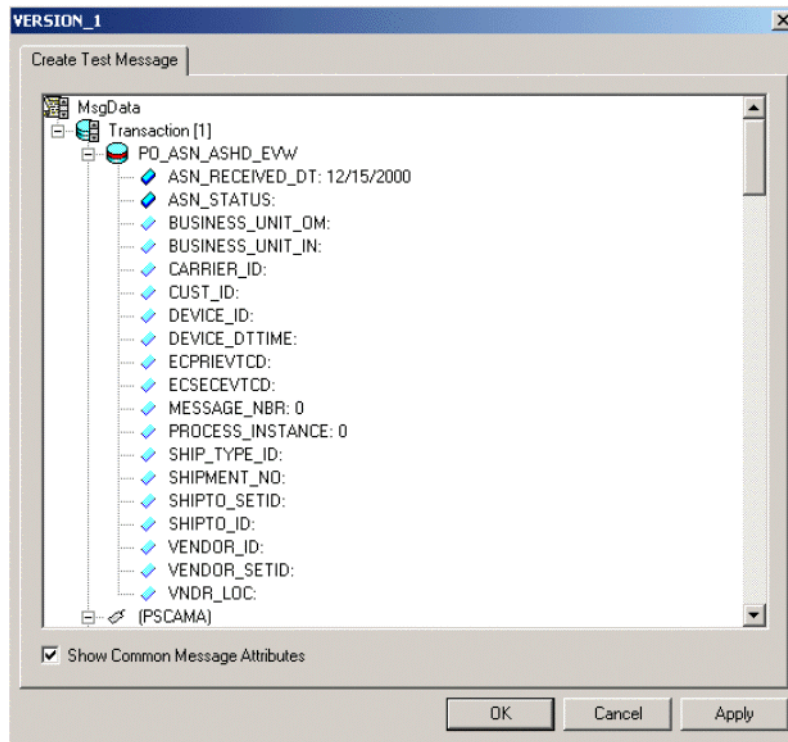
- If there are no constraints requiring you to populate all fields in a record, then generate a well-formed XML message by populating only one field in each record and sub-record.
  - If there are constraints, then all fields in each record and sub-record must be populated.
  - For most (but not necessarily all) Message Definitions, only one field is required to be populated with data. Also, some have values by default.
- 10 Enter data for the **PSCAMA** records (see Figure 20) as follows:
- A Double-click on a specific field. If the field displays empty, it is available for data input.
  - B Add the sample data.
  - C Continue populating all other records and sub-records.

**Figure 20** Version 1 - Create Test Message (4)



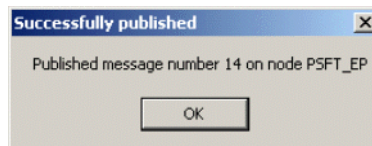
- 11 Continue entering data until all other required records and sub-records are populated, using the same method as above.
- 12 Once all records and sub-records of the message have been populated with data, click **Apply** to have the updates published to the **PSFT\_EP** Message Node. Now the message can be viewed (see Figure 21).

**Figure 21** Version 1 - Create Test Message (5)



- 13 A pop-up dialog box indicates successful publication (see Figure 22).

**Figure 22** Success Dialog Box



- 14 Click OK to dismiss the dialog box.

### 3.4.2 Extracting and Viewing the XML Test Message

The XML test message that you generated and published in the prior section can now be viewed by using a supported Web browser.

*Note:* See PeopleSoft PeopleBooks for more information on using the PeopleSoft 8 Application.

#### To View the XML Message

- 1 Within a supported Web browser, log onto the PeopleSoft 8 Application.

**Figure 23** PeopleSoft 8 Application Initial Page



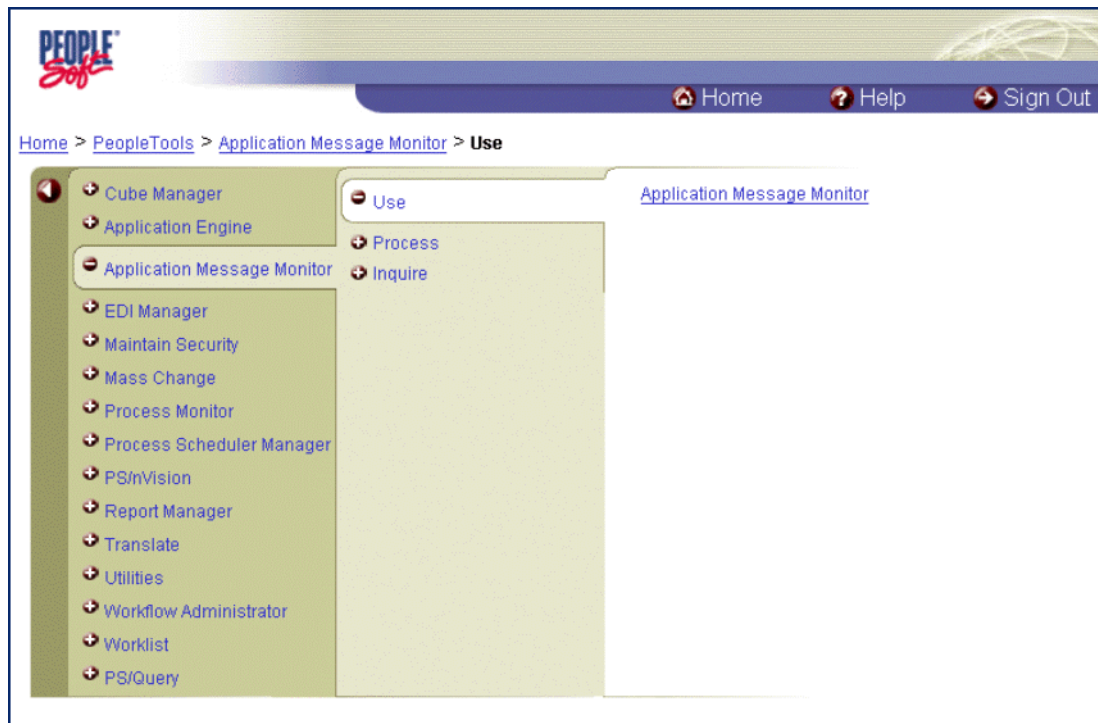
- 2 Once in PeopleSoft 8, scroll down to PeopleTools (see Figure 24).

**Figure 24** PeopleSoft 8 Application Contents Page



3 Click on **PeopleTools** to open the PeopleTools application.

**Figure 25** PeopleTools Directory Tree

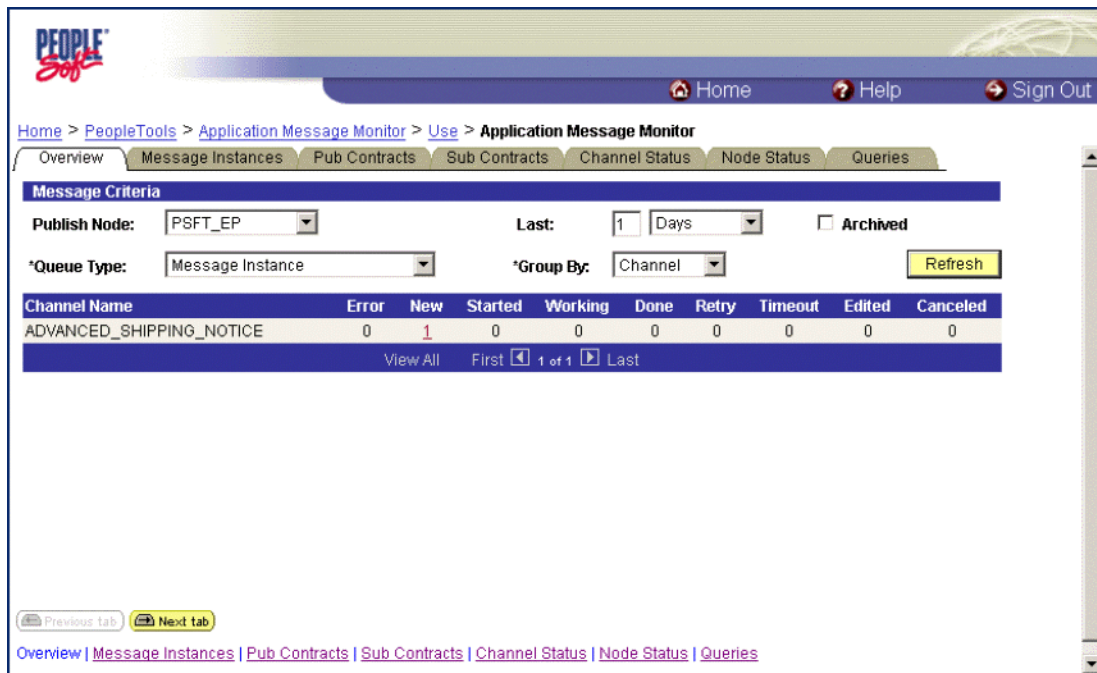


- 4 Follow the directory path

Application Message Monitor > Use > Application Message Monitor

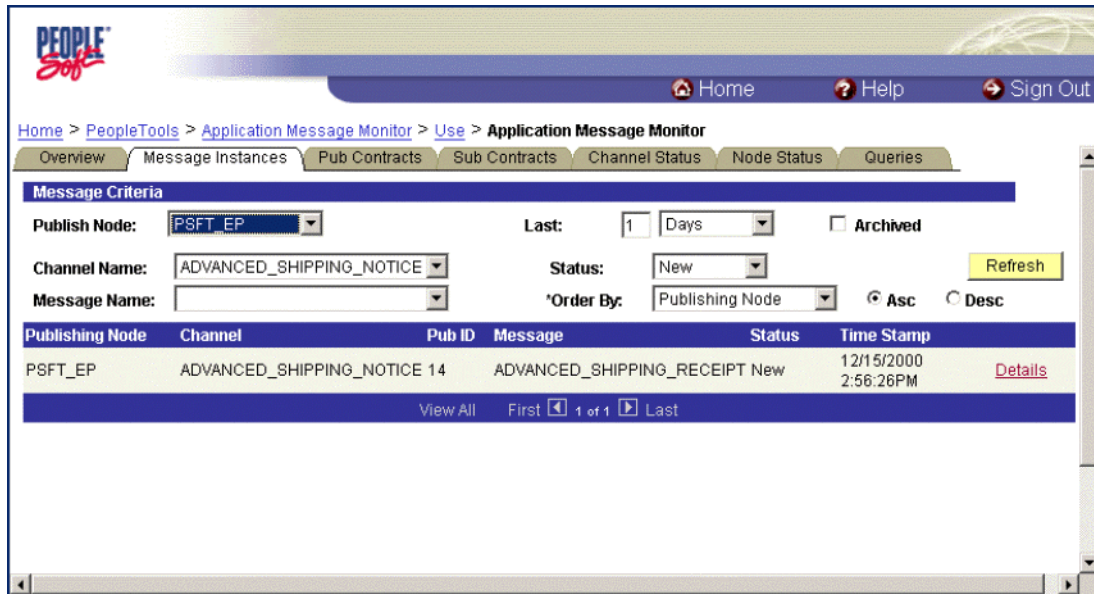
and click the hyperlink. The Application Message Monitor page opens to the Overview tab.

**Figure 26** Application Message Monitor - Overview Tab



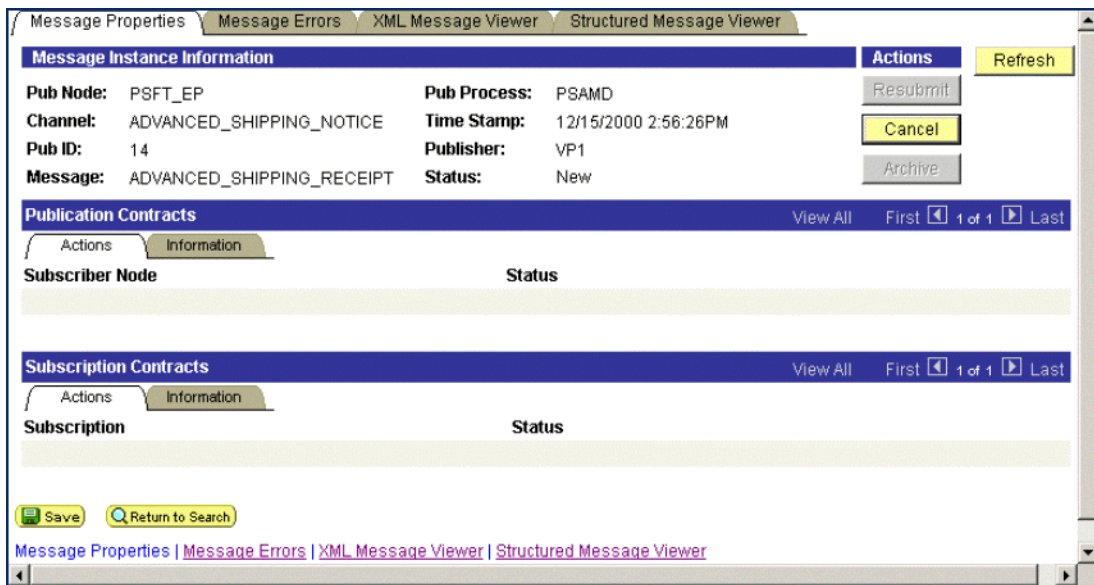
- 5 On the Overview tab, retrieve the list of published messages for the PSFT\_EP Message Node.
  - A Within the Publish Node box, select PSFT\_EP.
  - B Click the Refresh button, and the number of messages published for the selected grouping using the Create Test Message tool is indicated (in Figure 26, the Channel grouping was selected).
  - C Click the link indicated by the number of messages in the New, Done, or Working column (in Figure 26, the number 1 in the New column was selected). The Message Instances tab appears, showing a summary of the published messages (see Figure 27).

**Figure 27** Application Message Monitor - Message Instances Tab



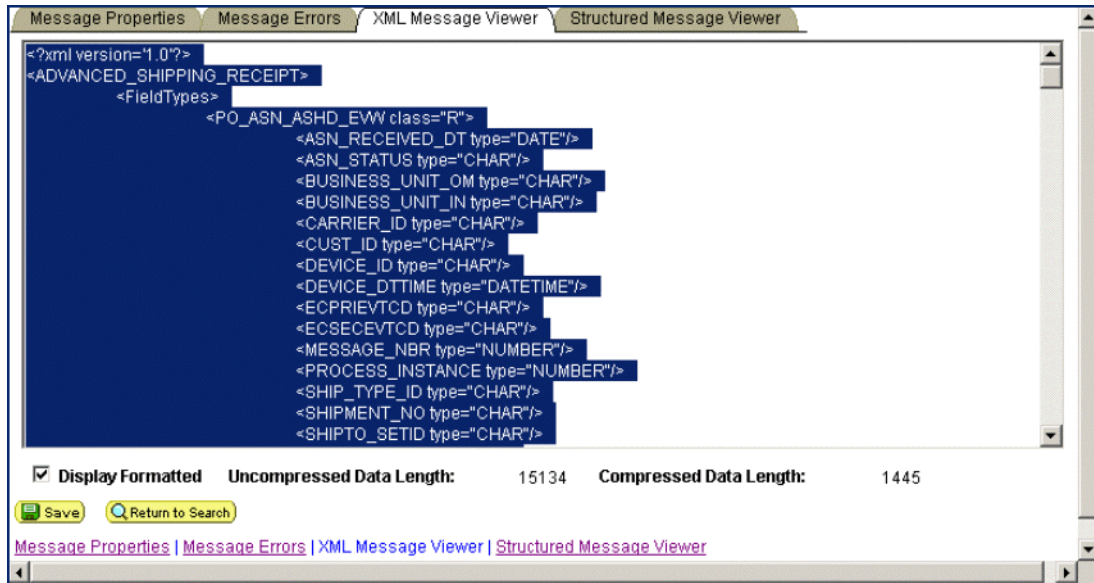
- 6 Click on the **Details** link (far right side, beneath **Refresh** button) to view properties of the XML message that was published (see Figure 28).

**Figure 28** Message Properties Tab



- 7 Click the **XML Message Viewer** tab to review the message itself.

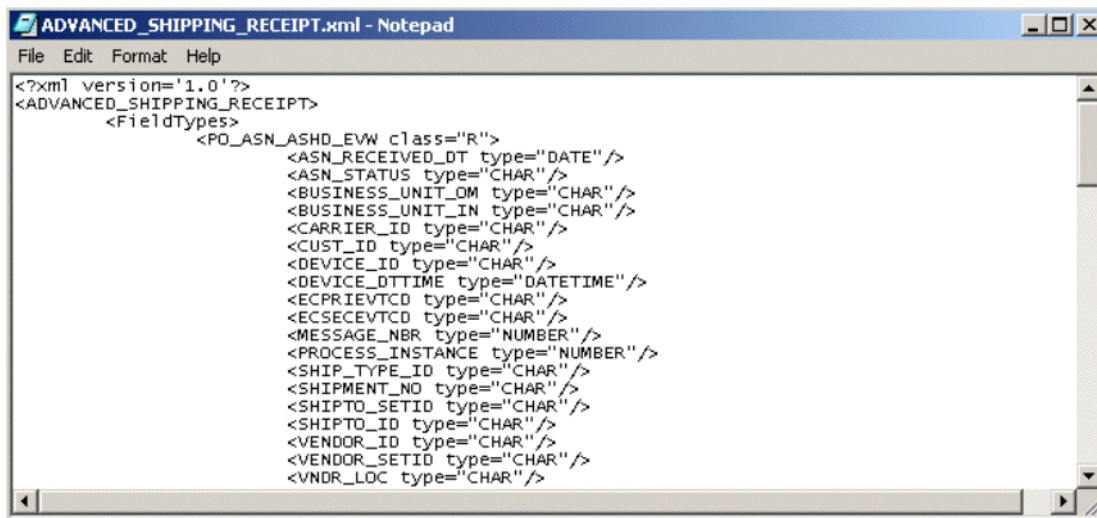
Figure 29 XML Message Viewer Tab



- 8 Save the message as an XML file.
  - A Select the entire XML message.
  - B Copy it to the clipboard.
  - C Paste the XML message into a text editor such as **Notepad** (Windows) and save it, with a **.xml** extension, to a temporary location (see Figure 30).

**Note:** Use the same naming convention used to name the Message Definition. This example shows that the XML Message **ADVANCED\_SHIPPING\_RECEIPT** was saved.

Figure 30 ADVANCED\_SHIPPING\_RECEIPT.xml





### 3.4.3 Generating a DTD for the XML File

The structure of the XML message now must be described in a Document Type Definition (DTD), from which a SeeBeyond Event Type Definition (ETD) is subsequently generated. PeopleSoft does not provide a DTD generation facility, but third-party utilities are available to accomplish this task.

For example, a free, online DTD Generator utility is available at the following URL:

<http://www.hitsw.com/Xmltools/>

This utility is shown only to illustrate the general procedure of DTD generation for the purposes of this User's Guide (see Figure 31 and Figure 32). You should always examine the results of the DTD generation process before generating the ETD, since some editing may be necessary to provide the correct XML format.

*Note:* SeeBeyond has no connection with—and does not support—this product.

**Figure 31** Example DTD Generator (1)

**XML Tools and Utilities**

Supply a file name and click the "Generate" button to display output in a browser page. Save the output to a file on your system.

Please ensure that any file you supply below does not contain references to other local files such as DTDs, external entities or XML schemas.

---

**DTD to XML Schema**

DTD File:

---

**XML Document to XML Schema**

XML Document:

---

**XML Document to DTD**

XML Document:

These conversion tools are based on work by Paul Tchistopolskii ([www.pault.com](http://www.pault.com)) and use the SAXON [DTDGenerator](#) developed by Michael Kay.

- 1 In the **XML Document to DTD** section, click the **Browse** button to open a navigator window.
- 2 Locate the .xml file where you saved the XML Message, in this example,  
`c:\temp\ADVANCED_SHIPPING_RECEIPT.xml`
- 3 Click **Open**, and the DTD Generator page reappears with the path and file displayed in the **XML Document** box (see Figure 32).

Figure 32 Example DTD Generator (2)

## XML Tools and Utilities

Supply a file name and click the "Generate" button to display output in a browser page. Save the output to a file on your system.

Please ensure that any file you supply below does not contain references to other local files such as DTDs, external entities or XML schemas.

---

### DTD to XML Schema

DTD File:

---

### XML Document to XML Schema

XML Document:

---

### XML Document to DTD

XML Document:

These conversion tools are based on work by Paul Tchistopolskii ([www.pault.com](http://www.pault.com)) and use the SAXON [DTDGenerator](#) developed by Michael Kay.

- 4 Click the **Generate DTD** button to generate the DTD.
- 5 The DTD appears as shown in Figure 33.

**Figure 33** Resulting DTD

```

Processing: C:\temp\ADVANCED_SHIPPING_RECEIPT.xml
<!ELEMENT ADDRESS1 EMPTY >
<!ATTLIST ADDRESS1 type NMTOKEN #IMPLIED >

<!ELEMENT ADDRESS2 EMPTY >
<!ATTLIST ADDRESS2 type NMTOKEN #IMPLIED >

<!ELEMENT ADDRESS3 EMPTY >
<!ATTLIST ADDRESS3 type NMTOKEN #IMPLIED >

<!ELEMENT ADDRESS4 EMPTY >
<!ATTLIST ADDRESS4 type NMTOKEN #IMPLIED >

<!ELEMENT ADVANCED_SHIPPING_RECEIPT ( FieldTypes, MsgData ) >

<!ELEMENT ASN_DEFAULT_KEY EMPTY >
<!ATTLIST ASN_DEFAULT_KEY type NMTOKEN #IMPLIED >

<!ELEMENT ASN_DESCR EMPTY >
<!ATTLIST ASN_DESCR type NMTOKEN #IMPLIED >

```

- 6 Save the message as an DTD file.
  - A Select only the DTD-related information (usually all information except the first line), as shown in Figure 33.
  - B Copy it to the clipboard.
  - C Paste the text into a text editor such as Notepad (Windows) and save it, with a .dtd extension, to a temporary location (see Figure 34).

**Note:** Use the same naming convention used to name the Message Definition (in the example, ADVANCED\_SHIPPING\_RECEIPT).

**Figure 34** DTD File

```

ADVANCED_SHIPPING_RECEIPT.dtd - Notepad
File Edit Format Help
<!ELEMENT ADDRESS1 EMPTY >
<!ATTLIST ADDRESS1 type NMTOKEN #IMPLIED >

<!ELEMENT ADDRESS2 EMPTY >
<!ATTLIST ADDRESS2 type NMTOKEN #IMPLIED >

<!ELEMENT ADDRESS3 EMPTY >
<!ATTLIST ADDRESS3 type NMTOKEN #IMPLIED >

<!ELEMENT ADDRESS4 EMPTY >
<!ATTLIST ADDRESS4 type NMTOKEN #IMPLIED >

<!ELEMENT ADVANCED_SHIPPING_RECEIPT ( FieldTypes, MsgData ) >

<!ELEMENT ASN_DEFAULT_KEY EMPTY >
<!ATTLIST ASN_DEFAULT_KEY type NMTOKEN #IMPLIED >

<!ELEMENT ASN_DESCR EMPTY >
<!ATTLIST ASN_DESCR type NMTOKEN #IMPLIED >

<!ELEMENT ASN_INV_ITEM_ID EMPTY >
<!ATTLIST ASN_INV_ITEM_ID type NMTOKEN #IMPLIED >

<!ELEMENT ASN_LOT_NBR ( #PCDATA ) >
<!ATTLIST ASN_LOT_NBR type NMTOKEN #IMPLIED >

<!ELEMENT ASN_RECEIVED_DT ( #PCDATA ) >
<!ATTLIST ASN_RECEIVED_DT type NMTOKEN #IMPLIED >

<!ELEMENT ASN_SCHED_NBR ( #PCDATA ) >
<!ATTLIST ASN_SCHED_NBR type NMTOKEN #IMPLIED >

<!ELEMENT ASN_SEQ_NBR ( #PCDATA ) >
<!ATTLIST ASN_SEQ_NBR type NMTOKEN #IMPLIED >

```

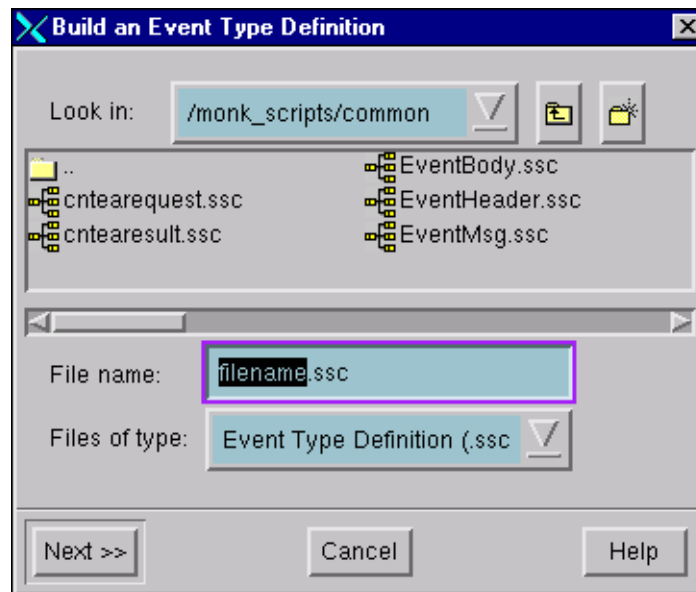
### 3.4.4 Generating an ETD from the DTD

The SeeBeyond XML Converter (toolkit) is used to produce an ETD (Event Type Definition) from the DTD that was generated. See the *XML Toolkit* documentation for details on installation and usage of the converter.

To create an ETD using the XML Converter from the GUI:

- 1 Launch the Monk ETD Editor.
- 2 On the ETD Editor's Toolbar, click **Build**. The Build an Event Type Definition dialog box appears.

**Figure 35** Build an Event Type Definition

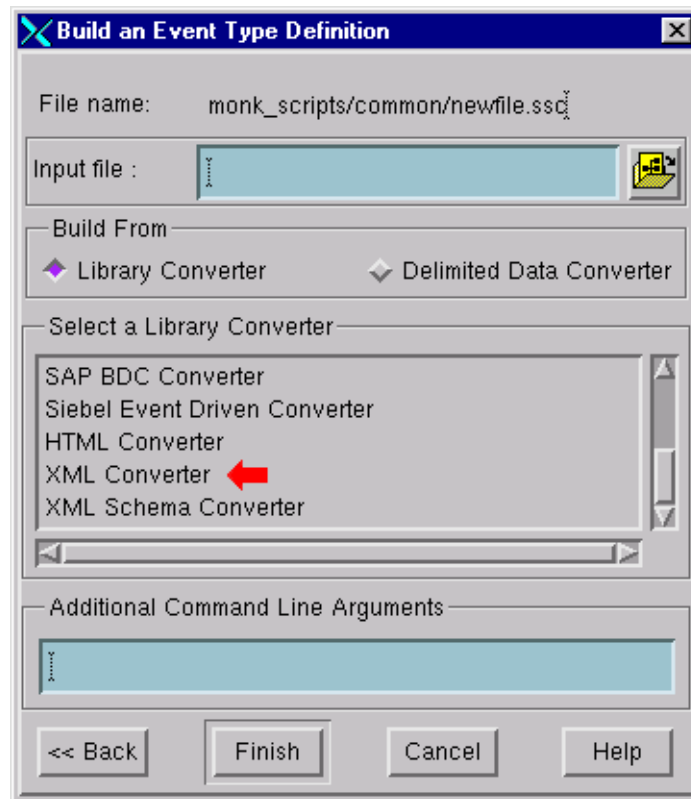


- 3 In the File name field, type the name of the ETD file you wish to build.

**Note:** The Editor automatically supplies the *.ssc* extension.

- 4 Click **Next**. A new dialog box appears.

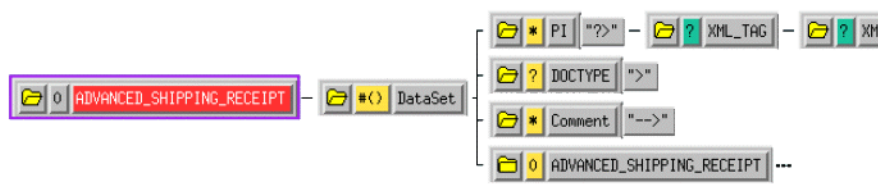
**Figure 36** Build an Event Type Definition - XML Converter



- 5 Insert the name of the DTD file you previously created.
- 6 Under Build From, select **Library Converter**.
- 7 Under Select a Library Converter, select **XML Converter**.
- 8 Click **Finish**, and the Build tool creates the ETD.

After converting the DTD to an ETD, return to the e\*Gate Schema Designer to verify the process (see Figure 37).


**Figure 37** ETD Structure



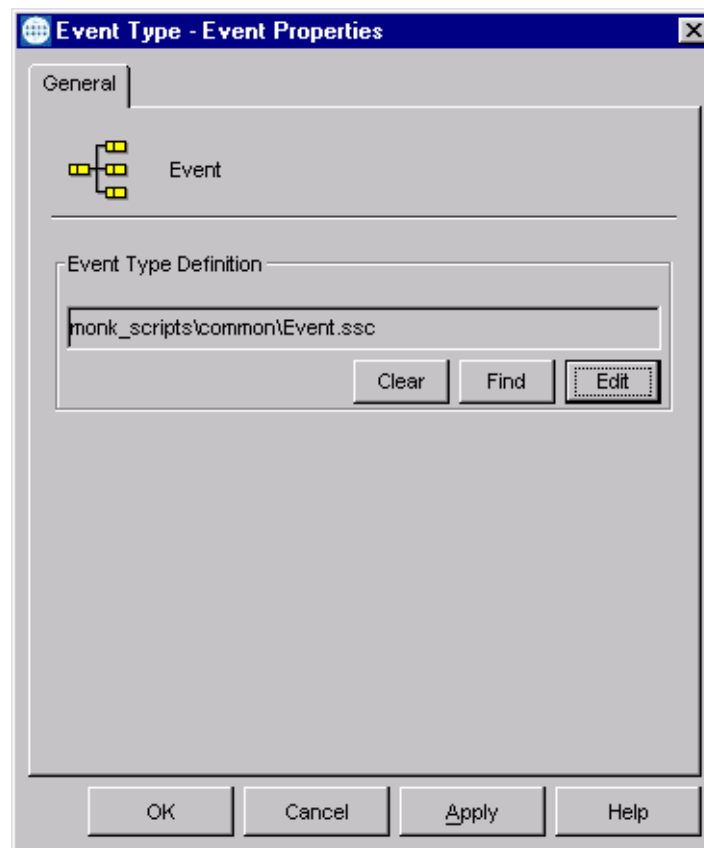
## 3.5 Assigning ETDs to Event Types

After you have created the e\*Gate system's ETD files, you can assign them to Event Types you have already created.

### To assign ETDs to Event Types

- 1 In the Schema Designer window, select the **Event Types** folder in the Navigator/Components pane.
- 2 In the Editor pane, select one of the Event Types you created.
- 3 Right-click on the Event Type and select **Properties** (or click  in the toolbar). The Event Type Properties dialog box appears. See Figure 38.

**Figure 38** Event Type Properties Dialog Box



- 4 Under Event Type Definition, click **Find**, and the Event Type Definition Selection dialog box appears (it is similar to the Windows Open dialog box).
- 5 Open the `monk_scripts\common` folder, then select the desired file name (.ssc).
- 6 Click **Select**. The file populates the Event Type Definition field.

- 7 To save any work in the properties dialog box, click **Apply** to enter it into the system.
- 8 When finished assigning ETDs to Event Types, click **OK** to close the properties dialog box and apply all the properties.

Each Event Type is associated with the specified Event Type Definition.

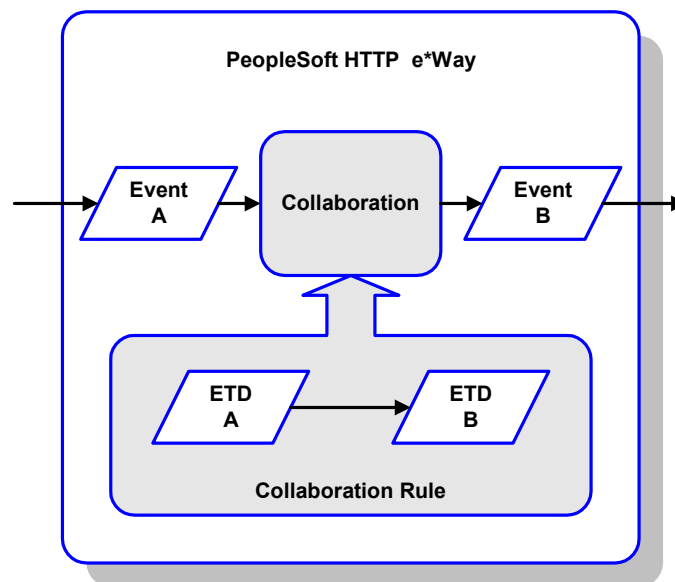
---

## 3.6 Defining Collaborations

After you have created the required Event Type Definitions, you must define a Collaboration to transform the incoming Event into the desired outgoing Event.

Collaborations are e\*Way components that receive and process Event Types, then forward the output to other e\*Gate components. Collaborations consist of the Subscriber, which “listens” for Events of a known type or from a given source, and the Publisher, which distributes the transformed Event to a specified recipient. The same Collaboration cannot be assigned to more than one e\*Gate component.

**Figure 39** Collaborations



The Collaboration is driven by a Collaboration Rule script, which defines the relationship between the incoming and outgoing ETDs. You can use an existing Collaboration Rule script, or use the Monk programming language to write a new Collaboration Rule script. Once you have written and successfully tested a script, you can then add it to the system’s run-time operation.

Collaborations are defined using the e\*Gate Monk Collaboration Rules Editor. See the *e\*Gate Integrator User’s Guide* for instructions on using this Editor. The file extension for Monk Collaboration Rules is **.tsc**.

---

## 3.7 Creating Intelligent Queues

The final step is to create and associate an IQ for the PeopleSoft HTTP e\*Way. IQs manage the exchange of information between components within the e\*Gate system, providing non-volatile storage for data as it passes from one component to another. IQs use IQ Services to transport data. IQ Services provide the mechanism for moving Events between IQs, handling the low-level implementation of data exchange (such as system calls to initialize or reorganize a database). See the *e\*Gate Integrator User's Guide* for complete information on queuing options and procedures for creating IQs.



## 3.8 Using the e\*Way

As explained in the Introduction, full integration with PeopleSoft 8 requires the use of the Java-enabled conjunction HTTP e\*Way, the e\*Gate API Kit, and SeeBeyond's customized PeopleSoft 8 MUX subscription handler classes. The PeopleSoft HTTP e\*Way is used to publish data to PeopleSoft 8 by utilizing the HTTP e\*Way. The API Kit (MUX e\*Way) is used in conjunction with the subscription handler classes written by SeeBeyond in order to receive data from PeopleSoft 8 and publish the data to e\*Gate.

### 3.8.1 Publishing to PeopleSoft

Publishing data to PeopleSoft 8 using the PeopleSoft HTTP e\*Way consists of four primary steps:

- 1 Compress the "business data" and base-64 encode it (optional).
- 2 Wrap the "business data" with PeopleSoft 8 XML "container" information.
- 3 Publish the wrapped XML data by using **http-post**.
- 4 Obtain the PeopleSoft 8 XML response by using **http-get-result-data** and check for a successful publish.

The PeopleSoft HTTP e\*Way does *not* contain any logic to create PeopleSoft 8 XML messages, nor does it contain any logic to map external data to PeopleSoft 8 XML messages. The e\*Way merely accepts and publishes the raw PeopleSoft 8 XML message, irrespective of what the data may look like.

### XML Messages

The following is a description of the XML message, as described in the PeopleSoft 8 PeopleTools documentation, that is created by the e\*Way prior to publishing to PeopleSoft 8:

```
<?xml version="1.0"?>
<request version="peopletools-version">
<to node="destination-node-name"/>
<from node="source-node-name" password="source-node-group-password"/>
<operations namespace="PublishSubscribe"
interface="PublishSubscribeSystem">
<invoke member="Publish">
<variable type="object" interface="Publication">
<publication>
<publishingnode>source-node-name</publishingnode>
<channel>channel-name</channel>
<publicationid>publication-id</publicationid>
<subchannel>subchannel</subchannel>
<subject>message-name</subject>
<subjectdetail>message-detail</subjectdetail>
<originatingnode>originating-node-name</originatingnode>
<publisher>publishing-operator-id</publisher>
<publicationprocess>publication-process</publicationprocess>
<publishtimestamp>publish-timestamp</publishtimestamp>
<status>publication-status</status>
<defaultdataversion>default-message-version-name</defaultdataversion>
<dataversions>
```

```

<publicationdataversion>
<version>VERSION_1</version>
<data length="identity-length" encoding="base64(deflate)"
encodedlength="base64-encoded-length(deflated-length)">deflated,
base-64 encoded XML data</data>
</publicationdataversion>
</dataversions>
</publication>
</variable>
</invoke>
</operations>
</request>

```

Compressed (“deflated”), base-64 encoded XML data is the data that is to be published. The e\*Way can be configured to compress and/or base-64 encode the data. The remainder of the XML data is PeopleSoft 8 “container” information, and most of these parameters are configurable in the e\*Way. By default, the e\*Way compresses and base-64 encodes the XML data. See PeopleSoft 8 PeopleTools documentation for detailed information on each of the elements of the XML message.

In response to a published message, PeopleSoft 8 Application Messaging Gateway returns a reply XML message with the following format:

```

<?xml version="1.0" ?>
<reply>
<operations namespace="PublishSubscribe"
interface="PublishSubscribeSystem">
<invoke opnum="1" member="Publish">
<return type="number">0</return>
<variable type="object" interface="Publication">
<publication>
<publishingnode>publisher_node</publishingnode>
<channel>published_channel</channel>
<publicationid>published_id</publicationid>
<publishtimestamp>date_time_of_publish</publishtimestamp>
</publication>
</variable>
</invoke>
</operations>
</reply>

```

The e\*Way checks the value of <return>, which can be:

- **0**, which means the message was delivered to PeopleSoft 8 successfully
- **6**, which means the message was already delivered successfully
- **8**, which means the message was *not* delivered successfully—in which case the e\*Way marks the message for re-sending.

## Compressing the XML Message

The PeopleSoft subscribing node can accept XML data in either a compressed and encoded, or an uncompressed and unencoded format.

- The compression algorithm that PeopleSoft accepts is base64(deflate). See *Uncompressing Messages* for more information.

- When compressed:
  - ♦ The encoding attribute of the <data> element contains information about how many bytes each of the compressed routines produced.
  - ♦ The length attribute of the <data> element is the number of Unicode characters when uncompressed.

Because PeopleTools 8 uses Unicode, two bytes represent a character that was previously represented by a single byte. One way to derive the Unicode byte length is to multiply the character length of the inflated contents of the <data> tags by 2 to get the correct length that PeopleTools can use to inflate to Unicode.

For example, the following code describes data that contains 4126 Unicode characters when uncompressed. When deflated, the size of the data is 532 bytes and when base64 encoded, becomes 712 bytes in size:

```
<data encoding="base64(deflate)" encodedlength="712 (532)"  
      length="4126">
```

If you do not deflate and base64-encode the <data> element contents, then you must remove the <-xml version="1.0" -> processing instruction tag. Otherwise, the PeopleSoft application server fails (because XML does not support nested documents).

## 3.8.2 Subscribing to PeopleSoft

To send data from PeopleSoft 8 to e\*Gate, a servlet subscription handler called the **MuxHandler** is loaded into PeopleSoft's Application Messaging Gateway servlet. The **MuxHandler** implements the IPS Handler interface allowing it to intercept messages published from PeopleSoft 8 and directing them to e\*Gate API Kit (MUX e\*Way).

Publishing data to e\*Gate consists of four steps:

- 1 Decompress and/or base-64 decode the "business data" (optional).
- 2 Instantiate an **IPMPReqReply** object and connect to the appropriate MUX e\*Way.
- 3 Send the "business data" to the MUX e\*Way by using **sendMessage**.
- 4 Verify that **sendMessage** completes successfully and sends a positive acknowledgment to PeopleSoft 8.

The **MuxHandler** subscription handler has eight configuration parameters:

- **Node Name** (Message Node that was created with the PeopleSoft 8 Application Designer, and is associated with the configured MUX Handler)
- **MUX Host**
- **MUX Port**
- **MUX Expire** (default is 10 sec.)
- **MUX Timeout** (default is 10000 msec.)
- **Uncompress** (default is YES)
- **Base64 Decode** (default is YES)
- **Log File** (default is C:\\yyyyymmddhhmmss on Windows and /tmp/yyyyymmddhhmmss on UNIX).

The **Uncompress** and **Base64 Decode** configurations can be used to decompress and/or decode the XML data before publishing to the Multiplexer e\*Way. By default these are set to **YES**. Publishing encoded and compressed data is highly recommended since it reduces both the network bandwidth required and the processing load for the Multiplexer e\*Way.

**Note:** **Base64 Decode** must be selected if **Uncompress** is selected.

If **Base64 Decode** and **Uncompress** are *not* selected, then the XML message, along with its "container" information, is sent to the MUX e\*Way as-is. In this case, you must configure the MUX e\*Way to decode and decompress the data when the data is received. See the *e\*Gate API Kit User's Guide* for details on the MUX configurations.

The Publication ID and the Subject (Message Definition) of every message published is written to the log file as well as any errors that may have occurred. Note that the log file should be cleared out periodically to prevent it from growing too large. By default, the log file is named using the creation date and time of the subscription handler. You have the option of renaming the log file to something more meaningful to you. See [Installing the MUX Handler Classes](#) on page 22 for more details.

# PeopleSoft 8 Setup

This chapter describes procedures for configuring the PeopleSoft 8 Application to interact properly with the PeopleSoft HTTP e\*Way.

---

## 4.1 Overview

To publish data to, or subscribe to data from, PeopleSoft 8, the following must be created and configured within the PeopleSoft 8 environment:

- Message Nodes
- Messages
- Message Channels
- Subscription Handlers

For purposes of this User's Guide, *publishing* refers to sending outbound messages to PeopleSoft 8 and *subscribing* refers to receiving inbound messages from PeopleSoft 8.

PeopleSoft 8 comes with a set of predefined Message Definitions and Message Channels that can be used as-is. You can also create your own Message Definitions and Message Channels. You must know in advance which Message Definition(s) and which Message Channel(s) to use during the external configuring process. Please refer to the *PeopleSoft 8 EIP Catalog* for more information. Also, the PeopleSoft 8 documentation on *Adding and Configuring Subscription Handlers* contains valuable information regarding Subscription Handlers.

**Note:** You should be familiar with using the *PeopleSoft 8 Application Designer* and *Servlet Configuration Tool* for subscription handlers.

## 4.2 Configuring for Publication

To enable the PeopleSoft HTTP e\*Way to publish XML Messages to PeopleSoft 8, the following configuration steps must be performed.

[Creating PeopleSoft 8 Message Node for the e\\*Way](#) on page 54

[Activating the Message Definition for Publication](#) on page 58

[Defining Message Channel Routing Rules](#) on page 61

[Defining Routing Directions for Message Nodes](#) on page 65

[Adding the PeopleSoft 8 Subscription Handler](#) on page 67

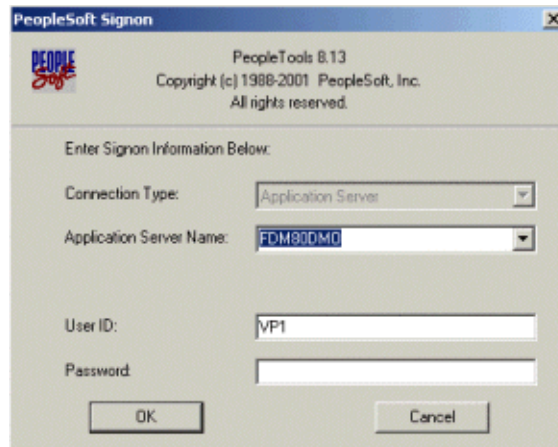
### 4.2.1 Creating PeopleSoft 8 Message Node for the e\*Way

Complete the following procedures to create a PeopleSoft 8 Message Node in order to configure the PeopleSoft HTTP e\*Way to publish XML messages.

To create the Message Node to publish XML messages

- 1 Sign on to People Tools, and start the Application Designer.

**Figure 40** PeopleSoft Signon



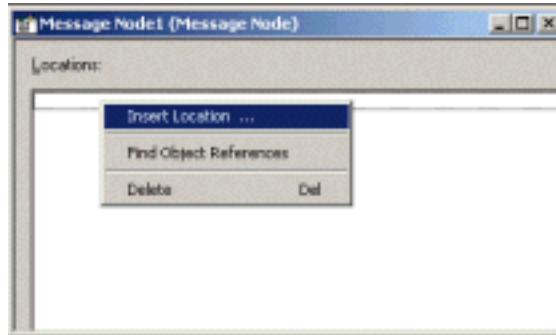
- 2 From the Application Designer File menu, select New.
- 3 In the New dialog box, select Message Node and click OK to display the Message Node dialog box for Node 1.

**Figure 41** New Pop-Up Menu - Message Node



- 4 Right-click within the **Locations** pane, and a pop-up menu appears. Select **Insert Location** from the pop-up menu to enter the PeopleSoft 8 Application Messaging Gateway (servlet) URL.

**Figure 42** Message Node Window - Insert Location



- 5 Type the following URL of the PeopleSoft 8 Gateway Servlet into the Location dialog box.

**A** For Apache:

**`http://PSFTHOST/servlets/psft.pt8.gateway.GatewayServlet`**

**B** For WebLogic:

**`http://PSFTHOST/servlets/gateway`**

**Note:** You must replace the name **PSFTHOST** in the above URL with the actual name of the host computer on which PeopleSoft 8 is installed.

**Figure 43** Location Dialog Box



- 6 Click **OK** to save the URL. The URL name you have entered then appears in the Message Node dialog box.
- 7 Save the Message Node and commit it to the PeopleSoft 8 database as follows:
  - A Select **Save As** from the **File** drop-down menu.
  - B Type the name of the Message Node you are saving into the text box. (The example above uses **STCPUBLISHER**.) This name is needed for the e\*Way configuration as the **From Node** parameter.

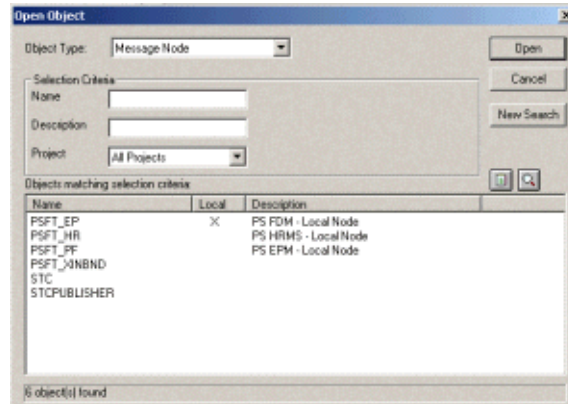
**Figure 44** Save As Dialog Box



- 8 From the Application Designer **File** menu, select **Open** to invoke the **Open Object** window.
- 9 Verify that the Message Node you created is ready for use by PeopleSoft 8.
  - A In the **Open Object** window, select **Message Node** from the **Object Type** list and click the **Open** button.
  - B A list of all Message Nodes within the system appears. The name of the newly-created message node should appear in the **Objects matching selection criteria** pane. If so, you have successfully completed creating a Message Node for SeeBeyond to publish data to PeopleSoft 8.



**Figure 45** Open Object Window - Message Node



**Note:** All Message Nodes with **PSFT** prefixes were created by the PeopleSoft 8 installation. **PSFT\_EP** is the PeopleSoft Local Node for the Financials application. It is specified as a subscriber to messages sent from the HTTP e\*Way, and a publisher of messages to the Multiplexer e\*Way.

## 4.2.2 Activating the Message Definition for Publication

As mentioned previously, PeopleSoft 8 comes with a set of predefined Message Definitions. The desired Message Definition is configurable in the e\*Way as the **Subject** parameter. The following instructions describe how to activate the Message Definition for subscription to the SeeBeyond Multiplexer e\*Way.

*Note:* For purposes of this publication the **ADVANCED\_SHIPPING\_RECEIPT** Message Definition is activated for publish / subscribe.

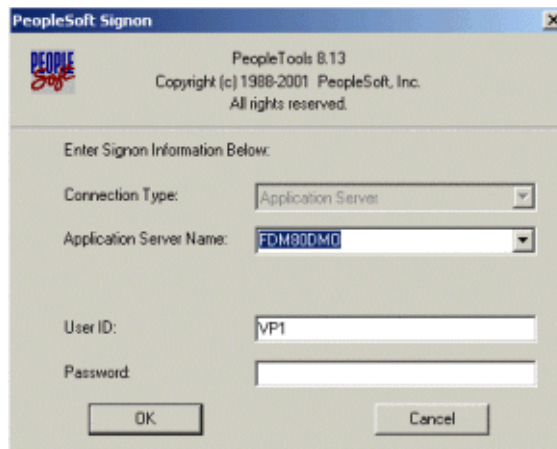
### PeopleSoft 8 Message Definitions List

Each message used for publication must be defined. This definition corresponds to the XML Message the e\*Way publishes, and contains the elements of the data to be published. However, before the e\*Way can publish any data, the Message Definition must be activated. A list of these definitions can be found within the Application Designer.

To activate the Message Definition for subscription to the e\*Way

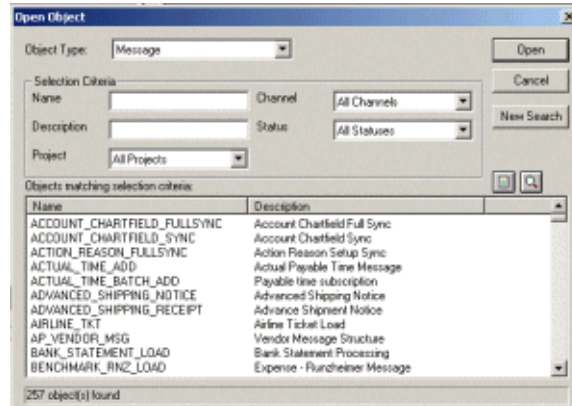
- 1 Sign on to PeopleTools Application Designer.

**Figure 46** PeopleSoft 8 Signon



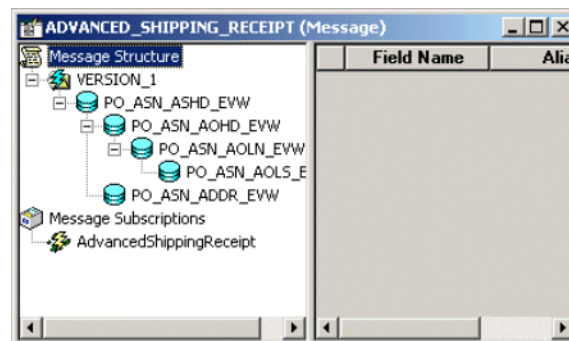
- 2 From the Application Designer **File** menu, select **Open** to invoke the **Open Object** window.
- 3 Select **Message** from the **Object Type** list, and you are presented with a list of all available PeopleSoft 8 Message Definitions.

Figure 47 Open Object Window



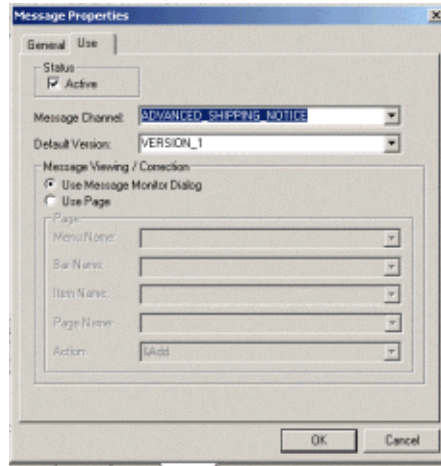
- 4 Select the Message Definition you want and double-click on that selection; for example, **ADVANCED\_SHIPPING\_RECEIPT**.
- 5 The Message window appears, displaying the complete record details of the chosen Message Structure.

Figure 48 ADVANCED\_SHIPPING\_RECEIPT Details



- 6 From the Application Designer **File** menu, select **Object Properties** to open the Message Properties dialog box.
- 7 In the Message Properties dialog box, click the **Use** tab to display the Status field.

**Figure 49** Message Properties Dialog Box - Use Tab



- 8 Check the Active button and click OK to save the settings.
- 9 From the File menu, select Save to save and commit the changes to the Message Definition. You have now activated the Message Definition for publishing or subscribing.

### 4.2.3 Defining Message Channel Routing Rules

Before proceeding with this process, you should determine which Message Channel to use. The Message Channel to use is configurable in the e\*Way as the **Channel** parameter.

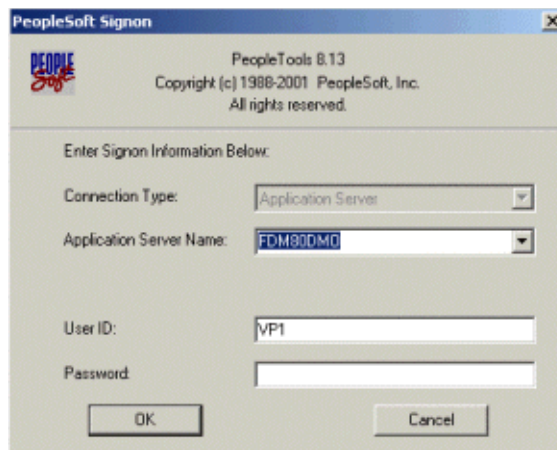
#### Configuring the Message Channel

Each Message Channel logically groups Messages together. For purposes of this documentation, **ADVANCED\_SHIPPING\_RECEIPT** Message is grouped into the **ADVANCED\_SHIPPING\_NOTICE** Message Channel.

To configure the Message Channel

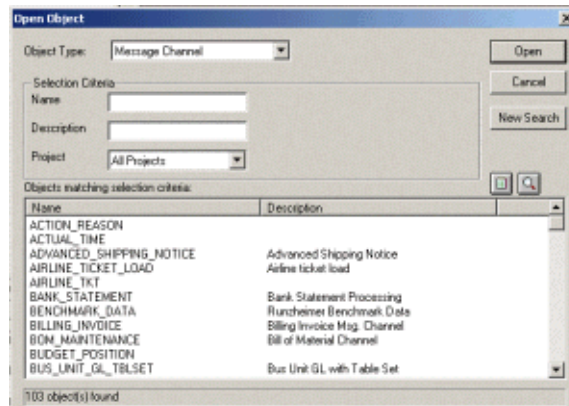
- 1 Sign on to PeopleTools Application Designer.

**Figure 50** PeopleSoft Signon



- 2 From the Application Designer **File** menu, select **Open** to invoke the **Open Object** window.

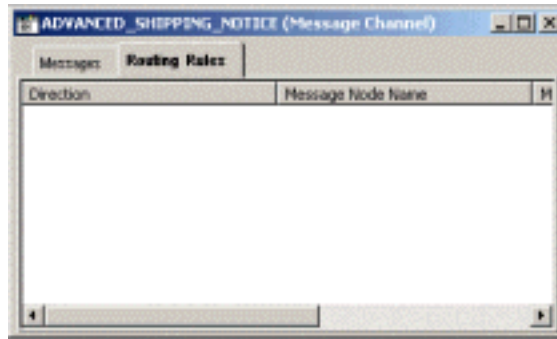
**Figure 51** Open Object Window



- 3 Select **Message Channel** from the **Object Type** list.

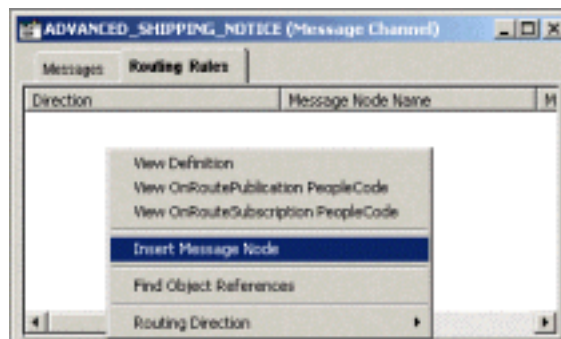
- 4 Click **Open**, and a list of all available Message Channels on the system appears.
- 5 Double-click on the name of the desired Message Channel. The Message Channel window appears for that channel.

**Figure 52** Message Channel - ADVANCED\_SHIPPING\_NOTICE (1)



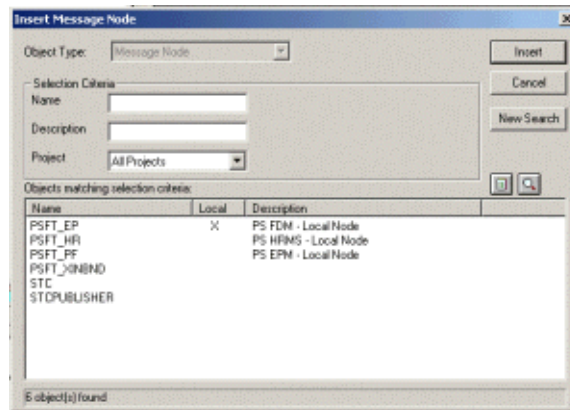
- 6 Left-click the **Routing Rules** tab, then right-click in the data pane. A pop-up menu appears with the following options:
  - ◆ View Definition
  - ◆ View **OnRoutePublication** PeopleCode
  - ◆ View **OnRouteSubscription** PeopleCode
  - ◆ Insert Message Node
  - ◆ Find Object References
  - ◆ Routing Direction

**Figure 53** Message Channel - ADVANCED\_SHIPPING\_NOTICE (2)



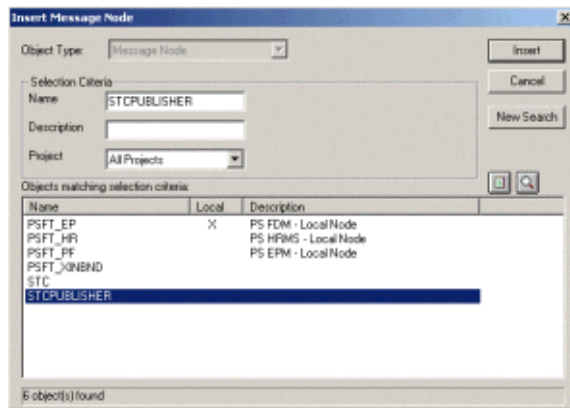
- 7 Left-click to select **Insert Message Node**. The Insert Message Node window appears, displaying the available Message Nodes.

**Figure 54** Insert Message Node Window



- 8 Click on **PSFT\_EP**, then click the **Insert** button. This is inserted the message into the Routing Rules Table.
- 9 Click on **STCPUBLISHER**, then click **Insert**.

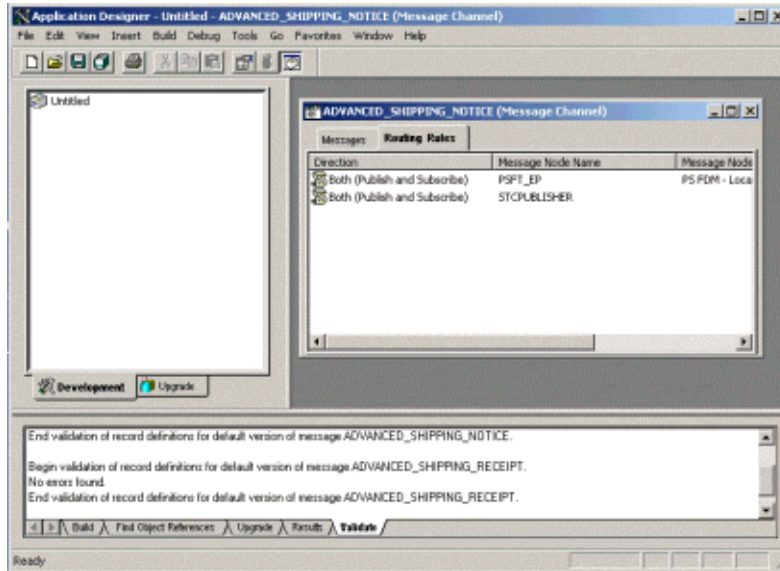
**Figure 55** Insert Message Node Window - STCPUBLISHER



- 10 Click **Cancel** to exit the Insert Message Node window.

The Message Nodes now are defined on the Routing Rules tab of the Message Channel window.

Figure 56 Message Channel - ADVANCED\_SHIPPING\_NOTICE (3)





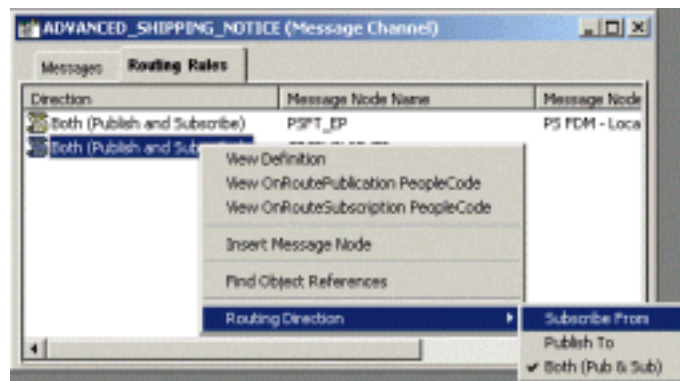
## 4.2.4 Defining Routing Directions for Message Nodes

Routing Directions provide you with the ability to assign destinations, either Publish To or Subscribe From, a Message Node. This section describes the procedure for defining the Routing Directions for the SeeBeyond Message Node, **Subscribe From**.

To define the Routing Directions

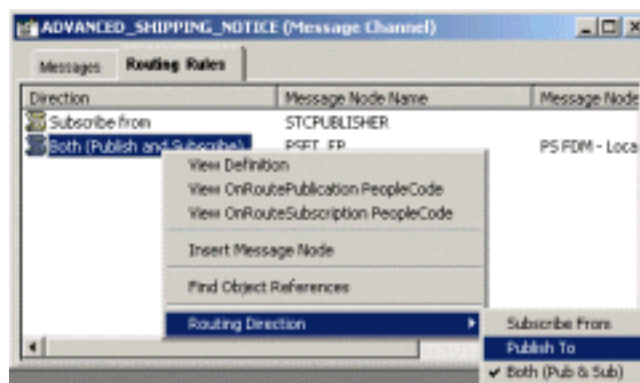
- 1 From the **Direction** column of the **Routing Rules** tab, right-click on **Both (Publish and Subscribe)** for the Message Node Name **STCPUBLISHER**. A pop-up menu appears with several options.

Figure 57 Subscribe From Selection



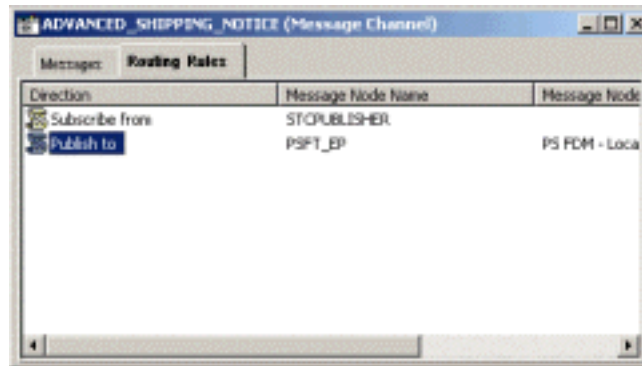
- 2 On the menu, left-click **Routing Direction**, then left-click **Subscribe From** on the secondary menu.
- 3 Within the Message Channel window, right-click on **Both (Publish and Subscribe)** for the Message Node Name **PSFT\_EP**. Again, the pop-up menu appears.

Figure 58 Publish To Selection



- 4 Left-click **Routing Direction**, then left-click **Publish To** on the secondary menu.

**Figure 59** Routing Rules > Verify



- 5 Under the Routing Rules tab, you now see that the SeeBeyond Message Node is subscribing to messages from **STCPUBLISHER** and is publishing to **PSFT\_EP**.
- 6 From the **File** menu, select **Save** to save and commit the changes to the Message Definition. You have now defined the Routing Rule that allows the appropriate Message to be published from the e\*Way to PeopleSoft 8.

## 4.2.5 Adding the PeopleSoft 8 Subscription Handler

**Note:** In performing the procedures in this section you must know the following

- PeopleSoft 8 parameters:*
- Jolt Listener Host*
  - Jolt Listener Port*
  - People Tools version*
  - Operator ID*
  - Operator ID password*

### To Access the PeopleSoft 8 Handler Directory (ConfigServlet)

- 1 Start any supported browser and open the Handler Directory by typing the following URL, where **PSFTHOST** represents the host on which PeopleSoft 8 Application Messaging Gateway is installed.

**A** For Apache:

**http://PSFTHOST/servlets/psft.pt8config.ConfigServlet**

**B** For WebLogic:

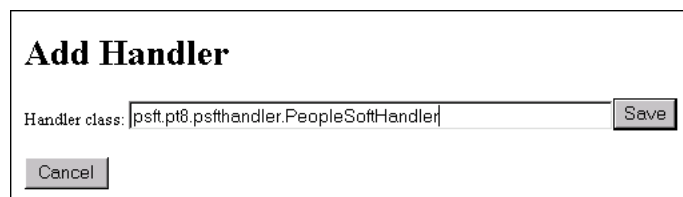
**http://PSFTHOST/servlets/gateway.administration**

**Figure 60** PeopleSoft 8 Handler Directory Page (1)



- 2 On the initial page, click the **Add handler** button to add a new handler. The **Add Handler** page appears (see Figure 61).

**Figure 61** Add Handler Page



- 3 On the **Add Handler** page, type the PeopleSoft 8 Handler class into the Handler class text box:

**psft.pt8.psfthandler.PeopleSoftHandler**

- 4 Click the **Save** button to save the newly-added PeopleSoft 8 Handler.

After the Handler has been saved, the **Handler Directory** page reappears, showing the newly-added Handler. Also, two additional buttons appear: **Load** and **Delete** (see Figure 62).

**Figure 62** PeopleSoft 8 Handler Directory Page (2)

**PeopleSoft 8.13 Handler Directory**

Handler	Status	Load	Unload	Configure	Delete
psft.pt8.psfthandler.PeopleSoftHandler	Not loaded	Load			Delete

Add handler

- Click the **Load** button to load the PeopleSoft 8 Handler class just added. The Status changes from **Not loaded** to **Loaded successfully** and two new buttons appear: **Unload** and **Configure** (see Figure 63).

**Figure 63** PeopleSoft 8 Handler Directory Page (3)

**PeopleSoft 8.13 Handler Directory**

Handler	Status	Load	Unload	Configure	Delete
psft.pt8.psfthandler.PeopleSoftHandler	Loaded successfully		Unload	Configure	

Add handler

- Click the **Configure** button to configure the handler. The **Manage Lookup Table** page for the PeopleSoft 8 Handler appears (see Figure 64).

**Figure 64** Manage Lookup Table Page (1)

**Manage Lookup Table**

Node	Machine address:port#	Tools Version	OPRID	Actions
Add a new node				

- Click the **Add a new node** button to associate the node with this subscription handler. The **Add an address** window appears (see Figure 65).

**Figure 65** Add an Address Page

**Add an address**

Node	Machine address:port#	Tools Version	OPRID	Password
PSFT_EP <small>e.g., BGEE_REMOTE</small>	//solutions9:9000 <small>e.g., //AKTT:9000,...</small>	8.13 <small>e.g., 8.10</small>	VP1 <small>e.g., PTDMO</small>	<input type="password"/> <small>e.g., PASSWORD</small>

Save address

Cancel

- 8 Type the values for the new node, PSFT\_EP, associated with the subscription handler.

*Note:* These values are required, and must be entered.

- 9 Click **Save address** to save the values just entered. You automatically return to the Manage Lookup Table page, now showing the node just added and configured (see Figure 66).

**Figure 66** Manage Lookup Table Page (2)

**Manage Lookup Table**

Node	Machine address:port#	Tools Version	OPRID	Actions		
PSFT_EP	//solutions9:9000	8.13	VP1	Edit	Delete	Add

Add a new node

The Application Messaging Gateway is now ready to receive XML messages from the e\*Way and publish the XML messages to PeopleSoft 8.

## 4.3 Configuring for Subscription

To enable PeopleSoft 8 to publish XML Messages to the PeopleSoft HTTP e\*Way, the following configuration steps must be performed.

[Creating a MUX e\\*Way Message Node](#) on page 70

[Activating the Message Definition for Subscription](#) on page 70

[Defining the Message Channel Routing Rules](#) on page 70

[Adding the SeeBeyond MUX Subscription Handler](#) on page 70

### 4.3.1 Creating a MUX e\*Way Message Node

Refer to the section [“Creating PeopleSoft 8 Message Node for the e\\*Way” on page 54](#) to create a message node associated with the Multiplexer e\*Way. A Message Node called STCMUX is used as an example.

### 4.3.2 Activating the Message Definition for Subscription

Refer to the section [“Activating the Message Definition for Publication” on page 58](#) to activate the appropriate Message to be published to the Multiplexer e\*Way. In this case, activate PO-EXPECTED\_RECEIPT\_SHIPTO Message.

### 4.3.3 Defining the Message Channel Routing Rules

Refer to the section [“Defining Message Channel Routing Rules” on page 61](#) to define the Routing rules for the Message Channel to be used.

- Insert the PSFT\_EP Message Node and the MUX Message Node previously created.
- Define the Routing Direction. Select **Subscribe From** for PSFT\_EP and **Publish To** for the MUX Message Node (STCMUX).

### 4.3.4 Adding the SeeBeyond MUX Subscription Handler

Obtain the Multiplexer configuration values for the e\*Way which is to receive the XML message(s) from PeopleSoft. These are required when configuring the Message Node corresponding to the Subscription Handler.

To Obtain the MUX Configuration Values

- 1 Access the PeopleSoft 8 Handler Directory (ConfigServlet).
- 2 Start up any supported browser.
- 3 Open the Handler Directory page by typing the following URL, where PSFTHOST represents the host on which PeopleSoft 8 Application Messaging Gateway is installed (see Figure 67):

A For Apache:

```
http://PSFTHOST/servlets/psft.pt8config.ConfigServlet
```

B For WebLogic:

**http://PSFTHOST/servlets/gateway.administration**

**Figure 67** PeopleSoft Handler Directory Page (1)

PeopleSoft 8.13 Handler Directory					
Handler	Status	Load	Unload	Configure	Delete
psft.pt8.psfthandler.PeopleSoftHandler	Loaded successfully		Unload	Configure	
<input type="button" value="Add handler"/>					

4 Click the **Add handler** button to add the MUX handler.

**Figure 68** Add Handler Page

### Add Handler

Handler class:

5 Add the SeeBeyond MUX Subscription Handler by typing the following into the Handler class box:

**com.stc.ewpsoft8.stcmuxhandler.MuxHandler**

6 Click the **Save** button. The system returns you to the **Handler Directory** page, now showing the MUX Subscription Handler you just added (see Figure 69).

**Figure 69** PeopleSoft Handler Directory Page (2)

PeopleSoft 8.13 Handler Directory					
Handler	Status	Load	Unload	Configure	Delete
psft.pt8.psfthandler.PeopleSoftHandler	Loaded successfully		Unload	Configure	
com.stc.ewpsoft8.stcmuxhandler.MuxHandler	Not loaded	Load			Delete
<input type="button" value="Add handler"/>					

7 Click the **Load** button. The Status field changes to **Loaded successfully** and two additional buttons appear: **Unload** and **Configure** (see Figure 70).

**Figure 70** PeopleSoft Handler Directory Page (3)

**PeopleSoft 8.13 Handler Directory**

Handler	Status	Load	Unload	Configure	Delete
psft.pt3.psftHandler.PeopleSoftHandler	Loaded successfully		Unload	Configure	
com.stc.ewpsoft2.stcmuxhandler.MuxHandler	Loaded successfully		Unload	Configure	

Add handler

- Click the **Configure** button for the SeeBeyond MUX Handler. The SeeBeyond MUX Handler Directory page for the MUX Handler opens (see Figure 71).

**Figure 71** SeeBeyond MUX Handler Directory Page (1)

**SeeBeyond MUX Handler Directory**

Node Name	MUX Host	MUX Port	MUX Expire [sec]	MUX Timeout [msec]	Uncompress?	Base64 Decode?	Include Headers?	Log File	Edit	Delete
Add a SeeBeyond MUX node										

- Click the **Add a SeeBeyond MUX node** button to associate a node with this Subscription Handler.

**Figure 72** Add SeeBeyond MUX Handler Page

**Add SeeBeyond MUX Handler**

Node Name	MUX Host	MUX Port	MUX Expire [secs]	MUX Timeout [msecs]	Uncompress?	Base64 Decode?	Include Headers?	Log File
STCMUX	XXX	26051	10	10000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	C:\20020122035008.log

Save

Cancel

- Enter the values for the new node associated with the subscription handler. Scroll to the right to access additional columns.
- In the **Include Headers?** column, indicate whether or not you want header information to be retained in the received messages.
  - Selecting the check box preserves the header information.
  - Deselecting the check box strips the header information.



**Note:** All values are required.

- 12 Click the **Save** button to save the values entered.

**Note:** You should now be able to ping the MUX host from the machine where the **stcph.jar** file is installed. You may need to use the full machine host name (for example, johndoe.seebeyond.com).

Click the **Save** button to display the SeeBeyond MUX Handler Directory page with the STCMUX node entries added and configured (see Figure 73).

**Figure 73** SeeBeyond MUX Handler Directory Page (2)

SeeBeyond MUX Handler Directory										
Node Name	MUX Host	MUX Port	MUX Expire [sec]	MUX Timeout [msec]	Uncompress?	Base64 Decode?	Include Headers?	Log File	Edit	Delete
STCMUX	XXX	26051	10	10000	Yes	Yes	No	C:\psft_logs\WAMESMUX.log	Edit	Delete
<input type="button" value="Add a SeeBeyond MUX node"/>										

If the entries are correct, the Application Messaging Gateway now can receive XML messages from PeopleSoft 8 and publish the XML messages to the PeopleSoft HTTP e\*Way (in MUX mode).

# Setup Procedures

This chapter summarizes the setup procedures for the e\*Way Intelligent Adapter for PeopleSoft (HTTP).

---

## 5.1 Overview

After creating a schema, you must instantiate and configure the PeopleSoft HTTP e\*Way to operate within the schema. A wide range of setup options allow the e\*Way to conform to your system's operational characteristics and your facility's operating procedures.

The topics discussed in this chapter include the following:

### Setting Up the e\*Way

[Defining e\\*Way](#) on page 75

[Modifying e\\*Way Properties](#) on page 76

[Configuring the e\\*Way](#) on page 77

[Changing the User Name](#) on page 81

[Setting Startup Options or Schedules](#) on page 81

[Activating or Modifying Logging Options](#) on page 83

[Activating or Modifying Monitoring Thresholds](#) on page 84

### Troubleshooting the e\*Way

[Configuration Problems](#) on page 85

[System-related Problems](#) on page 86

## 5.2 Setting Up the e\*Way

*Note:* The e\*Gate Schema Designer GUI runs only on the Windows operating system.

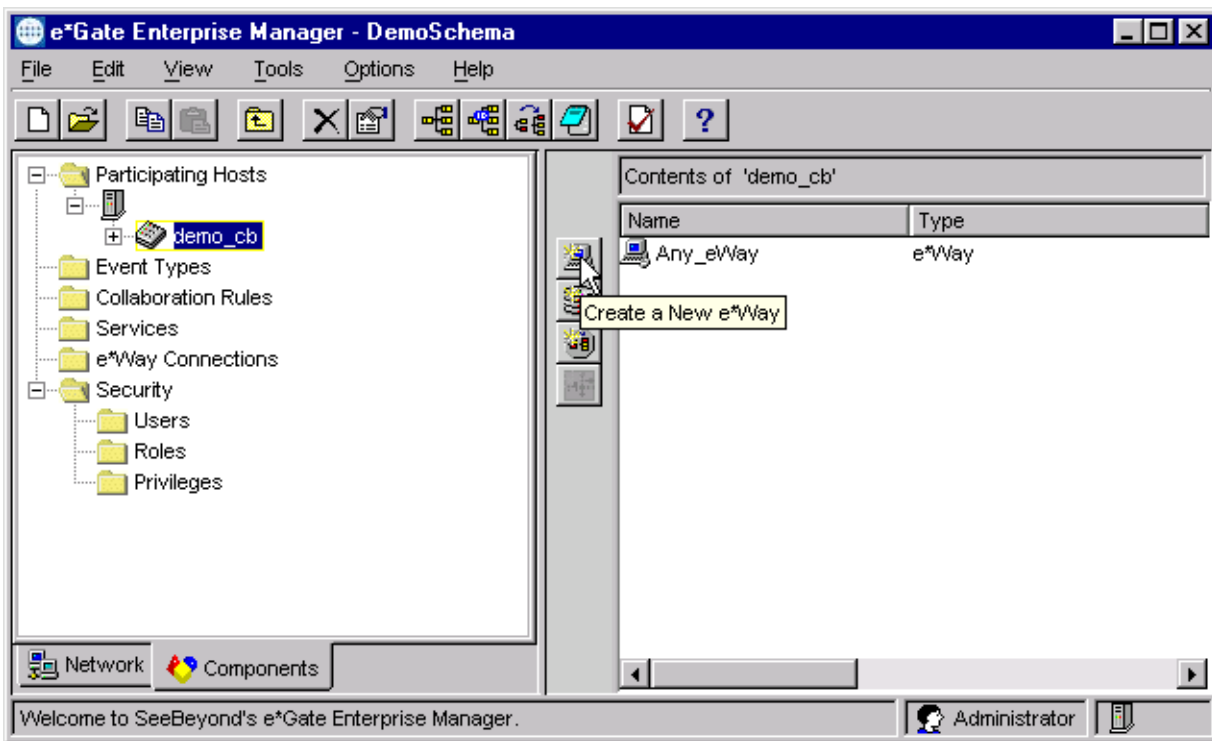
### 5.2.1 Defining e\*Way

The first step in implementing an e\*Way is to define the e\*Way component using the e\*Gate Schema Designer.

To create an e\*Way

- 1 Open the schema in which the e\*Way is to operate.
- 2 Select the e\*Gate Schema Designer Navigator's **Components** tab.
- 3 Open the host on which you want to create the e\*Way.
- 4 Select the Control Broker you want to manage the new e\*Way.

**Figure 74** e\*Gate Schema Designer Window (Components View)



- 5 On the Palette, click **Create a New e\*Way**.
- 6 Enter the name of the new e\*Way, then click **OK**.
- 7 All further actions are performed in the e\*Gate Schema Designer Navigator's **Components** tab.

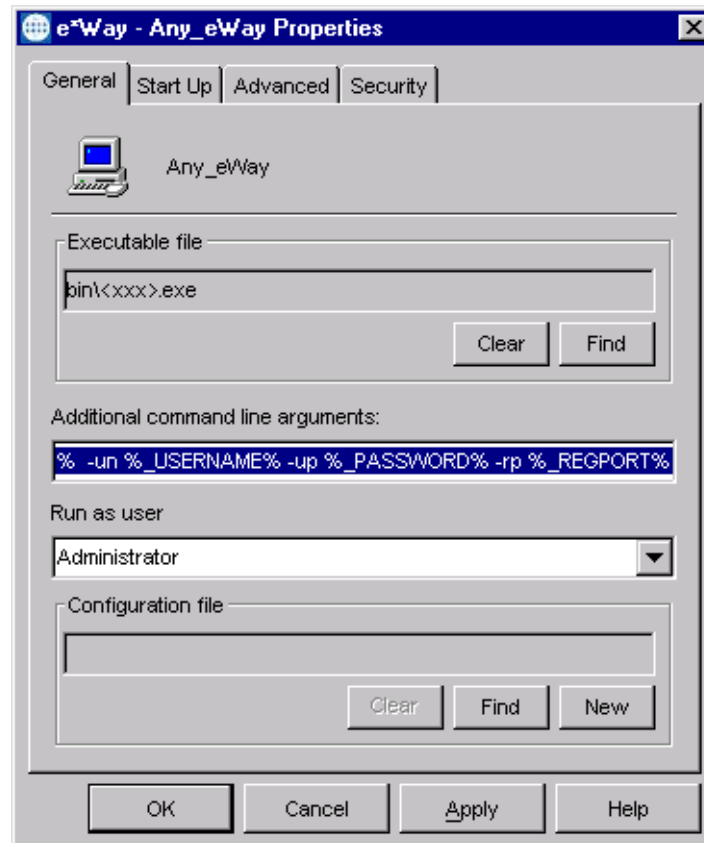
## 5.2.2 Modifying e\*Way Properties

To modify any e\*Way properties

- 1 Right-click on the desired e\*Way and select **Properties** to edit the e\*Way's properties. The properties dialog opens to the **General** tab (shown in Figure 75).

*Note:* The executable files are `stcewgenericmonk.exe` and `stcewipmp.exe`.

**Figure 75** e\*Way Properties (General Tab)



- 2 Make the desired modifications, then click **OK**.

### 5.2.3 Configuring the e\*Way

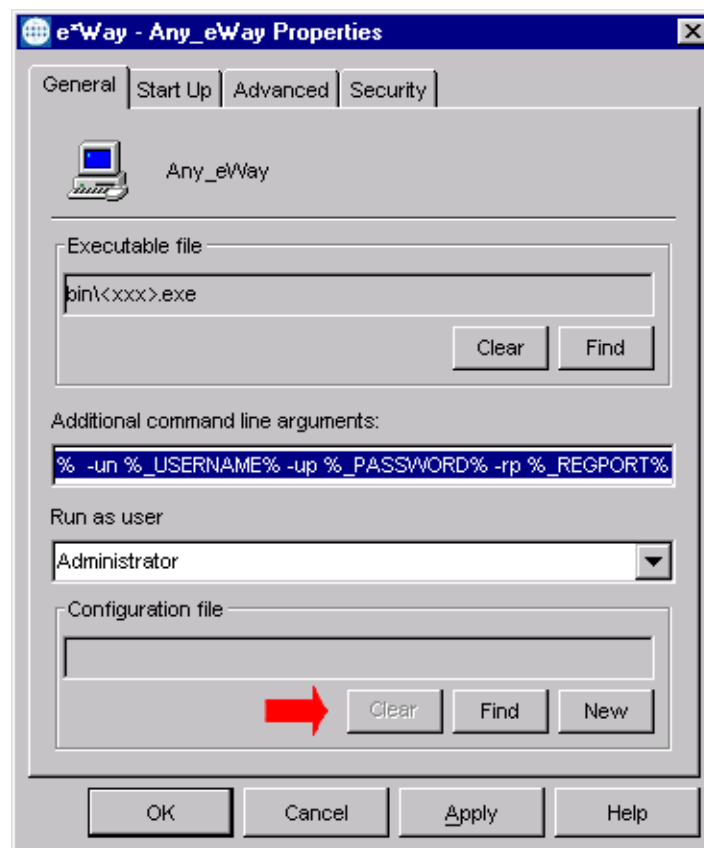
The e\*Way's default configuration parameters are stored in an ASCII text file with a .def extension. The e\*Way Editor provides a simple graphical interface for viewing and changing those parameters to create a working configuration (.cfg) file.

To change e\*Way configuration parameters

- 1 In the e\*Gate Schema Designer's Component editor, select the e\*Way you want to configure and display its properties.

*Note:* The default configuration file is **stcewpsoft8.def**.

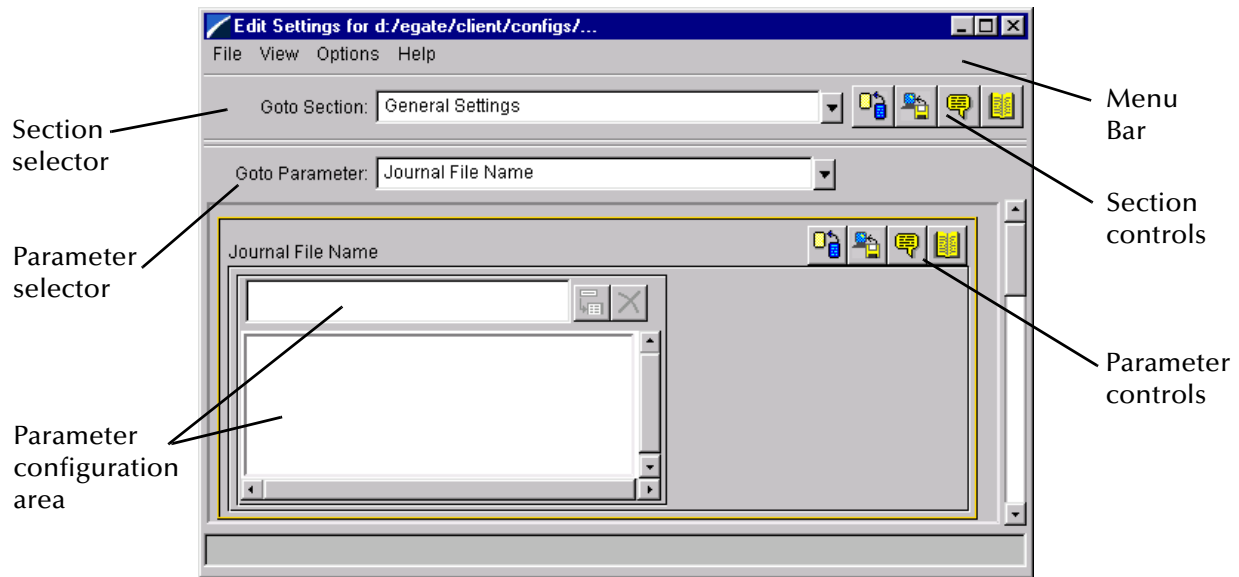
**Figure 76** e\*Way Properties - General Tab



- 2 Under **Configuration File**, click **New** to create a new file or **Find** to select an existing configuration file. If you select an existing file, an **Edit** button appears, which you can click to edit the currently selected file.
- 3 You are now in the e\*Way Configuration Editor.

## 5.2.4 Using the e\*Way Editor

Figure 77 The e\*Way Configuration Editor







The e\*Way Editor controls fall into one of six categories:

- The **Menu bar** allows access to basic operations (e.g., saving the configuration file, viewing a summary of all parameter settings, and launching the Help system)
- The **Section selector** at the top of the Editor window enables you to select the category of the parameters you wish to edit
- **Section controls** enable you to restore the default settings, restore the last saved settings, display tips, or enter comments for the currently selected section
- The **Parameter selector** allows you to jump to a specific parameter within the section, rather than scrolling
- **Parameter controls** enable you to restore the default settings, restore the last saved settings, display tips, or enter comments for the currently selected parameter
- **Parameter configuration controls** enable you to set the e\*Way's various operating parameters

## Section and Parameter Controls

The section and parameter controls are shown in Table 5 below.

**Table 5** Parameter and Section Controls

Button	Name	Function
	<b>Restore Default</b>	Restores default values
	<b>Restore Value</b>	Restores saved values
	<b>Tips</b>	Displays tips
	<b>User Notes</b>	Enters user notes



*Note: The section controls affect all parameters in the selected section, whereas the parameter controls affect only the selected parameter.*

## Parameter Configuration Controls

Parameter configuration controls fall into one of two categories:

- Option buttons
- Selection lists, which have controls as described in Table 6

**Table 6** Selection List Controls

Button	Name	Function
	<b>Add to List</b>	Adds the value in the text box to the list of available values.
	<b>Delete Items</b>	Displays a "delete items" dialog box, used to delete items from the list.

---

## Command-line Configuration

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.

---

## Getting Help

To launch the e\*Way Editor's Help system

From the **Help** menu, select **Help topics**.

To display tips regarding the general operation of the e\*Way

From the **File** menu, select **Tips**.

To display tips regarding the selected Configuration Section

In the **Section Control** group, click .

To display tips regarding the selected Configuration Parameter

In the **Parameter Control** group, click .

**Note:** *“Tips” are displayed and managed separately from the online Help system. You cannot search for Tips within the Help system, or view Help system topics by requesting Tips.*

For detailed descriptions and procedures for using the e\*Way Configuration Editor, see the *e\*Gate Integrator User's Guide*.



## 5.2.5 Changing the User Name

Like all e\*Gate executable components, e\*Ways run under an e\*Gate user name. By default, all e\*Ways run under the **Administrator** user name. You can change this if your site's security procedures so require.

### To change the user name

- 1 Display the e\*Way's properties dialog.
- 2 On the **General** tab, use the **Run as user** list to select the e\*Gate user under whose name this component runs.

See the *e\*Gate Integrator System Administration and Operations Guide* for more information on the e\*Gate security system.

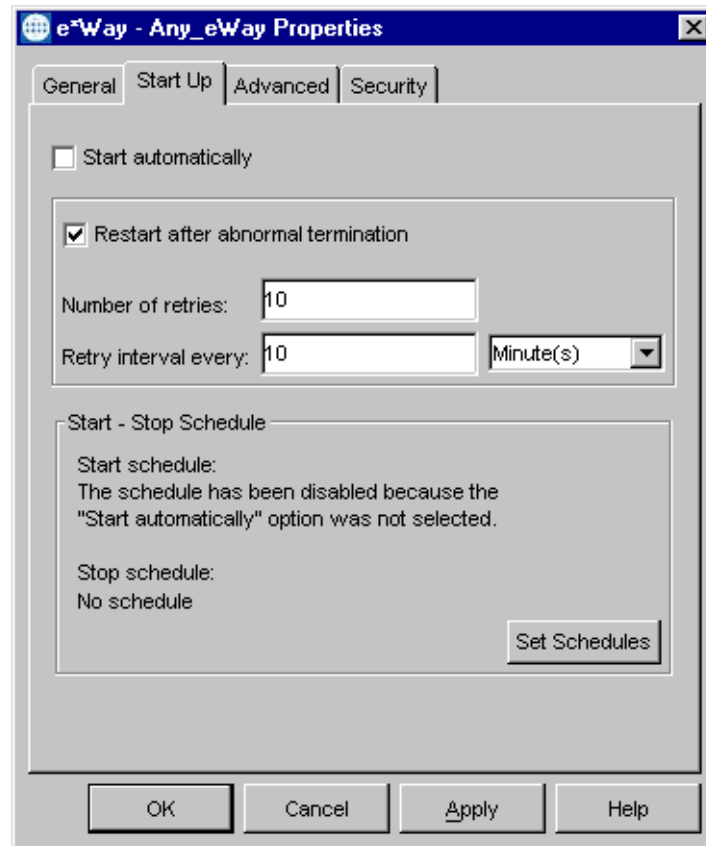
## 5.2.6 Setting Startup Options or Schedules

SeeBeyond e\*Ways can be started or stopped by any of the following methods:

- The Control Broker can start the e\*Way automatically whenever the Control Broker starts.
- The Control Broker can start the e\*Way automatically whenever it detects that the e\*Way terminated execution abnormally.
- The Control Broker can start or stop the e\*Way on a schedule that you specify.
- Users can start or stop the e\*Way manually using an interactive monitor.

You determine how the Control Broker starts or shuts down an e\*Way using options on the e\*Way properties **Start Up** tab (see Figure 78). See the *e\*Gate Integrator System Administration and Operations Guide* for more information about how interactive monitors can start or shut down components.

Figure 78 e\*Way Properties (Start-Up Tab)



To set the e\*Way's startup properties

- 1 Display the e\*Way's properties dialog.
- 2 Select the **Start Up** tab.
- 3 To have the e\*Way start automatically when the Control Broker starts, select the **Start automatically** check box.
- 4 To have the e\*Way start manually, clear the **Start automatically** check box.
- 5 To have the e\*Way restart automatically after an abnormal termination:
  - A Select **Restart after abnormal termination**.
  - B Set the desired number of retries and retry interval.
- 6 To prevent the e\*Way from restarting automatically after an abnormal termination, clear the **Restart after abnormal termination** check box.
- 7 Click **OK**.

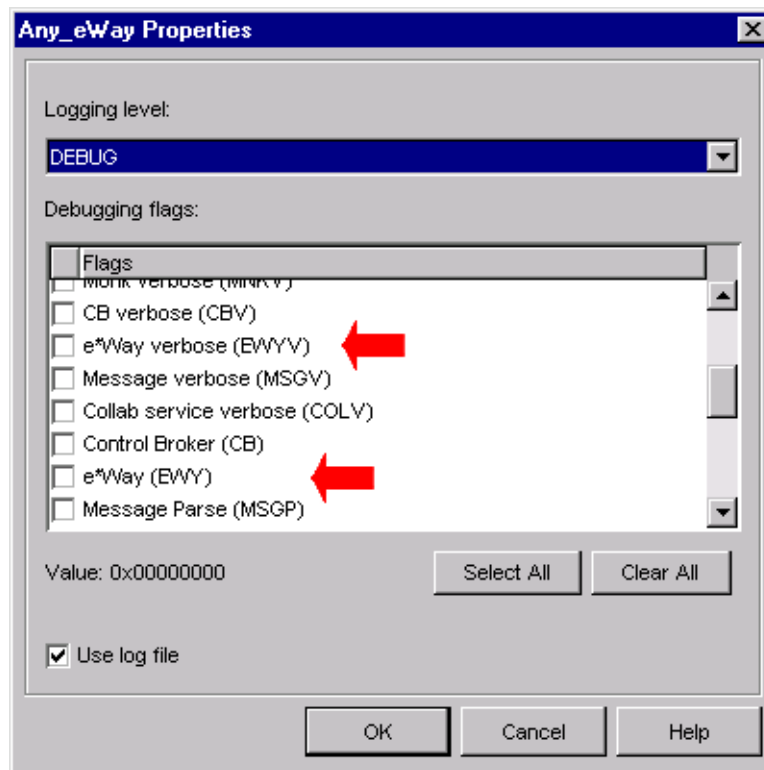
## 5.2.7 Activating or Modifying Logging Options

Logging options enable you to troubleshoot problems with the e\*Way and other e\*Gate components.

To set the e\*Way debug level and flag

- 1 Display the e\*Way's Properties dialog.
- 2 Select the **Advanced** tab.
- 3 Click **Log**, and the dialog window appears (see Figure 79).

**Figure 79** e\*Way Properties (Advanced Tab - Log Option)



- 4 Select **DEBUG** for the **Logging level**.
- 5 Select either **e\*Way (EWY)** or **e\*Way Verbose (EWYV)** for the **Debugging flag**. Note that the latter has a significant negative impact on system performance.
- 6 Click **OK**.

The other options apply to other e\*Gate components and are activated in the same manner. See the *e\*Gate Integrator Alert and Log File Reference* for additional information concerning log files, logging options, logging levels, and debug flags.

## 5.2.8 Activating or Modifying Monitoring Thresholds

Monitoring thresholds enable you to monitor the throughput of the e\*Way. When the monitoring thresholds are exceeded, the e\*Way sends a Monitoring Event to the Control Broker, which routes it to the Schema Manager and any other configured destinations.

- 1 Display the e\*Way's properties dialog.
- 2 Select the **Advanced** tab.
- 3 Click **Thresholds**.
- 4 Set the desired threshold options and click **OK**.

See the *e\*Gate Integrator Alert and Log File Reference* for more information concerning threshold monitoring, routing specific notifications to specific recipients, or for general information about e\*Gate's monitoring and notification system.

---

## 5.3 Troubleshooting the e\*Way

In the initial stages of developing your e\*Gate Integrator system administration system, most problems with e\*Ways can be traced to configuration.

### 5.3.1 Configuration Problems

#### In the Schema Designer

- Does the e\*Way have the correct Collaborations assigned?
- Do those Collaborations use the correct Collaboration Services?
- Is the logic correct within any Collaboration Rule script employed by this e\*Way's Collaborations?
- Do those Collaborations subscribe to and publish Events appropriately?
- Are all the components that "feed" this e\*Way properly configured, and are they sending the appropriate Events correctly?
- Are all the components that this e\*Way "feeds" properly configured, and are they subscribing to the appropriate Events correctly?

#### In the e\*Way Editor

- Check that all configuration options are set appropriately.
- Check that all settings you changed are set correctly.
- Check all required changes to ensure they have not been overlooked.
- Check the defaults to ensure they are acceptable for your installation.

#### On the e\*Way's Participating Host

- Check that the Participating Host is operating properly, and that it has sufficient disk space to hold the IQ data that this e\*Way's Collaborations publish.
- Check that the *path* environment variable includes the location of the PeopleSoft HTTP dynamically-loaded libraries. The name of this variable on the different operating systems is:
  - ♦ PATH (Windows)
  - ♦ LD\_LIBRARY\_PATH (Solaris)
  - ♦ LIBPATH (AIX)

#### In the PeopleSoft Application

- Check that the application is configured correctly, is operating properly, and is sending or receiving the correct data appropriately.

## 5.3.2 System-related Problems

- Check that the connection between the external application and the e\*Way is functioning appropriately.
- Once the e\*Way is up and running properly, operational problems can be due to:
  - ♦ External influences (network or other connectivity problems).
  - ♦ Problems in the operating environment (low disk space or system errors)
  - ♦ Problems or changes in the data the e\*Way is processing.
  - ♦ Corrections required to Collaboration Rule scripts that become evident in the course of normal operations.

**Note:** *If you are running AIX 4.3.3, you should ensure that you have **bos.rte.libpthreads** version 4.3.3.51 or later. Earlier versions contain a recognized bug that can produce a memory leak. A patch is available from IBM.*

One of the most important tools in the troubleshooter's arsenal is the e\*Way log file. See the *e\*Gate Integrator Alert and Log File Reference Guide* for an extensive explanation of log files, debugging options, and using the Schema Manager to monitor operations and performance.

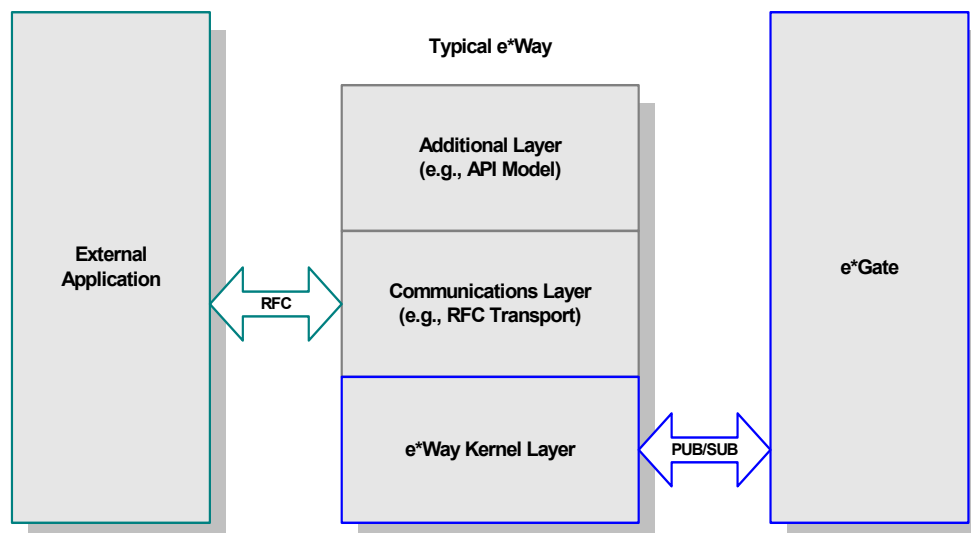
# Operational Overview

This chapter contains an overview of the architecture and basic internal processes of the e\*Way Intelligent Adapter for PeopleSoft (HTTP).

## 6.1 e\*Way Architecture

Conceptually, an e\*Way can be viewed as a multi-layered structure, consisting of one or more layers (see Figure 80). Each layer contains Monk scripts and/or functions, and makes use of lower-level Monk functions residing in the layer beneath. You, as user, primarily use the highest-level functions, which reside in the upper layer(s).

**Figure 80** Typical e\*Way Architecture

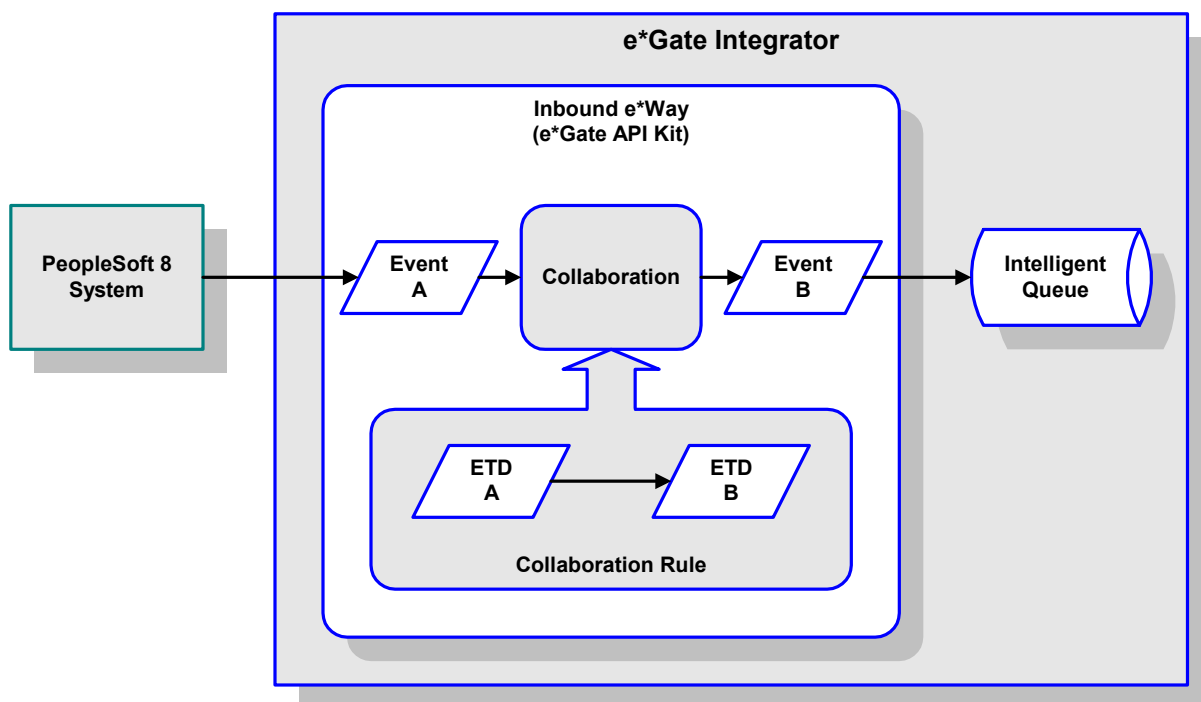


The upper layers of the e\*Way use Monk functions to perform Business Process modeling and ETD mapping, package data as e\*Gate *Events*, send those Events to Collaborations, and manage interaction with the external system. These layers are built upon an e\*Way Kernel layer that manages the basic operations of the e\*Way, data processing, and communication with other e\*Gate components.

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

Collaborations execute the business logic that enable the e\*Way to do its intended work. In turn, each Collaboration executes a Collaboration Rule, containing the actual instructions to execute the business logic. Each Collaboration that publishes its processed Events internally (within e\*Gate Integrator) requires one or more IQs to receive the Events, as shown in Figure 81. Any Collaboration that publishes its processed Events only to an external system does *not* require *any* IQs.

**Figure 81** Collaborations and IQs



Configuration options that control the Monk environment and define the Monk functions used to perform various e\*Way operations are discussed in [Chapter 7](#). You can create and modify these functions using the SeeBeyond Collaboration Rules Editor or a text editor (such as *Microsoft Word* or *Notepad*, or UNIX *vi*). The available set of e\*Way API functions is described in [Chapter 8](#). Generally, e\*Way Kernel Monk functions should be called directly only when there is a specific need not addressed by higher-level Monk functions, and should be used only by experienced developers.

For more information on defining Collaborations, defining IQs, assigning Collaborations to e\*Ways, or configuring Collaborations to publish Events, see the *e\*Gate Integrator User's Guide*.

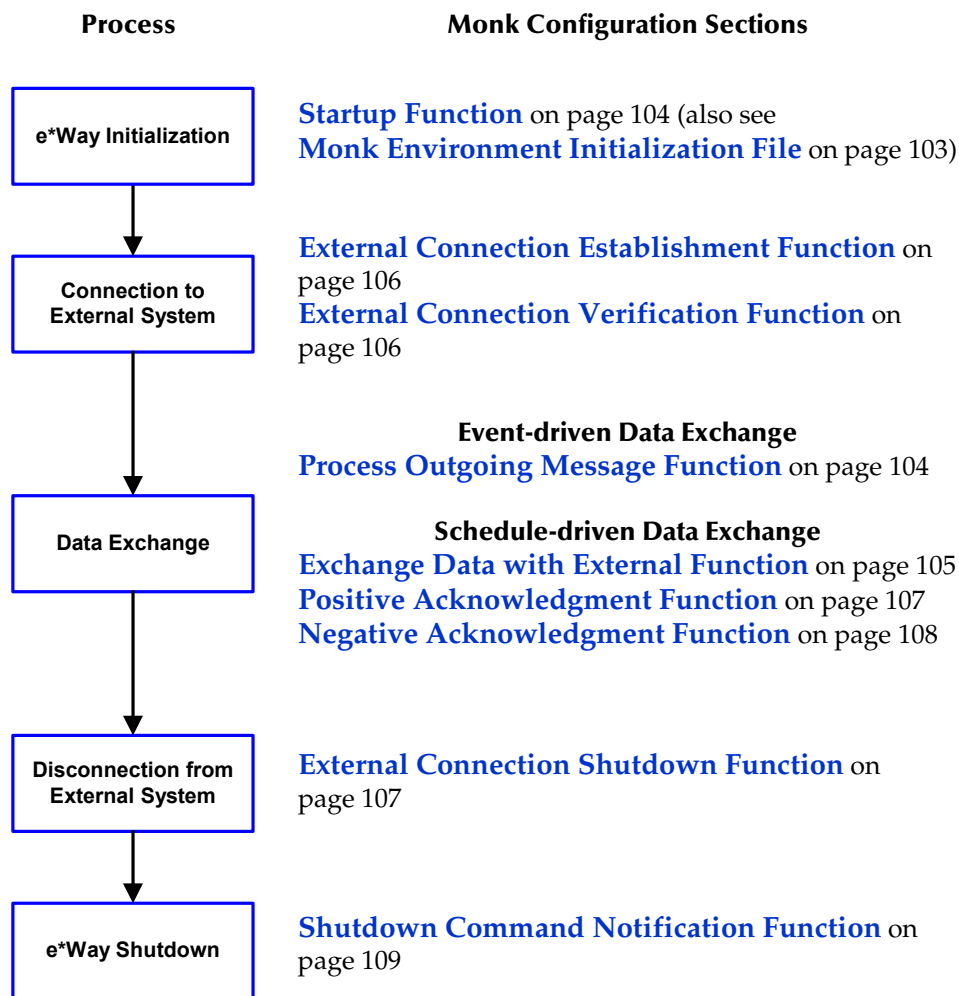


## 6.2 Basic e\*Way Processes

**Note:** This section describes the basic operation of a typical e\*Way based on the Generic e\*Way Kernel. Not all functionality described in this section is used routinely by this e\*Way.

The most basic processes carried out by an e\*Way are listed in Figure 82. In e\*Ways based on the Generic Monk e\*Way Kernel (using `stcewgenericmonk.exe`), these processes are controlled by the listed Monk functions. Configuration of these functions is described in the referenced sections of this User's Guide.

**Figure 82** Basic e\*Way Processes

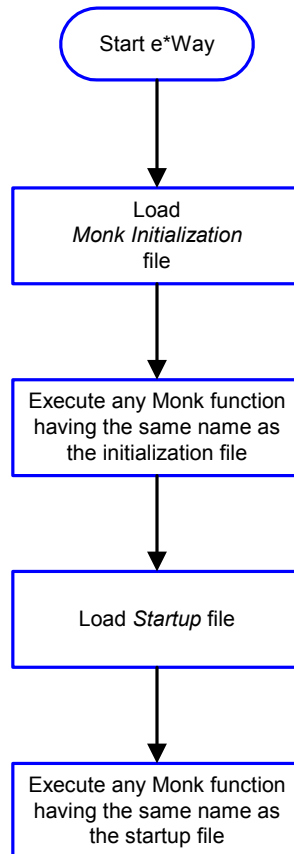


A series of diagrams on the next several pages illustrate the interaction and operation of these functions during the specified processes. Configuring the parameters associated with these functions is covered in [Chapter 7](#), while the functions themselves are described in [Chapter 8](#).

## Initialization Process

Figure 83 illustrates the e\*Way's initialization process, using the **Monk Environment Initialization File** and **Startup Function**.

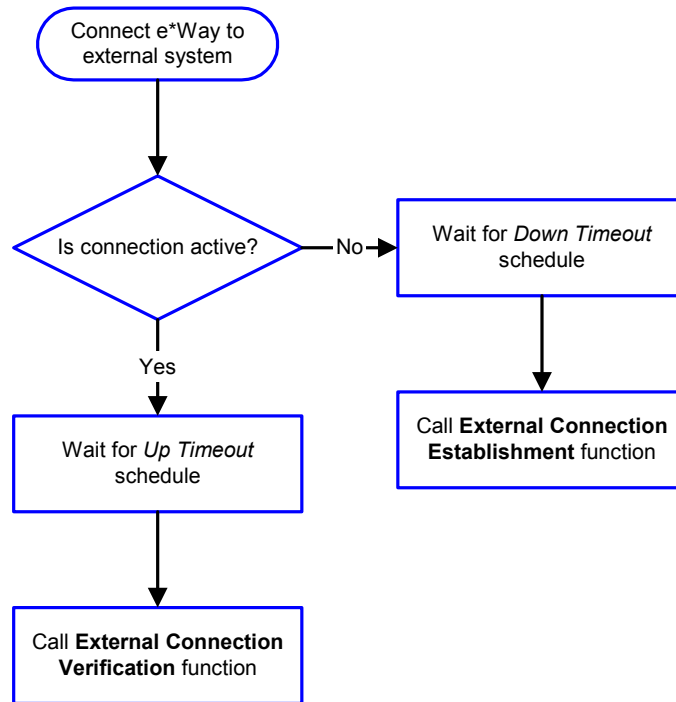
**Figure 83** Initialization Process



## Connect to External Process

Figure 84 illustrates how the e\*Way connects to the external system, using the **External Connection Establishment Function** and **External Connection Verification Function**.

**Figure 84** Connection Process



**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 86 on page 93** and **Figure 85 on page 92** 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 137 and **send-external-down** on page 137 for more information.

## Data Exchange Process

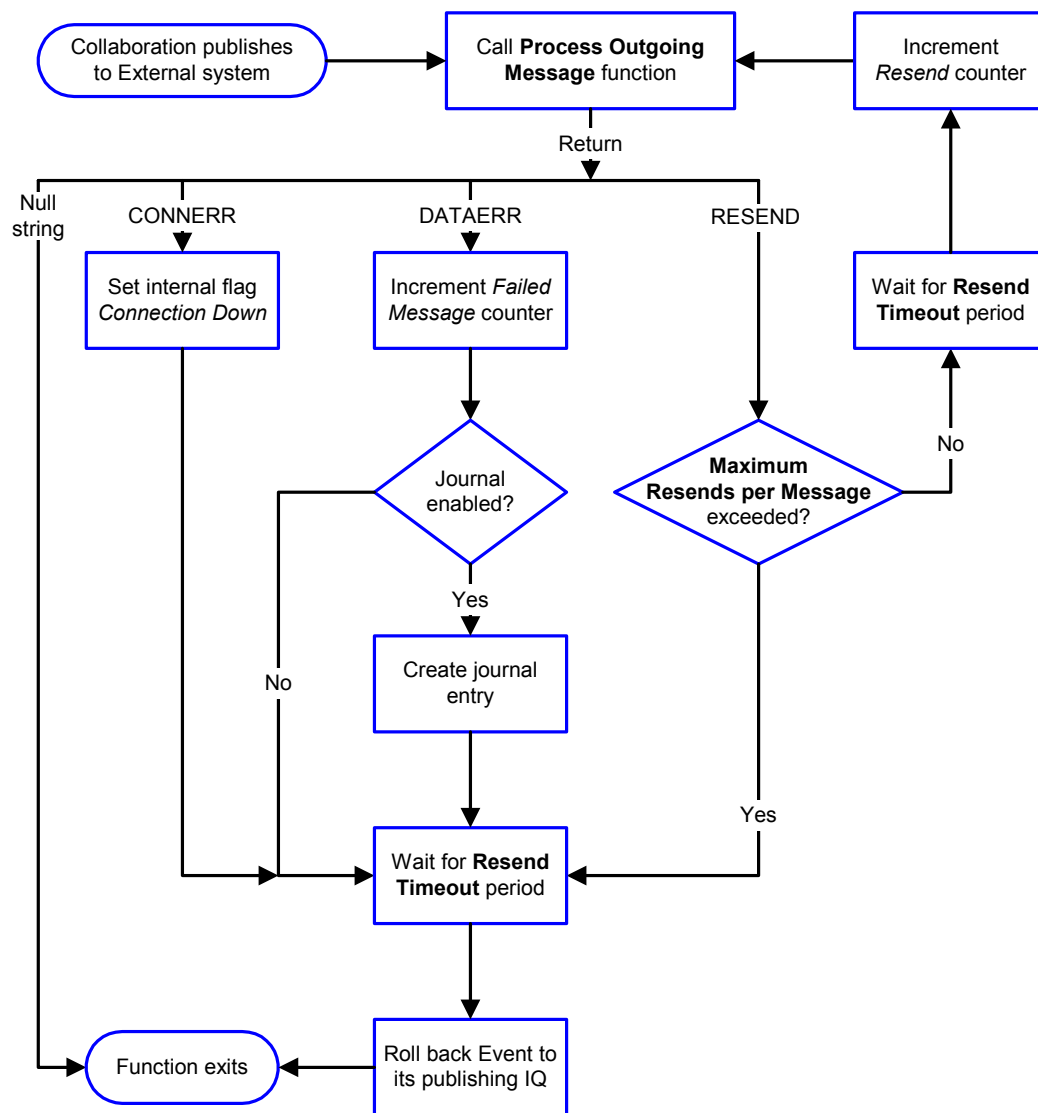
### Event-driven

Figure 85 illustrates how the e\*Way's event-driven data exchange process works, using the **Process Outgoing Message Function**.

The e\*Way periodically 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.

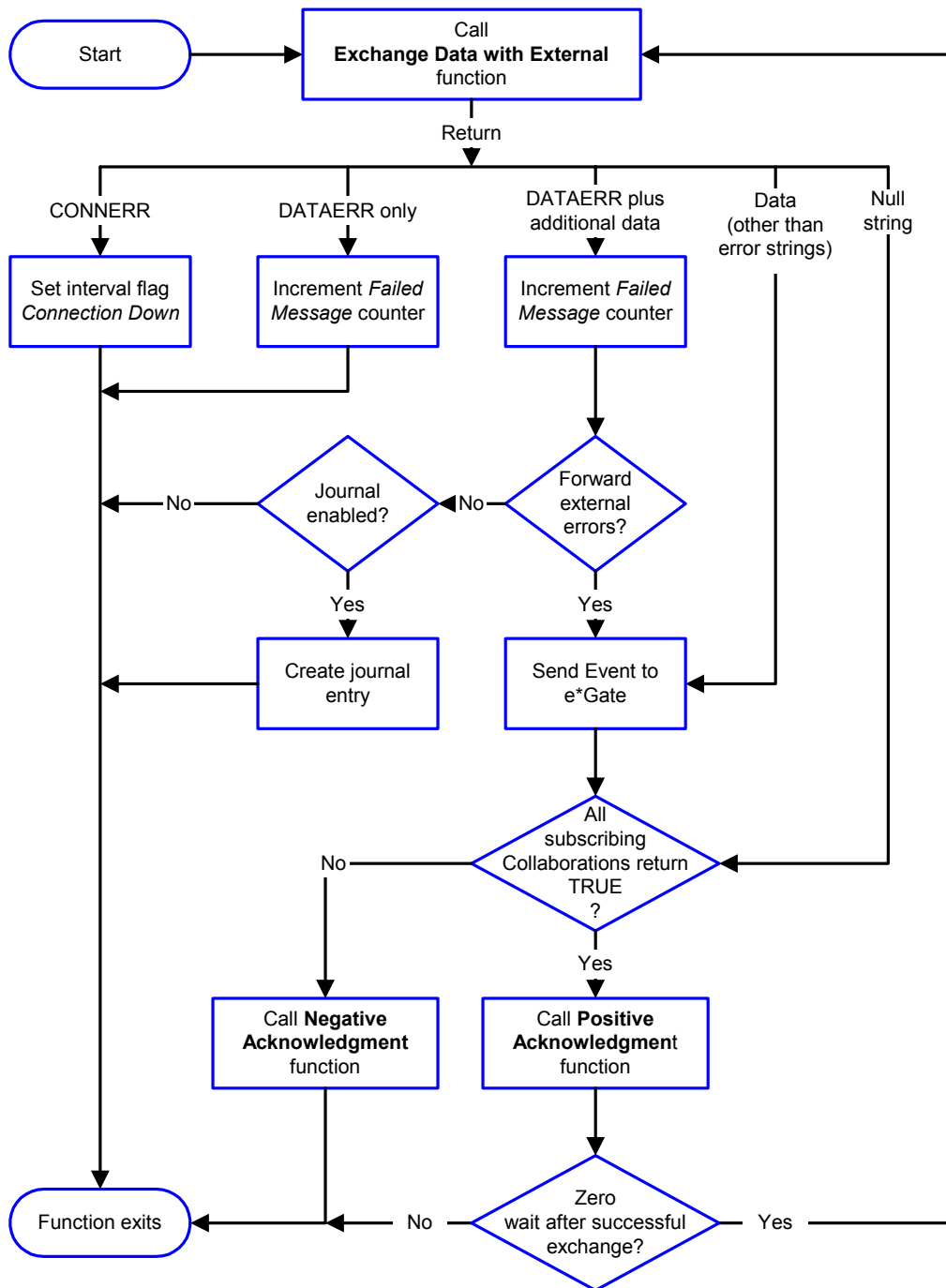
**Figure 85** Event-Driven Data Exchange Process



Schedule-driven

Figure 86 illustrates how the e\*Way's schedule-driven data exchange process works for incoming data, using the **Exchange Data with External Function, Positive Acknowledgment Function**, and **Negative Acknowledgment Function**.

**Figure 86** Schedule-Driven Data Exchange Process



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

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

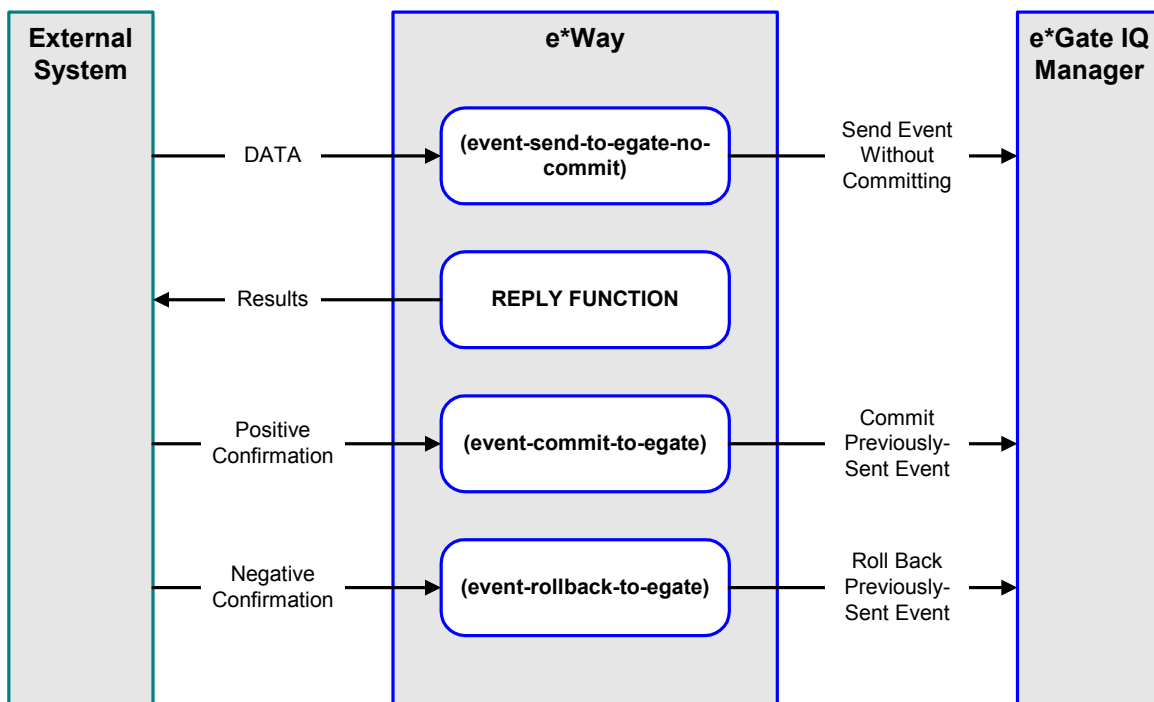
*Send Events to e\*Gate* can be implemented using any of the following Monk functions:

- **event-send-to-egate**
- **event-send-to-egate-ignore-shutdown**
- **event-send-to-egate-no-commit**

The last of these is used when confirmation of correct transmission is required from the external system. In this case, the e\*Way sends information back to the external system after receiving data. Depending upon whether the acknowledgment is positive or negative, you subsequently use one of the following functions to complete the process (see Figure 87):

- **event-commit-to-egate**
- **event-rollback-to-egate**

**Figure 87** Send Event to e\*Gate with Confirmation

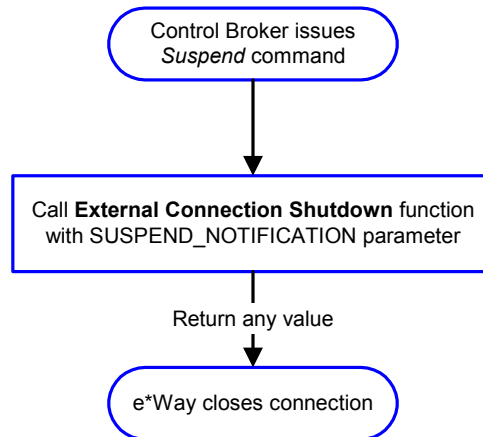


After the function exits, the e\*Way waits for the next *Start* time or command.

## Disconnect from External Process

Figure 88 illustrates how the e\*Way disconnects from the external system, using the **External Connection Shutdown Function**.

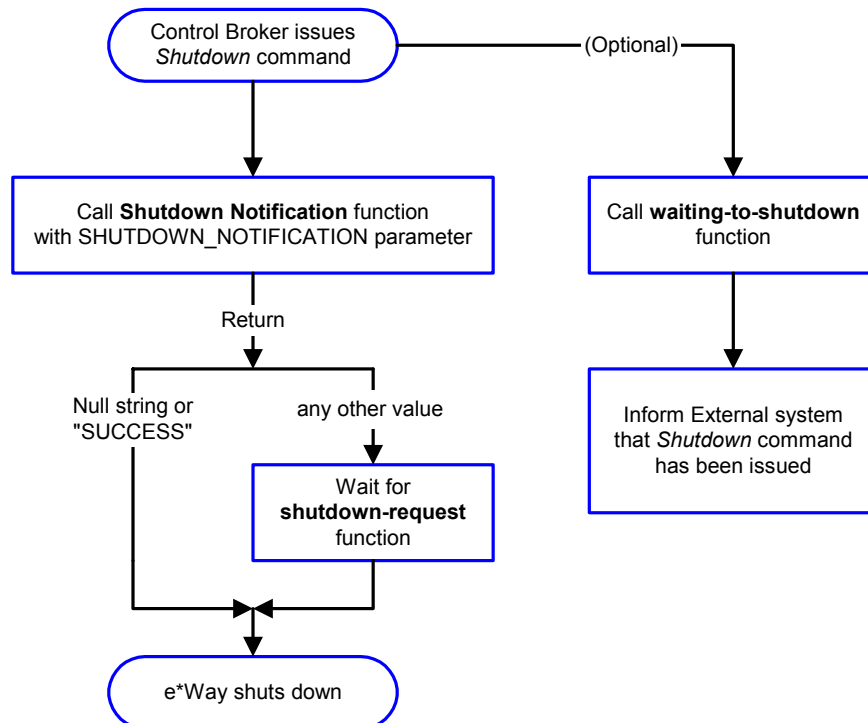
**Figure 88** Disconnect Process



## Shutdown Process

Figure 89 illustrates how the e\*Way shuts itself down, using the **Shutdown Command Notification Function**.

**Figure 89** Shutdown Process



# Configuration Parameters

This chapter describes the configuration parameters for the e\*Way Intelligent Adapter for PeopleSoft (HTTP).

---

## 7.1 Overview

The e\*Way's configuration parameters are set using the e\*Way Editor; see [Configuring the e\\*Way](#) on page 77 for procedural information. The Monk-enabled PeopleSoft HTTP e\*Way's configuration parameters are organized into the following sections. The default configuration is provided in `stcewpsoft8.def`.

[General Settings](#) on page 97

[Communication Setup](#) on page 99

[Monk Configuration](#) on page 102

[HTTP Configuration](#) on page 110

[HTTP Proxy Configuration](#) on page 113

[HTTP SSL Configuration](#) on page 115

[PeopleSoft Application Messaging](#) on page 117



---

## 7.2 General Settings

The General Settings control basic operational parameters.

---

### Journal File Name

#### Description

Specifies the name of the journal file.

#### Required Values

A valid filename, optionally including an absolute path (for example, `c:\temp\filename.txt`). If an absolute path is not specified, the file 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 Journalled 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
- 

### Max Resends Per Message

#### Description

Specifies the number of times the e\*Way attempts to resend a message (Event) to the external system after receiving an error. When this maximum is reached, the e\*Way waits for the number of seconds specified by the [Resend Timeout](#) parameter, and then rolls back the Event to its publishing IQ.

#### Required Values

An integer from 1 through 1,024. The default is 5.

---

### Max Failed Messages

#### Description

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

#### Required Values

An integer from 1 through 1,024. The default is 3.

---

---

## Forward External Errors

### Description

Selects whether or not error messages received from the external system that begin with the string "DATAERR" are queued to the e\*Way's configured queue. See [Exchange Data with External Function](#) on page 105 for more information.

### Required Values

**Yes** or **No**. The default value, **No**, specifies that error messages are not to be forwarded. See [Data Exchange Process](#) on page 92 for more information about how the e\*Way uses this function.

---

## 7.3 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 e\*Gate Schema Designer controls when the e\*Way executable runs. The schedule that you set within the parameters discussed in this section (using the e\*Way Editor) determines when data are exchanged. Be sure you set the "exchange data" schedule to fall within the "run the executable" schedule.*

---

### 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 from 0 through 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 setting of this parameter is ignored and the e\*Way invokes the **Exchange Data with External Function** immediately
- If it is desired to invoke the **Exchange Data with External Function** again as soon as possible when data is **not** queued to e\*Gate via the return mechanism, the e\*Way Kernel Monk function **insert-exchange-data-event** can be called directly (prior to leaving the exchange function) to accomplish this
- If this parameter is set to zero, then no exchange data schedule is set, and the **Exchange Data with External Function** is never called

#### See also

[Start Exchange Data Schedule](#) on page 100

[Stop Exchange Data Schedule](#) on page 101

---

### Zero Wait Between Successful Exchanges

#### Description

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

#### Required Values

Yes or No. The default is No.

### Additional Information

- If this parameter is set to **Yes**, and the previous exchange function returned data, the e\*Way invokes the **Exchange Data with External Function** immediately
- If it is desired to invoke the **Exchange Data with External Function** again as soon as possible when data is **not** queued to e\*Gate via the return mechanism, the e\*Way Kernel Monk function **insert-exchange-data-event** can be called directly (prior to leaving the exchange function) to accomplish this
- 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**

---

## Start Exchange Data Schedule

### Description

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

### Required Values

One of the following:

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

### Other Requirements

If you set a schedule using this parameter, you must also define *all* of the following parameters. If you do not, the e\*Way terminates execution when the schedule attempts to start.

- **Exchange Data with External Function**
- **Positive Acknowledgment Function**
- **Negative Acknowledgment Function**

### Additional Information

When the schedule starts, the e\*Way determines whether or not:

- it is waiting to send an **ACK** or **NAK** to the external system (using the **Positive Acknowledgment Function** or **Negative Acknowledgment Function**)
- the connection to the external system is active

If *no* **ACK/NAK** is pending and the connection *is* active, the e\*Way immediately executes the **Exchange Data with External Function**. Thereafter, the **Exchange Data with External Function** is called according to the **Exchange Data Interval** parameter until the **Stop Exchange Data Schedule** time is reached.

---

## 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, regular, interval (such as weekly, daily, or every  $n$  seconds)

---

## Down Timeout

### Description

Specifies the number of seconds for the e\*Way to wait between calls to the **External Connection Establishment Function**.

### Required Values

An integer from 1 through 86,400. The default is 15.

---

## Up Timeout

### Description

Specifies the number of seconds for the e\*Way to wait between calls to the **External Connection Verification Function** to verify that the connection is still up.

### Required Values

An integer from 1 through 86,400. The default is 15.

---

## Resend Timeout

### Description

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

### Required Values

An integer from 1 through 86,400. The default is 15.

---

## 7.4 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.

### Specifying Function or File Names

Parameters that require the name of a Monk function accept either a function name (implied by the absence of a period <.>) or the name of a file (optionally including path information) containing a Monk function. If a file name is specified, the function invoked is given by the base name of the file (for example, for a file named **my-startup.monk**, the e\*Way would attempt to execute the function **my-startup**). If path information is specified, that path is appended to the **Load Path**.

If you specify a file name, be sure that the file has one of the following extensions:

- .monk
- .tsc
- .dsc

### Specifying Multiple Directories

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.

### Load Path

The Monk *load path* is the path Monk uses to locate files and data (set internally within Monk). The default load paths are determined by the **SharedExe** and **SystemData** settings in the **.egate.store** file. See the *e\*Gate Integrator System Administration and Operations Guide* for more information about this file.

---

## Additional Path

### Description

Specifies a path to be appended to the **Load Path**. A directory specified here is searched *after* searching the default load path.

### Required Values

A pathname, or a series of paths separated by semicolons. There is no default value for this parameter.

**Note:** *This parameter is optional and may be left blank.*

### Additional information

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 is automatically loaded into the e\*Way's Monk environment.

### Required Values

A pathname, or a series of paths separated by semicolons. The default value is **monk\_library/ewpsoft8**.

*Note:* This parameter is optional and may be left blank.

---

## Monk Environment Initialization File

### Description

Specifies a file that contains environment initialization functions, which is loaded after the **Auxiliary Library Directories** are loaded. Any environment initialization functions called by this file accept no input, and must return a string.

### Required Values

A filename within the **Load Path**, or filename plus path information (relative or absolute). If path information is specified, that path is appended to the load path. The default value is **pssoft8-init**.

*Note:* This parameter is optional and may be left blank.

### Returns

The string "FAILURE" indicates that the function failed, and the e\*Way exits; any other string, including a *null string*, indicates success.

### Additional information

- 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); it is 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
- The e\*Way loads this file and try to invoke a function of the same base name as the file name

---

## 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. It is called after the e\*Way loads the specified **Monk Environment Initialization File** and any files within the specified **Auxiliary Library Directories**. This function accepts no input, and must return a string.

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 containing a Monk function. The default option is the standard function template, **psoft8-startup**.

*Note:* This parameter is optional and may be left blank.

### Returns

The string "FAILURE" indicates that the function failed, and the e\*Way exits; any other string (including a null string) indicates success.

---

## 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, rather than schedule-driven). The function requires a non-null string as input (i.e., the outgoing Event to be sent), and must return a string.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is **psoft8-outgoing**.

*Note:* This parameter is **required**, and must **not** be left blank.

### Returns

- A *null string* ("") indicates that the Event was published successfully to the external system
- A string beginning with **RESEND** indicates that the Event should be resent
- A string beginning with **CONNERR** indicates that there is a problem with the connection to the external system, and causes a rollback of the Event
- A string beginning with **DATAERR** indicates that there is a problem with the message (Event) data itself, and causes a rollback of the Event
- A string beginning with **SHUTDOWN** indicates that the e\*Way must exit immediately



- If any string other than one of the preceding is returned, the e\*Way creates an entry in the log file indicating that an attempt has been made to access an unsupported function

#### Additional Information

- The e\*Way invokes this function when one of its Collaborations publishes an Event to an *external* destination (as specified within the e\*Gate Schema Designer).
- Once this function has been called with a *non-null string*, the e\*Way does not process another Event until the current Event has been completely processed.

**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.*

---

## 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 invoked automatically by the [Start Exchange Data Schedule](#) or manually by the [start-schedule](#) Monk function, and is responsible for either sending data to or receiving data from the external system. If this function returns data, it is queued to e\*Gate in an inbound Collaboration. The e\*Way must have at least one Collaboration configured suitably to process the inbound Event, as well as any required IQs.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is [pssoft8-exchange](#).

**Note:** *This parameter is **conditional** and must be supplied only if the [Exchange Data Interval](#) is set to a non-zero value.*

### Returns

- A *null string* ("" ) indicates that the data exchange was completed successfully, but with no resultant data sent back to the e\*Gate system
- A string beginning with **CONNERR** indicates that there is a problem with the connection to the external system
- A string beginning with **DATAERR** indicates that there is a problem with the message (Event) data itself. If the error string contains data beyond the keyword, the entire string is queued to e\*Gate if an inbound Collaboration is so configured and [Forward External Errors](#) is set to **Yes**. Queueing, however, is performed without the subsequent sending of a **ACK** or **NAK** to the external system.
- Any other string indicates that the contents of the string are packaged as an inbound Event

### Additional Information

- Data can be queued directly to e\*Gate by using the [event-send-to-egate](#) Monk function or, if a two-phase approach is required, by using [event-send-to-egate-no-commit](#) and then [event-commit-to-egate](#) or [event-rollback-to-egate](#) to commit or rollback the enqueued events, as appropriate

*Note:* Until an Event is committed, it is not revealed to subscribers of that Event.

---

## External Connection Establishment Function

### Description

Specifies a Monk function that the e\*Way calls (repeatedly) when it has determined that the connection to the external system is down. The function accepts no input and must return a string.

This function is executed according to the interval specified within the [Down Timeout](#) parameter, and is called *only* according to this schedule. Once the e\*Way has determined that its connection to the external system is up, it calls the [External Connection Verification Function](#) (see next).

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is [psoft8-connect](#).

*Note:* This parameter is **required**, and must **not** be left blank.

### Returns

- A string beginning with **SUCCESS** or **UP** indicates that the connection was established successfully
- A string beginning with **DOWN** indicates that the connection was not established successfully
- Any other string, including a *null string*, indicates that the attempt to establish the connection failed and the external state is unknown

---

## 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. It is executed according to the interval specified within the [Up Timeout](#) parameter, and is called *only* according to this schedule.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is [psoft8-verify](#).

*Note:* This parameter is **optional** and may be left blank.

### Returns

- “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

### Additional Information

If this function is not specified, the e\*Way executes the **External Connection Establishment Function** in its place. This latter function also 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. This function is invoked only when the e\*Way receives a *suspend* command from a Control Broker.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is **pssoft8-shutdown**.

*Note:* This parameter is **required**, and must **not** be left blank.

### Input

A string indicating the purpose for shutting down the connection.

- “SUSPEND\_NOTIFICATION” - the e\*Way is being suspended or shut down
- “RELOAD\_NOTIFICATION” - the e\*Way is being reconfigured

### Returns

A string, the value of which is ignored. Any return value indicates that the *suspend* command can proceed and that the connection to the external system can be broken immediately.

*Note:* Include in this function any required “clean up” operations that must be performed as part of the shutdown procedure, but before the e\*Way exits.

---

## Positive Acknowledgment Function

### Description

This function is loaded during the initialization process and is called when all data received from the external system has been processed and enqueued successfully. The function requires a non-null string as input (the Event to be sent to e\*Gate) and must return a string.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is **psoft8-ack**.

*Note:* This parameter is **required**, and must **not** be left blank.

### Input

A string, the inbound Event to e\*Gate.

### Returns

- The string beginning with **CONNERR** indicates a problem with the connection to the external system; when the connection is re-established, the function is called again, with the same input data
- Any other string, including a *null string*, indicates that the acknowledgement has been sent to the external system successfully

### Additional Information

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. The e\*Way executes this function only if the Event's processing is completed successfully by *all* the Collaborations to which it was sent; otherwise, the e\*Way executes the **Negative Acknowledgment Function**.

*Note:* If you configure the acknowledgment function to return a non-null string, you must configure a Collaboration (with appropriate IQs) to process the returned Event.

---

## Negative Acknowledgment Function

### Description

This function is loaded during the initialization process and is called when the e\*Way fails to process or enqueue data received from the external system successfully. The function requires a non-null string as input (the Event to be sent to e\*Gate) and must return a string.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is **psoft8-nack**.

*Note:* This parameter is **required**, and must **not** be left blank.

### Input

A string, the inbound Event to e\*Gate.

### Returns

- The string beginning with **CONNERR** indicates a problem with the connection to the external system; when the connection is re-established, the function is called again, using the same input data

- Any other string, including a *null string*, indicates that the acknowledgement has been sent to the external system successfully

#### Additional Information

- This function is called only 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. The e\*Way executes this function if the Event's processing is not completed successfully by *all* the Collaborations to which it was sent; otherwise, the e\*Way executes the [Positive Acknowledgment Function](#).
- This function can return data to be queued, but the e\*Way will *not* acknowledge the data with an ACK or NAK.

**Note:** *If you configure the acknowledgment function to return a non-null string, you must configure a Collaboration (with appropriate IQs) to process the returned Event.*

---

## Shutdown Command Notification Function

### Description

The e\*Way calls this Monk function automatically to notify the external system that it is about to shut down. This function also can be used to shut down the connection with the external. The function accepts a string as input and must return a string.

### Required Values

The name of a Monk function or the name of a file containing a Monk function. There is no default value for this parameter.

**Note:** *This parameter is **required**, and must **not** be left blank.*

### Input

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.

### Returns

- 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

### Additional Information

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

---

## 7.5 HTTP Configuration

The parameters in this section furnish the required HTTP variables.

---

### Request

#### Description

Defines whether this request is to use the **GET** or **POST** method.

#### Required Values

Either **GET** or **POST**; the default is **POST**.

---

### Timeout

#### Description

Specifies the number of milliseconds to wait for a response when connecting to the Web server.

#### Required Values

A number from **1** through **864,000**; the default value is **50,000**.

---

### URL

#### Description

Specifies the URL to which the **GET** or **POST** command is sent.

#### Required Values

A full URL. The default value is:

```
http://www.seebeyond.com/eai/start.swe
```

**Note:** You must replace **<Web Server>** with a valid Web server name or IP address.

#### Additional Information

When using the **GET** command, you can provide parameters using the **x-www-form-urlencoded** notation, if required by the interfacing CGI program; for example:

```
http://www.johndoe.com/search?p1+fort&p2=john&p3=doe
```

---

### User Name

#### Description

Specifies the user name for authentication when connecting to an HTTP server.

### Required Values

A string containing any valid user name; the default is **anonymous**.

*Note:* You must define the User Name before setting the Encrypted Password.

---

## Encrypted Password

### Description

Specifies the password to be associated with the user name (previous parameter) for authentication when connecting to the HTTP server.

### Required Values

A valid password string.

*Note:* You must define the User Name before setting the Encrypted Password.

---

## Agent

### Description

Specifies the agent name to pass to the HTTP server. This is an arbitrary name identifying the e\*Way to the HTTP server.

### Required Values

A string; the default is **e\*Gate HTTP e\*Way**.

---

## Content-type

### Description

Specifies the content type of the application data.

### Required Values

The default is **Content-Type: application/x-www-form-urlencoded**.

### Additional Information

In most cases, the default value is correct and should not be changed.

*Note:* In previous releases of the HTTP e\*Way this was performed automatically. With this release it is necessary to call **http-url-encode**.

---

## Request-content

### Description

Specifies the request content when using the **POST** command.

### Required Values

A string. The expected string must follow the `stringx=string_data` format, for example:

```
p1=john&p2=doe
```

---

## Accept-type

### Description

Provides the parameters for the `accept-type` request header.

### Required Values

A string, for example:

```
accept:text/html,accept:text/plain,accept:text/*, ...
```



---

## 7.6 HTTP Proxy Configuration

The parameters in this section furnish the required HTTP Proxy variables.

---

### Use Proxy Server

#### Description

Specifies whether or not to use a proxy server.

#### Required Values

Yes or No; the default is No.

If this parameter is set to **Yes**, the e\*Way uses the parameter values in this section to connect through a proxy server.

If this parameter is set to **No**, the e\*Way assumes a direct connection.

---

### User Name

#### Description

The user name for authentication when connecting to a proxy server. This parameter is required by URLs that require “HTTP Basic Authentication” to access the site.

#### Required Values

A valid username string.

If **Use Proxy Server** is set to **No**, this parameter may be left blank.

*Note:* You must define the **User Name** before setting the **Encrypted Password**.

---

### Encrypted Password

#### Description

The password for authentication when connecting to a proxy server. This parameter is required by URLs that require “HTTP Basic Authentication” to access the site.

#### Required Values

A valid password string.

If **Use Proxy Server** is set to **No**, this parameter may be left blank.

*Note:* You must define the **User Name** before setting the **Encrypted Password**.

---

## Server Address

### Description

Specifies the URL of the proxy server to be used.

### Required Values

A proxy server URL, for example:

```
http://myproxy
```

If **Use Proxy Server** is set to **No**, this parameter may be left blank.

**Note:** Do not include a port number as part of the URL. This parameter is set separately (see next parameter).

---

## Port Number

### Description

Specifies the port number to which the proxy server is listening.

### Required Values

A number from 1 through 864,000; the default value is 8080.

---

## 7.7 HTTP SSL Configuration

The parameters in this section furnish the required Secure Sockets Layer (SSL) variables.

---

### Use SSL

#### Description

Specifies whether or not to use SSL.

#### Required Values

Yes or No; the default is No.

If this parameter is set to **Yes**, the e\*Way uses the parameter values in this section to configure the required certificate information.

If this parameter is set to **No**, the e\*Way ignores any certificate information.

---

### Trusted CA Certificates Directory

#### Description

The directory (within the repository) where all CA certificates are located. These certificates are used to verify a trust relationship between you and the CA.

#### Required Values

A valid directory name; the default value is **pkicerts/trustedcas**.

If **Use SSL** is set to **No**, this parameter may be left blank.

---

### Use Client Certificate Map

#### Description

A certificate map is a text file that maps a base URL to a client certificate file and a client private key file. This parameter specifies whether or not to use such a file, as identified by the [Client Certificate Map File](#) parameter.

#### Required Values

Yes or No; the default is No.

If this parameter is set to **Yes**, the e\*Way selects client certificates based on the provided certificate map.

If this parameter is set to **No**, the e\*Way ignores any certificate map.

If **Use SSL** is set to **No**, this parameter may be left blank.

---

## Client Certificate Map File

### Description

Specifies the location of the client certificate map to be used.

### Required Values

The directory and file name of the text file containing the client certificate map. The default value is `pkcerts/client/certmap.txt`.

If `Use SSL` is set to `No`, this parameter may be left blank.

### Additional Information

The certificate map file contains four columns, separated by a bar symbol (`|`). The respective columns contain the following information:

- base URL
- logical path and file name of client certificate to use
- logical path and file name of client key to use
- encoding type for certificate and key (ASN or PEM)

### Example

```
www.stc.com|pkicerts/client/certs/mycert1.cer|pkicerts/client/  
keys/mycert1.key|ASN
```

### Notes

- An asterisk (\*) in the first column, replacing the base URL, is interpreted to mean “all others”
- A hash symbol (#) in the first column causes the line to be treated as a comment
- Lines in the file are processed from top to bottom; the first base URL match found is used

---

## 7.8 PeopleSoft Application Messaging

The parameters in this section help you set up the required information for Publishing XML messages to PeopleSoft's Application Messaging Gateway using the PeopleSoft Application Messaging protocol.

---

### Base64 Deflate

#### Description

Enables or disables base64-encoded compression of XML messages. This compression is recommended to increase network throughput.

#### Required Values

Yes or No. The default is Yes.

Yes enables base64-encoded compression the <data> portion of the XML message to be posted to PeopleSoft 8. No disables compression, and the message is published as-is.

---

### Request Version

#### Description

Specifies the version of PeopleTools for which the XML messages are valid.

#### Required Values

Installed version of PeopleTools; the default value is 8.13.

---

### To Node

#### Description

Specifies the name of the local node on the receiving PeopleSoft system (the intended receiving node for messages sent to PeopleSoft 8).

#### Required Values

A valid node name; the default value is PSFT\_EP.

#### Additional Information

This parameter is also referred to as the *node definition*, and corresponds to an entry in the node lookup table on the gateway servlet.

---

## From Node

### Description

Specifies the node from which messages are sent to PeopleSoft 8. The node name must match the node definition for the third party system, as defined in the receiving PeopleSoft system

### Required Values

The node name for the sending system; the default is **STC**.

### Additional Information

The parameters **From Node**, **Channel**, and **Publication ID** uniquely identify the publication.

---

## Encrypted Password

### Description

Specifies the password associated with the destination node. The value is stored in the PeopleSoft database and must be communicated to the system administrators for the publishing system.

### Required Values

A valid password string; no default is set.

*Note:* This parameter is conditional—see below.

### Additional Information

If the node definition on the sending system has a node group defined, the password is present. If the node definition on the receiving system has a node group defined, the password *must* be present and *must* match the node group password.

---

## Channel

### Description

Specifies the name of the message channel containing the message.

### Required Value

A valid message channel name; the default value is **SOME\_MSG\_CHANNEL**, which must be replaced with the actual value.

### Additional Information

The parameters **From Node**, **Channel**, and **Publication ID** uniquely identify the publication.

---

## Subject

### Description

Specifies the name of the message, as defined in the PeopleSoft system.

### Required Values

A valid message name; the default value is **SOME\_MSG\_NAME**, which must be replaced with the actual value.

---

## Message Version

### Description

Specifies the message version. In publications containing multiple data versions, there are multiple occurrences of the data.

### Required Values

A valid message version name; the default value is **VERSION\_1**, which must be replaced with the actual value.

---

## Subject Detail

### Description

Specifies a subtype of a message name, when this information is required by the application.

### Required Values

A valid message-name subtype. There is no default value.

*Note:* This parameter is not generally required.

---

## Publication ID

### Description

Specifies the identifier for the publication.

### Required Values

A valid ID string. There is no default value.

*Note:* Specification of this parameter is optional, since it can be system-generated (see below).

### Additional Information

The parameters **From Node**, **Channel**, and **Publication ID** uniquely identify the publication. If the **From Node** value is specified, but this parameter is not, the

publication ID is set automatically to the next available publication ID on the specified channel within the subscribing PeopleSoft 8 database.

---

## Subchannel

### Description

Specifies the name of the subchannel containing the message.

### Required Values

This field should contain the concatenated values that represent the subchannel. For example, if the subchannel is:

```
Business Unit\Journal ID
```

then the value of this field is:

```
M04123456789
```

where Business Unit = M04 and Journal ID = 123456789.

*Note:* This parameter is conditional—see below.

### Additional Information

This parameter should be specified if a subchannel is defined in the subscribing PeopleSoft system; otherwise, it may be omitted.

Messages in the same channel but in different subchannels are assumed to refer to distinct objects—for example, different purchase orders or different employees. They are processed in parallel whenever possible.

---

## Originating Node

### Description

Specifies the name of the node that originally published the message. If not included in the XML file, the system sets it to the publishing node name.

### Required Values

Name of the node from which the message originated. There is no default value.

*Note:* This parameter is optional.

### Additional Information

The purpose of this parameter is to prevent circular publishing.



---

## Publisher

### Description

Specifies the operator ID (or class) that generated the message, if required by the receiving application.

### Required Values

An application-defined operator ID or class. There is no default value.

*Note: This parameter is conditional, and not required by the e\*Way.*

---

## Publication Process

### Description

Specifies the name of the program that generated the message, if required by the receiving application.

### Required Values

An application-defined name of the program that generated the message. There is no default value.

*Note: This parameter is conditional, and not required by the e\*Way.*

---

## Default Version

### Description

Identifies the default message version for the sending system, if required by the receiving application.

### Required Values

A string representing the default message version. There is no default value.

*Note: This parameter is conditional, and not required by the e\*Way.*

# API Functions

This chapter describes Monk functions and scripts installed by the PeopleSoft HTTP e\*Way. Refer to the *HTTP(S) e\*Way Intelligent Adapter User's Guide* for descriptions of the HTTP(S) e\*Way API functions, which are also used by this e\*Way.

---

## 8.1 Overview

The PeopleSoft HTTP e\*Way's functions fall into the following categories:

[Helper Functions](#) on page 123

[Standard e\\*Way Functions](#) on page 126

[Generic e\\*Way Functions](#) on page 133

## 8.2 Helper Functions

The auxiliary functions and scripts described in this section are called by the Standard e\*Way Functions to accomplish specific tasks such as pinging the PeopleSoft server and parsing responses. These functions consist of:

[psoft8-helper-ping](#) on page 123

[psoft8-helper-ping-response-isok](#) on page 123

[psoft8-helper-print-exception](#) on page 124

[psoft8-helper-pub-response-isok](#) on page 124

[psoft8-helper-wrap-xml](#) on page 125

---

### psoft8-helper-ping

#### Description

This function is called by [psoft8-verify](#), and sends an XML ping message to inform the e\*Way whether or not the PeopleSoft 8 server is up.

#### Signature

`(psoft8-helper-ping)`

#### Parameters

None.

#### Returns

A Boolean true (**#t**) if the PeopleSoft 8 server is up; a Boolean false (**#f**) if it is down.

#### Location

`psoft8-helper-ping.monk`

---

### psoft8-helper-ping-response-isok

#### Description

This transformation script is called by [psoft8-helper-ping](#). It parses the XML ping message to PeopleSoft 8 and retrieves the ping response message from PeopleSoft 8.

#### Signature

`(psoft8-helper-ping-response-isok message-string)`

#### Parameters

Name	Type	Description
message-string	string	The PeopleSoft 8 ping response XML message.

### Returns

A Boolean true (#t) if the PeopleSoft 8 ping response message has been retrieved; a Boolean false (#f) if it has not.

### Location

`psoft8-helper-ping-response-isok.tsc`

## psoft8-helper-print-exception

### Description

This transformation script parses exception messages from PeopleSoft 8 and displays them as a string.

### Signature

`(psoft8-helper-print-exception message-string)`

### Parameters

Name	Type	Description
message-string	string	The exception message to parse and display.

### Returns

The string "FAILURE" is returned if the script does not load during initialization.

### Location

`psoft8-helper-print-exception.tsc`

## psoft8-helper-pub-response-isok

### Description

This transformation script is called by [psoft8-outgoing](#), and helps determine whether or not the XML message was successfully published to the PeopleSoft application.

### Signature

`(psoft8-helper-pub-response-isok message-string)`

### Parameters

Name	Type	Description
message-string	string	The PeopleSoft 8 publication response XML message.

### Returns

Upon success, a Boolean true (#t); otherwise, a Boolean false (#f).

### Location

`psoft8-helper-pub-response-isok.tsc`

---

## psoft8-helper-wrap-xml

### Description

This function is called by [psoft8-outgoing](#), and wraps the XML Message with the PeopleSoft XML wrapper.

### Signature

`(psoft8-helper-wrap-xml psoft_xml_data)`

### Parameters

Name	Type	Description
psoft-xml-data	string	The PeopleSoft 8 XML message.

### Returns

None

### Location

`psoft8-helper-wrap-xml.monk`

## 8.3 Standard e\*Way Functions

The functions described in this section control the PeopleSoft HTTP e\*Way's communications center and are defined within the configuration file. None of these functions is available to Collaboration Rules scripts executed by the e\*Way.

The current set of standard Monk functions for the PeopleSoft HTTP e\*Way is:

- [psoft8-ack](#) on page 126
- [psoft8-connect](#) on page 127
- [psoft8-exchange](#) on page 127
- [psoft8-init](#) on page 128
- [psoft8-nack](#) on page 128
- [psoft8-notify](#) on page 129
- [psoft8-outgoing](#) on page 129
- [psoft8-shutdown](#) on page 130
- [psoft8-startup](#) on page 131
- [psoft8-verify](#) on page 131

---

### psoft8-ack

#### Description

Sends a positive acknowledgment to PeopleSoft 8, after all Collaborations to which the e\*Way sent data have processed and enqueued that data successfully.

#### Signature

`(psoft8-ack message-string)`

#### Parameters

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

#### Returns

- An empty string indicates a successful operation, after which the e\*Way can proceed with the next request.
- The string "CONNERR" indicates that a problem occurred with the connection to PeopleSoft. When the connection is re-established, the function is called again.

#### Location

`psoft8-ack.monk`

#### See also

[Positive Acknowledgment Function](#) on page 107.

---

## psoft8-connect

### Description

Establishes a connection to PeopleSoft 8.

### Signature

(psoft8-connect)

### Parameters

None.

### Returns

The string “UP” indicates that the connection has been established; anything else indicates failure to connect.

### Throws

None.

### Location

psoft8-connect.monk

### See also

[External Connection Establishment Function](#) on page 106.

---

## psoft8-exchange

### Description

Sends an Event received from PeopleSoft. The function expects no input.

### Signature

(psoft8-exchange)

### Parameters

None.

### Returns

- An empty string indicates a successful operation, and nothing is sent to e\*Gate.
- A message-string indicates a successful operation, and the Event is sent to e\*Gate.
- The string “CONNERR” indicates that a problem occurred with the connection to PeopleSoft. When the connection is re-established, this function is re-executed with the same input Event.

### Throws

None.

### Location

psoft8-exchange.monk

See also

[Exchange Data with External Function](#) on page 105.

---

## psoft8-init

### Description

Begins the initialization process for the e\*Way. This function loads the file `stc_monkpsft8.dll` and the initialization file, making the function scripts available for use.

### Signature

`(psoft8-init)`

### Parameters

None.

### Returns

If the string `"FAILURE"` is returned, the e\*Way shuts down; any other return indicates success.

### Throws

None.

### Additional Information

Any global variables needed by the function scripts can be defined within this function.

### Location

`psoft8-init.monk`

See also

[Monk Environment Initialization File](#) on page 103.

---

## psoft8-nack

### Description

Sends a negative acknowledgment to PeopleSoft when the e\*Way fails to process and enqueue Events from the external system.

### Signature

`(psoft8-nack message-string)`

### Parameters

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



### Returns

- An empty string indicates a successful operation.
- The string “CONNERR” indicates that a problem occurred with the connection to PeopleSoft. When the connection is re-established, the function is called again.

### Throws

None.

### Location

`psoft8-nack.monk`

### See also

[Negative Acknowledgment Function](#) on page 108.

## psoft8-notify

### Description

Notifies PeopleSoft that the e\*Way is shutting down.

### Signature

`(psoft8-notify command)`

### Parameters

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

### Returns

Returns a null string.

### Throws

None.

### Location

`psoft8-notify.monk`

### See also

[Shutdown Command Notification Function](#) on page 109.

## psoft8-outgoing

### Description

Used for sending a received message from e\*Gate to PeopleSoft 8.

### Signature

`(psoft8-outgoing event-string)`

## Parameters

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

## Returns

- An empty string indicates a successful operation.
- The string "RESEND" causes the Event to be resent immediately.
- The string "CONNERR" indicates that a problem occurred with the connection to PeopleSoft. When the connection is re-established this function is re-executed with the same input Event.
- The string "DATAERR" indicates the function had a problem processing data. If the e\*Gate journal is enabled, the Event is recorded and the failed Event count is increased. (The input Event is essentially skipped in this process.) Use the [event-send-to-egate](#) function to place "bad" events in a "bad Event" queue.

## Location

psoft8-outgoing.monk

## See also

[Process Outgoing Message Function](#) on page 104.

## psoft8-shutdown

### Description

Requests that the connection to PeopleSoft shut down. A return value of "SUCCESS" indicates that the shutdown can occur immediately. Any other return value indicates that the shutdown Event must be delayed. You must then execute a [shutdown-request](#) call from within a Monk function to allow the requested shutdown process to continue.

### Signature

(psoft8-shutdown *shutdown*)

### Parameters

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

### Returns

The string "SUCCESS" allows an immediate shutdown to occur. Anything else delays shutdown until the [shutdown-request](#) call is executed successfully.

### Throws

None.

## Location

`psoft8-shutdown.monk`

## See also

[External Connection Shutdown Function](#) on page 107.

## psoft8-startup

### Description

Used for function loads that are specific to this e\*Way, and invokes startup.

### Signature

`(psoft8-startup)`

### Parameters

None.

### Returns

The string “FAILURE” causes shutdown of the e\*Way. Any other value indicates success.

### Throws

None.

### Additional Information

This function should be used to initialize PeopleSoft before data exchange starts. Any additional variables can be defined here.

### Location

`psoft8-startup.monk`

### See also

[Startup Function](#) on page 104.

## psoft8-verify

### Description

Used to verify whether or not the connection to PeopleSoft 8 has been established.

### Signature

`(psoft8-verify)`

### Parameters

None.

### Returns

The string “UP” or “SUCCESS” if the connection has been established. Any other value indicates that the connection was not established.

### Throws

None.

### Location

`psoft8-verify.monk`

### See also

[External Connection Verification Function](#) on page 106.

## 8.4 Generic e\*Way Functions

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

The current set of basic Monk functions is:

- [event-commit-to-egate](#) on page 133
- [event-rollback-to-egate](#) on page 134
- [event-send-to-egate](#) on page 134
- [event-send-to-egate-ignore-shutdown](#) on page 135
- [event-send-to-egate-no-commit](#) on page 135
- [get-logical-name](#) on page 136
- [insert-exchange-data-event](#) on page 136
- [send-external-up](#) on page 137
- [send-external-down](#) on page 137
- [shutdown-request](#) on page 138
- [start-schedule](#) on page 138
- [stop-schedule](#) on page 139
- [waiting-to-shutdown](#) on page 139

---

### event-commit-to-egate

#### Description

Commits the Event sent previously to the e\*Gate system using [event-send-to-egate-no-commit](#).

#### Signature

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

#### Parameters

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

#### Returns

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

#### Throws

None.

## event-rollback-to-egate

### Description

Rolls back the Event sent previously to the e\*Gate system using [event-send-to-egate-no-commit](#), following receipt of a rollback command from the external system.

### Signature

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

### Parameters

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

### Returns

Boolean true (**#t**) if the data is rolled back successfully; otherwise, false (**#f**).

### Throws

None.

## event-send-to-egate

### Description

Sends data that the e\*Way has already received from the external system into the e\*Gate system as an Event.

### Signature

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

### Parameters

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

### Returns

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

### Throws

None.

### Additional information

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

See also

[event-send-to-egate-ignore-shutdown](#) on page 135

[event-send-to-egate-no-commit](#) on page 135

## event-send-to-egate-ignore-shutdown

### Description

Sends data that the e\*Way has already received from the external system into the e\*Gate system as an Event—but ignores any pending shutdown issues.

### Signature

(event-send-to-egate-ignore-shutdown *string*)

### Parameters

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

### Returns

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

### Throws

None.

See also

[event-send-to-egate](#) on page 134

[event-send-to-egate-no-commit](#) on page 135

## event-send-to-egate-no-commit

### Description

Sends data that the e\*Way has received from the external system to the e\*Gate system as an Event—but without Committing, pending confirmation from the external system of correct transmission of the data.

### Signature

(event-send-to-egate-no-commit *string*)

### Parameters

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

### Returns

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

### Throws

None.

### See also

[event-commit-to-egate](#) on page 133

[event-rollback-to-egate](#) on page 134

[event-send-to-egate](#) on page 134

[event-send-to-egate-ignore-shutdown](#) on page 135

---

## get-logical-name

### Description

Returns the logical name of the e\*Way.

### Signature

(get-logical-name)

### Parameters

None.

### Returns

The name of the e\*Way (as defined by the e\*Gate Schema Designer).

### Throws

None.

---

## insert-exchange-data-event

### Description

While the [Exchange Data with External Function](#) is still active, this function can be called to initiate a repeat call to it—whether or not data was queued to e\*Gate via the function's return mechanism following the initial call.

### Signature

(insert-exchange-data-event)

### Parameters

None.

### Returns

None.



### Throws

None.

### See also

[Exchange Data Interval](#) on page 99

[Zero Wait Between Successful Exchanges](#) on page 99

---

## send-external-up

### Description

Informs the e\*Way that the connection to the external system is up.

### Signature

(send-external-up)

### Parameters

None.

### Returns

None.

### Throws

None.

---

## send-external-down

### Description

Informs the e\*Way that the connection to the external system is down.

### Signature

(send-external-down)

### Parameters

None.

### Returns

None.

### Throws

None.

---

## shutdown-request

### Description

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**. Once this function is called, shutdown proceeds immediately.

### Signature

(shutdown-request)

### Parameters

None.

### Returns

None.

### Throws

None.

### Additional Information

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.

---

## start-schedule

### Description

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.

### Signature

(start-schedule)

### Parameters

None.

### Returns

None.

### Throws

None.

---

## stop-schedule

### Description

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 effect any defined schedules, and does not halt the e\*Way process itself.

### Signature

(stop-schedule)

### Parameters

None.

### Returns

None.

### Throws

None.

---

## waiting-to-shutdown

### Description

Informs the external application that a shutdown command has been issued.

### Signature

(waiting-to-shutdown)

### Parameters

None.

### Returns

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

### Throws

None.

# MUX Subscription Handler

---

## A.1 Java Classes

The MUX subscription handler consists of ten java classes, which are contained in the `stcph.jar` file. This file is available when the e\*Gate participating host is installed.

*Note: The `MuxPublicationHandler` class depends on the MUX Java Client classes, which also are contained in the `stcph.jar` file.*

---

### Entry.class

#### Description

Holds the following information for an instance of a MUX subscription handler:

- Node Name
  - MUX Host
  - MUX Port
  - MUX Expire
  - MUX Timeout
  - Uncompress?
  - Base64-Decode?
  - Include Headers?
  - Log File
- 

### MuxHandlerConstants.class

#### Description

Contains constant values for the MUX handler package such as the name of the configuration file to store the parameters persistently.

---

## MuxHandlerEntry.class

### Description

Maintains a collection of Entry classes. This class Loads and parses the configuration file in order to load the data into memory. It also saves changes to the values for the Entry classes to the configuration file.

---

## AdministerMuxHandler.class

### Description

Displays the MUX administration root page for administering the MUX subscription handler.

---

## AdministerMuxHandlerAddMode.class

### Description

Displays the MUX administration page for adding a MUX subscription handler.

---

## AdministerMuxHandlerDeleteMode.class

### Description

Displays the MUX administration page for deleting a MUX subscription handler.

---

## AdministerMuxHandlerEditMode.class

### Description

Displays the MUX administration Page for editing the configuration values of a MUX subscription handler.

---

## AdministerMuxHandlerError.class

### Description

Displays the error page when an error occurs while administering the MUX subscription handler.

---

## MuxPublicationHandler.class

### Description

Handles the publishing of XML messages from PeopleSoft to the MUX e\*Way using the MUX Java Client APIs.

---

## MuxHandler.class

### Description

Loaded by the Application Messaging Gateway and serves as an entry point to the MUX subscription handler. It loads any pre-existing MUX subscription handlers.

# Index

## A

Accept-type parameter 112  
 Additional Path parameter 102  
 Agent parameter 111  
 AIX 15  
 APIs - see functions, Monk  
 Assigning ETDs to Event Types 46  
 Autorun 16

## B

Base64 Deflate parameter 117

## C

Changing the User Name 81  
 Channel parameter 118  
 Client Certificate Map File parameter 116  
 Collaboration 47, 85, 88  
   Rules 47, 85, 86, 88  
   Rules Editor 88  
   Service 85  
 Components, e\*Way 12  
 configuration  
   Communication Setup 99–101  
   General Settings 97–98  
   HTTP Configuration 110–112  
   HTTP Proxy Configuration 113–114  
   HTTP SSL Configuration 115–116  
   Monk Configuration 102–109  
   PeopleSoft Application Messaging 117–121  
 configuration files, default  
   stcewpsoft8.def 12  
 configuration parameters  
   Accept-type 112  
   Additional Path 102  
   Agent 111  
   Base64 Deflate 117  
   Channel 118  
   Client Certificate Map File 116  
   Content-type 111  
   Default Version 121  
   Down Timeout 101  
   Encrypted Password 111, 113, 118

Exchange Data Interval 99  
 Exchange Data With External Function 105  
 External Connection Establishment Function 106  
 External Connection Shutdown Function 107  
 External Connection Verification Function 106  
 Forward External Errors 98  
 From Node 118  
 Journal File Name 97  
 Max Failed Messages 97  
 Max Resends Per Message 97  
 Message Version 119  
 Monk Environment Initialization File 103  
 Negative Acknowledgment Function 108  
 Originating Node 120  
 Port Number 114  
 Positive Acknowledgement Function 107  
 Process Outgoing Message Function 104  
 Publication ID 119  
 Publication Process 121  
 Publisher 121  
 Request 110  
 Request Version 117  
 Request-content 111  
 Resend Timeout 101  
 Server Address 114  
 Shutdown Command Notification Function 109  
 Start Exchange Data Schedule 100  
 Startup Function 104  
 Stop Exchange Data Schedule 101  
 Subchannel 120  
 Subject 119  
 Subject Detail 119  
 Timeout 110  
 To Node 117  
 Trusted CA Certificates Directory 115  
 Up Timeout 101  
 URL 110  
 Use Client Certificate Map 115  
 Use Proxy Server 113  
 Use SSL 115  
 User Name (HTTP Proxy) 113  
 User Name (HTTP) 110  
 Zero Wait Between Successful Exchanges 99  
 configuration procedures 77  
 Content-type parameter 111  
 conventions, writing in document 9  
 Creating an e\*Way 75

## D

default configuration file 12  
 Default Version parameter 121  
 Down Timeout parameter 101

## E

## e\*Way

- Components 12
- configuration 77
- creating 75
- Installation 16
- Properties 76
- Schedules 81
- Startup Options 81
- troubleshooting 85

## Editor

- Collaboration Rules 88

Encrypted Password parameter 111, 113, 118

Event Type 46

Event Type Definition (ETD) 46

event-commit-to-egate function 133

event-rollback-to-egate function 134

Events 87

event-send-to-egate function 134

event-send-to-egate-ignore-shutdown function 135

event-send-to-egate-no-commit function 135

Exchange Data Interval parameter 99

Exchange Data with External Function parameter 105

executable files

- stcewgenericjava.exe 12
- stcewipmp.exe 12

External Connection Establishment Function parameter 106

External Connection Shutdown Function parameter 107

External Connection Verification Function parameter 106

## F

Forward External Errors parameter 98

From Node parameter 118

functions (see also functions, Monk)

- Generic 133–139
- Helper 123–125
- Standard 126–132

functions, Monk

- event-commit-to-egate 133
- event-rollback-to-egate 134
- event-send-to-egate 134
- event-send-to-egate-ignore-shutdown 135
- event-send-to-egate-no-commit 135
- get-logical-name 136
- insert-exchange-data-event 136
- psoft8-ack 126
- psoft8-connect 127
- psoft8-exchange 127

- psoft8-helper-ping 123, 124
- psoft8-helper-ping-response-isok 123
- psoft8-helper-pub-response-isok 124
- psoft8-helper-wrap-xml 125
- psoft8-init 128
- psoft8-nack 128
- psoft8-notify 129
- psoft8-outgoing 129
- psoft8-shutdown 130
- psoft8-startup 131
- psoft8-verify 131
- send-external down 137
- send-external-up 137
- shutdown-request 138
- start-schedule 138
- stop-schedule 139
- waiting-to-shutdown 139

## G

Generic e\*Way Functions 133–139

get-logical-name function 136

## H

Helper Functions 123–125

## I

insert-exchange-data-event function 136

Installation procedure

- e\*Way (UNIX) 20
- e\*Way (Windows) 16
- MUX Handler Classes (UNIX) 24
- MUX Handler Classes (Windows) 22

InstallShield 16

Intelligent Queue (IQ) 48, 85

## J

Journal File Name parameter 97

## L

Load Path, Monk 102

logging options 83

## M

Max Failed Messages parameter 97

Max Resends Per Message parameter 97

Message Version parameter 119

monitoring thresholds 84



Monk Configuration  
 Load Path 102  
 Specifying File Names 102  
 Specifying Function Names 102  
 Specifying Multiple Directories 102  
 Monk Environment Initialization File parameter 103  
 MUX Handler Classes  
 Description 140  
 Installation (UNIX) 24  
 Installation (Windows) 22  
 Setup in PeopleSoft 70

## N

Negative Acknowledgment Function parameter 108

## O

Oracle 15  
 Originating Node parameter 120

## P

Participating Host 85  
 PeopleSoft 8 Native Functions 123  
 Port Number parameter 114  
 Positive Acknowledgment Function parameter 107  
 procedures  
 configuration 77  
 installation 16  
 Process Outgoing Message Function parameter 104  
 Properties, e\*Way 76  
 psoft8-ack function 126  
 psoft8-connect function 127  
 psoft8-exchange function 127  
 psoft8-helper-ping function 123, 124  
 psoft8-helper-ping-response-isok function 123  
 psoft8-helper-pub-response-isok function 124  
 psoft8-helper-wrap-xml function 125  
 psoft8-init function 128  
 psoft8-nack function 128  
 psoft8-notify function 129  
 psoft8-outgoing function 129  
 psoft8-shutdown function 130  
 psoft8-startup function 131  
 psoft8-verify function 131  
 Publication ID parameter 119  
 Publication Process parameter 121  
 Publisher parameter 121

## Q

Queues 48

## R

Request parameter 110  
 Request Version parameter 117  
 Request-content parameter 111  
 Resend Timeout parameter 101

## S

Schedules 81  
 send-external-down function 137  
 send-external-up function 137  
 Server Address parameter 114  
 Setting Startup Options or Schedules 81  
 Shutdown Command Notification Function parameter 109  
 shutdown-request function 138  
 Standard e\*Way Functions 126–132  
 Start Exchange Data Schedule parameter 100  
 start-schedule function 138  
 Startup Function parameter 104  
 Startup Options 81  
 stcewgenericjava.exe file 12  
 stcewipmp.exe file 12  
 Stop Exchange Data Schedule parameter 101  
 stop-schedule function 139  
 Subchannel parameter 120  
 Subject Detail parameter 119  
 Subject parameter 119

## T

Timeout parameter 110  
 To Node parameter 117  
 troubleshooting the e\*Way 85  
 Trusted CA Certificates Directory parameter 115

## U

UNIX installation  
 e\*Way 20  
 MUX Handler 24  
 Up Timeout parameter 101  
 URL parameter 110  
 Use Client Certificate Map parameter 115  
 Use Proxy Server parameter 113  
 Use SSL parameter 115  
 User name 81  
 User Name parameter (HTTP Proxy) 113  
 User Name parameter (HTTP) 110

## W

waiting-to-shutdown function 139

## Index

Windows installation

e\*Way 16

MUX Handler 22

## Z

Zero Wait Between Successful Exchanges parameter

99