SeeBeyond ICAN Suite

e*Way Intelligent Adapter for PeopleSoft Message Agent User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

SEEBEYOND®

SeeBeyond Proprietary and Confidential

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 This work is protected as an unpublished work under the copyright laws.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 20050406030611.

Contents

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:

- Windows 2000 and/or UNIX operations and administration
- Windows-style GUI operations
- PeopleSoft Message Agent concepts and operations
- Integrating PeopleSoft Message Agent applications with external systems

P.2 Organization

This User's Guide is organized roughly into two parts. The first part, consisting of Chapters 1-4, 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. Chapter 3 also contains descriptions of the sample schemas provided with the product. These can be used to test your system following installation and, if appropriate, as templates you can modify to produce your own custom schemas. 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 5-7, 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 Message Agent is frequently referred to as the PeopleSoft Message Agent 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.

```
stcregutil -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*.

Chapter 1

Introduction

1.1 Overview

The SeeBeyond e*Way Intelligent Adapter for PeopleSoft Message Agent enables the e*Gate Integrator system to exchange data with PeopleSoft applications. It joins the PeopleSoft environment with the e*Gate environment by connecting to the PeopleSoft Application Server as a client, and communicating with the PeopleSoft Message Agent using Message Agent APIs. This e*Way provides transactionally-safe, enterprise-wide, bidirectional connectivity to PeopleSoft on all platforms.

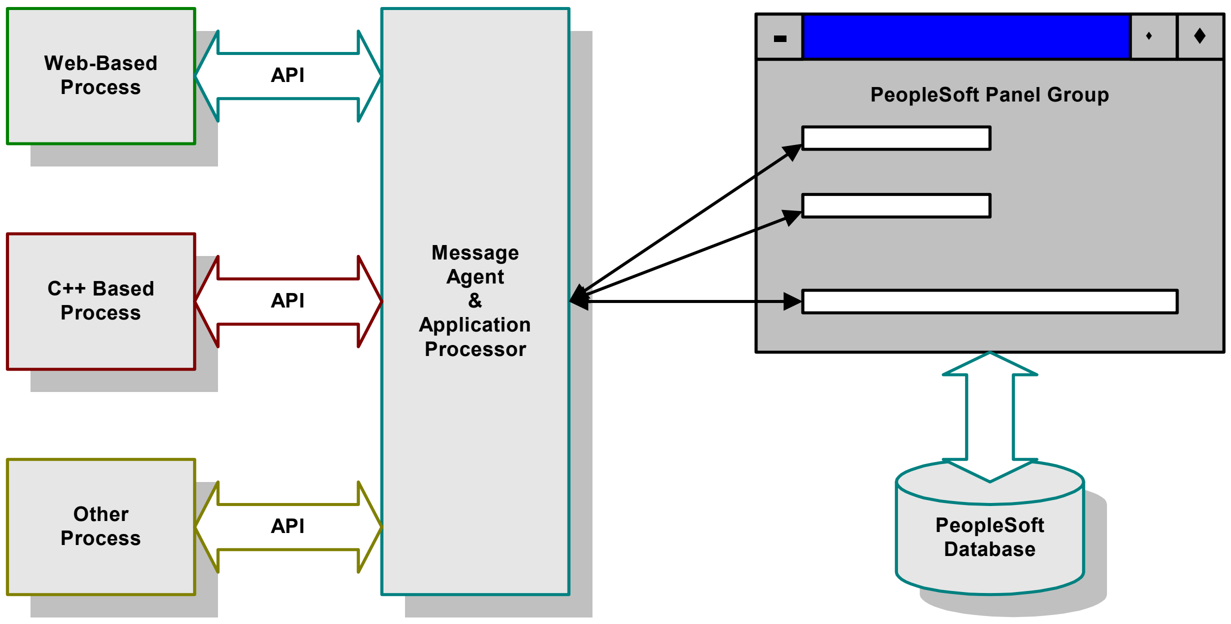
1.2 PeopleSoft Message Agent

The PeopleSoft Message Agent provides a means for third-party applications to access PeopleSoft panels, the imbedded business logic, and the associated data. In doing so, the Message Agent extends the reach of PeopleSoft business processes beyond the PeopleSoft applications themselves. It allows users throughout an organization to use applications other than PeopleSoft to process information into PeopleSoft panels. For example, the Message Agent can be used to submit a change of address, to enroll in a course, to issue a PTO balance inquiry, and so on.

Rather than entering or manipulating data directly into a PeopleSoft panel, users throughout the organization may enter the data using software applications familiar to them, such as a web application, C++ based tools, etc. The process is diagrammed in the following figure.

Access to PeopleSoft panels is accomplished through the use of a set of Application Programming Interfaces (APIs), in conjunction with a message definition. Using the Message Agent APIs, you can write programs that can reuse the business logic behind the PeopleSoft panel groups.

Figure 1 PeopleSoft Message Agent Process



To use the Message Agent you must have the following:

1. To allow other programs to be able to access a PeopleSoft panel, a message definition must be created using the Application Designer. The message definition creates the mapping between a set of fields in the message definition and the fields on a PeopleSoft panel. It also has various control information when accessing the panel.
2. A Message Agent client program must be written using the Message Agent APIs. The APIs are a set of programming interfaces that allow a user to write a program to access a panel through the message definition - the Message Agent client. These programs are created using the e*Gate Integrator GUI and Monk coding.
3. The Message Agent APIs allow the creation of a program that can emulate the actions of a user accessing the PeopleSoft application panels directly. For example, a program can be written to provide the keys to a panel just as an on-line user would through the search dialog—or retrieve data from the panel fields, or enter data into the panel fields—and save the panel.
4. PeopleSoft performs the same edits and security checks that an on-line user would use, and runs any business logic associated with the panel. If the panel has Workflow business logic associated with it, the Message Agent also triggers the business event.
5. Third, and finally, the Message Agent server is part of the PeopleSoft application server and runs as its own application server process. It is administered just like the other PeopleSoft application servers.

With the PeopleSoft 7.5 release, the Message Agent has been converted to run as a part of the PeopleSoft 3-tier architecture. The Message Agent now runs on the PeopleSoft Application Server as another server, PSAPISRV.

For more information on PeopleSoft three-tier configuration and the application server, see the PeopleSoft 7.5 (or later) PeopleBooks documentation.

1.3 e*Way Operation

1.3.1 e*Gate to PeopleSoft (Event Driven)

An inbound event-driven interface to PeopleSoft involves the real-time or near-real-time processing of a single, individual transaction triggered by a user or business event in the source application. Examples include the creation of a customer sales order or a modification of an employee's personal data. Data affected by these events in the source application are passed to e*Gate for translation, and are loaded into PeopleSoft by the PeopleSoft Message Agent e*Way (see **Figure 3** on page 15).

The main components of an inbound event-driven interface include:

- A source-specific e*Way, to receive the application data
- e*Gate Integrator, to transform and route the application data
- A PeopleSoft Message Agent e*Way, to load the transformed data into the PeopleSoft system
- An error-handling Files e*Way, to process failed transactions

1.3.2 e*Gate to PeopleSoft (Batch Mode)

An inbound batch interface involves the loading of a collection of transactions usually in the form of a data file into PeopleSoft. The interface can be triggered by a user or application event in the source system or by a scheduling mechanism. Examples include a nightly synchronization of inventory information with an external system or a weekly load of timecard information into PeopleSoft Payroll.

The interface data file from the source application is passed to e*Gate Integrator for translation and loaded into PeopleSoft via the Message Agent (see **Figure 4** on page 17). The main components of this inbound batch interface are identical to those in the event-driven implementation (above).

Typically, batch interfaces into PeopleSoft involve the direct insertion of data into the PeopleSoft application tables. This design, on the other hand, utilizes the PeopleSoft Message Agent, which handles the PeopleCode processing and data validation.

The architecture of this interface is a scaled-up extension of the inbound event-driven interface. For both interfaces, the Message Agent is used to load the data into PeopleSoft. However, since the batch interface is designed to process a higher volume of transactions—along with multiple PeopleSoft application servers and Message Agent processes—multiple PeopleSoft Message Agent e*Ways may be required.

Figure 2 e*Gate-to-PeopleSoft Event Driven Architecture

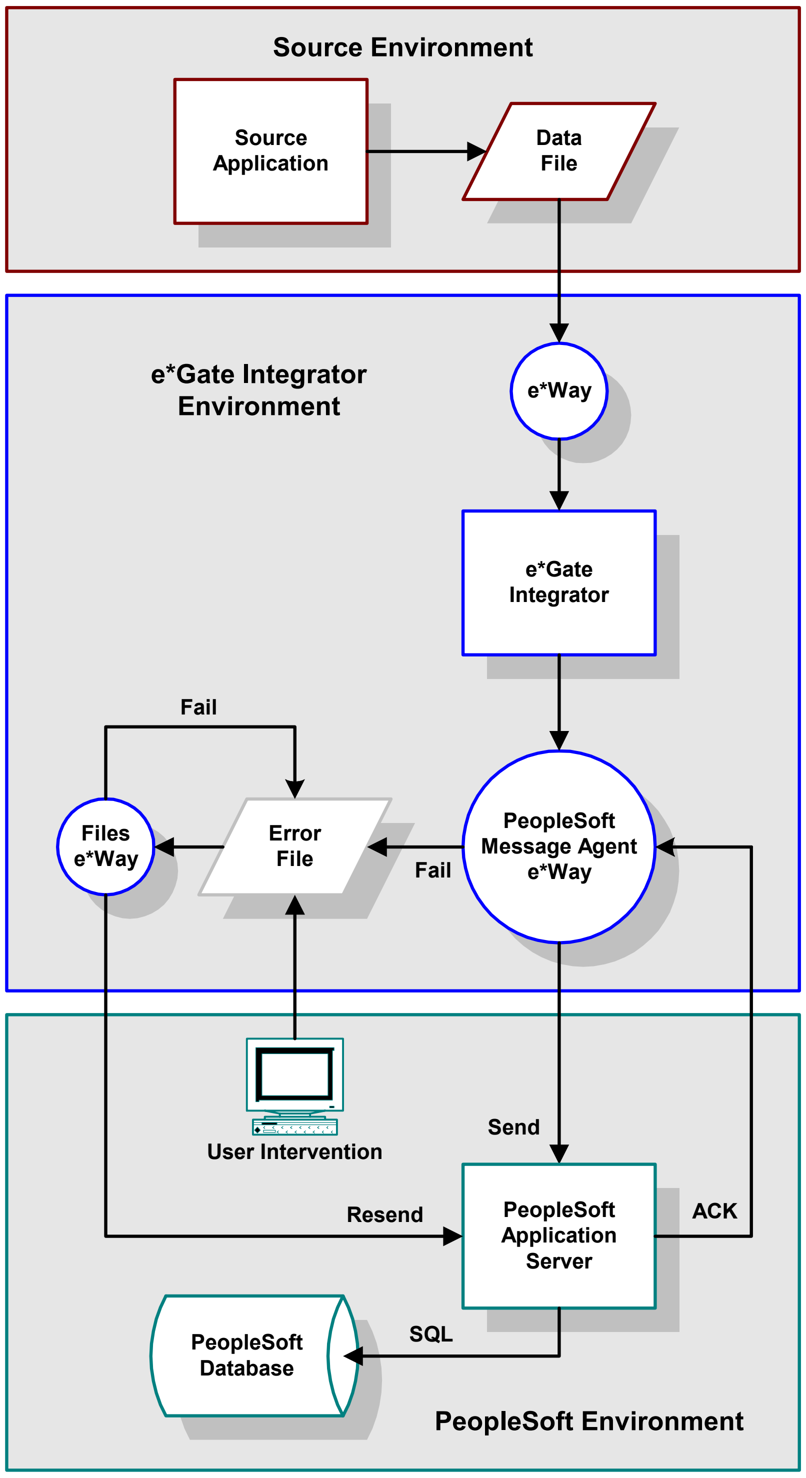
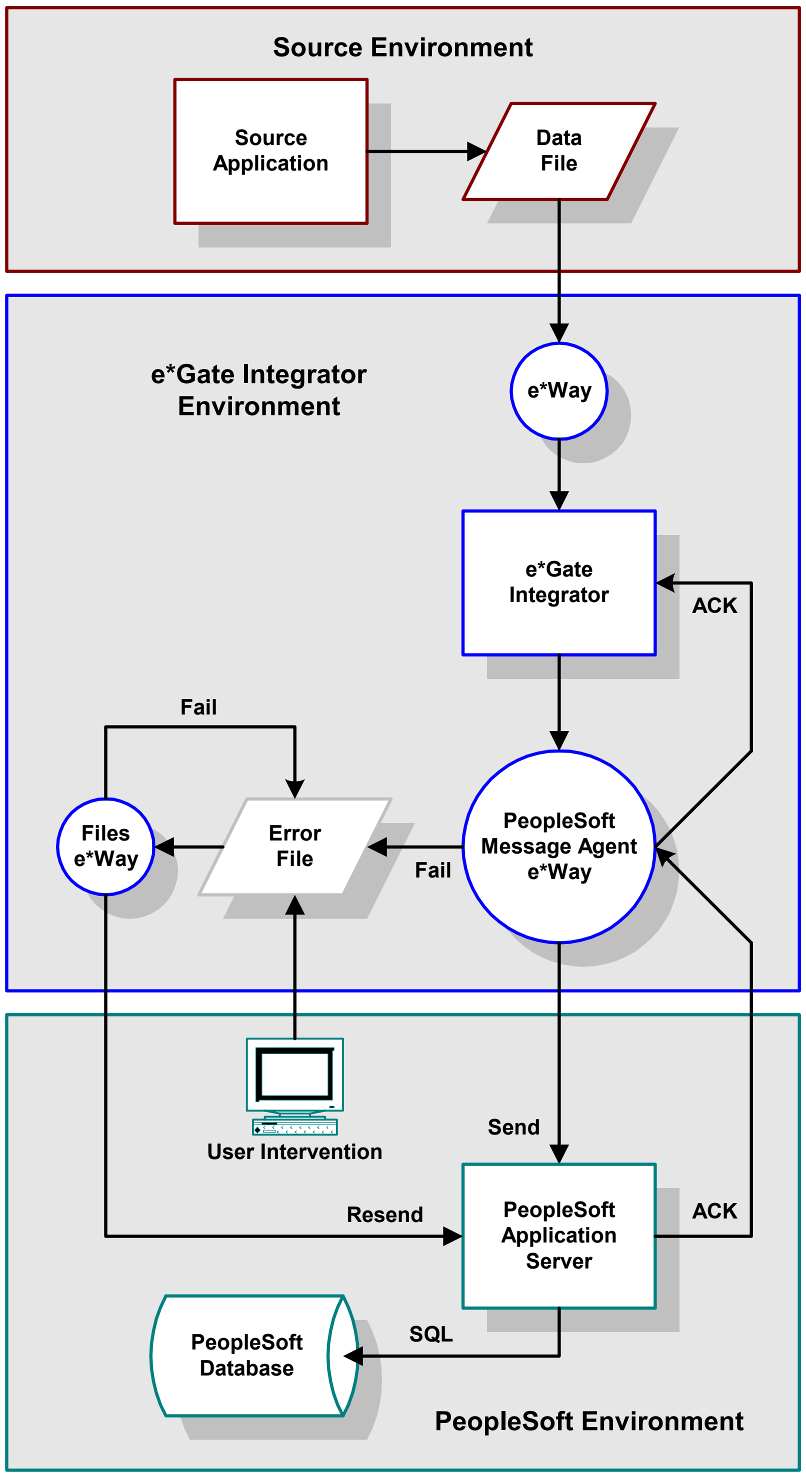


Figure 3 e*Gate-to-PeopleSoft Batch Mode Architecture



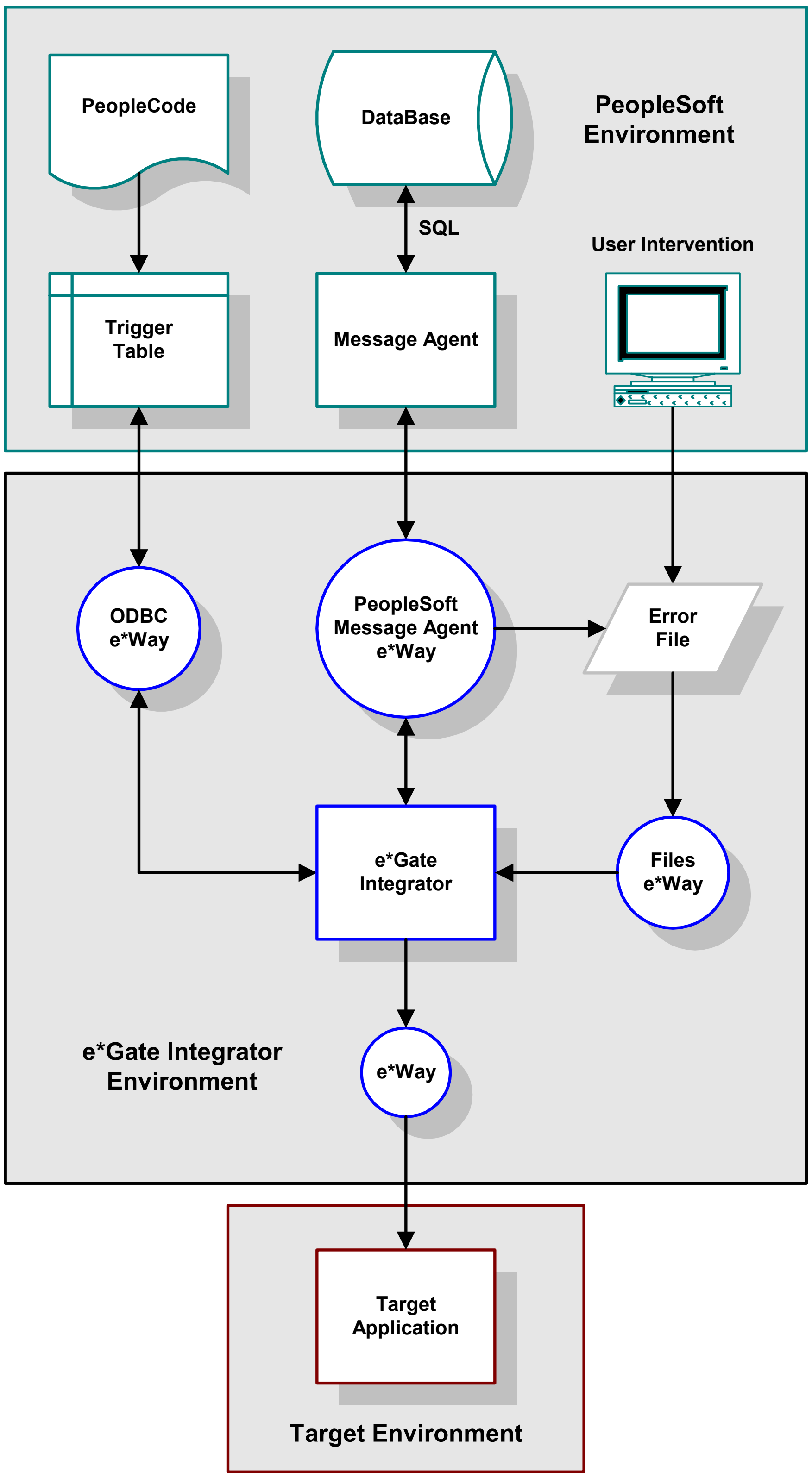
1.3.3 PeopleSoft to e*Gate (Event Driven)

An outbound event-driven interface from PeopleSoft involves the real-time or near-real-time processing of a single, individual transaction triggered by a user or application business event. Examples include the creation of a customer sales order or a modification of an employee's personal data.

These events in the PeopleSoft source system produce a trigger which e*Gate Integrator reads, transforms, and loads into the target system (see **Figure 4** on page 17). The main components of an outbound event-driven interface include:

- An ODBC e*Way, to retrieve a PeopleCode trigger from the source system
- A PeopleSoft Message Agent e*Way, to retrieve application data
- e*Gate Integrator, to transform the application data
- A target-system-specific e*Way, to load the transformed data into the target system

Figure 4 PeopleSoft-to-e*Gate Overview



1.4 e*Way Components

The PeopleSoft Message Agent e*Way incorporates the following components:

- Executable files
  - **stcewgenericmonk.exe**
  - **stcpsconvert.exe**
- Default configuration file
  - **psoft.def**
- Dynamic load libraries
  - **stc_ewpsoft.dll**
- Monk Library files

For a list of installed files, see **Chapter 2**.

1.5 Supported Operating Systems

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

The e*Way Intelligent Adapter for PeopleSoft Message Agent is available on the following operating systems:

- Windows 2000 and Windows Server 2003
- Sun Solaris 8

Chapter 2

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\ewpsoft 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 Message Agent, 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 140 KB on Windows systems
    - Approximately 370 KB on Solaris 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.*

2.1.1 External System Requirements

The e*Way Intelligent Adapter for PeopleSoft Message Agent supports the following applications:

- PeopleSoft 7.5
- PeopleTools 7.5.3 (a *full version* is required)

2.2 Environment Configuration

2.2.1 Participating Host

Only one change is required to the operating environment and the e*Gate system to support the PeopleSoft Message Agent e*Way. To your library path, add the PeopleSoft application **.dll** files, usually in**:**

```
pt750/client/bin/<platform>
```

where **platform** is your platform name (such as Win 32 or Solaris 26).

2.2.2 External Configuration Requirements

On UNIX platforms only, after installing the PeopleSoft Message Agent e*Way, the following needs to be added to the Load Library Path*:

```
$PS_HOME/API
```

where **$PS_HOME** refers to the home directory of the PeopleSoft client installation.

*As an example, the Load Library Path on Solaris is:

```
$LD_LIBRARY_PATH
```

Note: *PeopleSoft message definitions must be defined in PeopleSoft before the e*Way can access the system.*

2.3 Installing the e*Way

2.3.1 Windows Systems

Installation Procedure

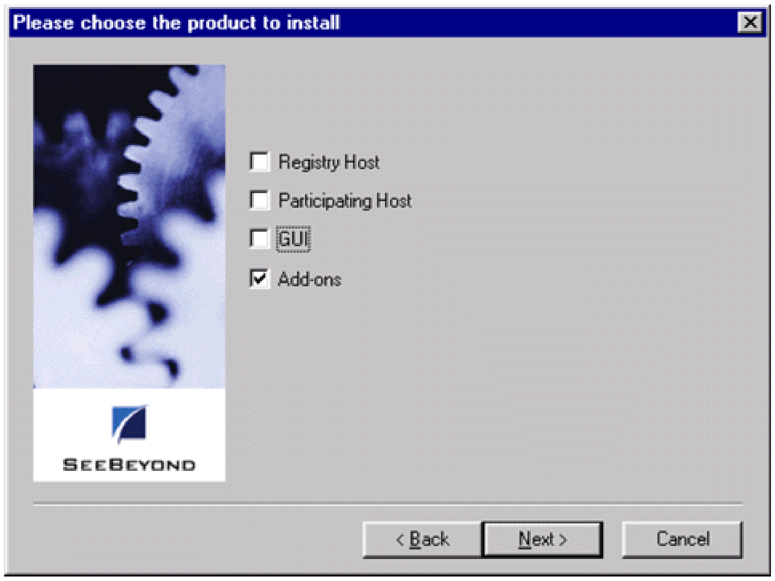
Note: *The installation utility detects and suggests the appropriate installation directory. Use this directory unless advised otherwise by SeeBeyond.*

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 5). Check **Add-ons**, then click **Next**.

Figure 5 Choose Product Dialog



Please choose the product to install

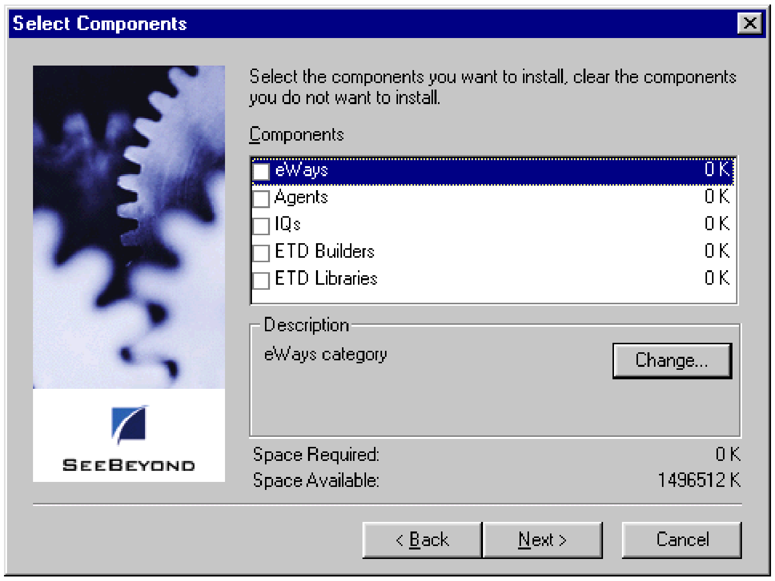
☐ Registry Host
☐ Participating Host
☐ GUI
☑ Add-ons

   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 6). Highlight—*but do not check*—**eWays** and then click **Change**.

Figure 6 Select Components Dialog



Select the components you want to install, clear the components you do not want to install.

Components

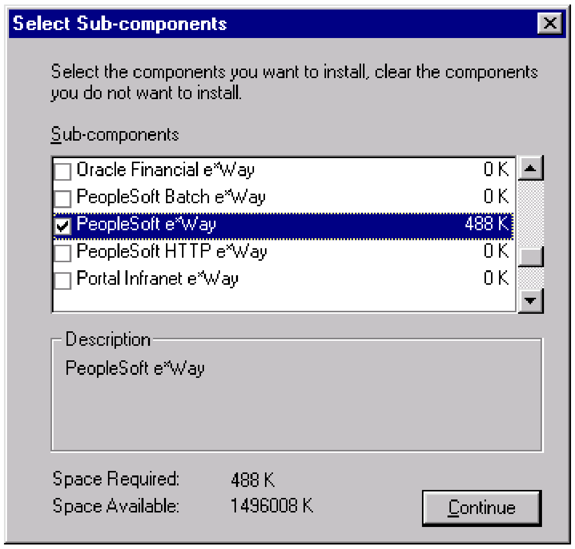
☐ eWays	0 K
☐ Agents	0 K
☐ IQs	0 K
☐ ETD Builders	0 K
☐ ETD Libraries	0 K

Description
eWays category

Space Required: 0 K
Space Available: 1496512 K

6 When the Select Sub-components dialog box appears (see Figure 7), check the **PeopleSoft e*Way**.

Figure 7 Select Sub-components Dialog



Select the components you want to install, clear the components you do not want to install.

Sub-components

☐ Oracle Financial e*Way	0 K
☐ PeopleSoft Batch e*Way	0 K
✓ PeopleSoft e*Way	488 K
☐ PeopleSoft HTTP e*Way	0 K
☐ Portal Infranet e*Way	0 K

Description
PeopleSoft e*Way

Space Required: 488 K
Space Available: 1496008 K

7 Click **Continue**, and the Select Components dialog box reappears.

8 Click **Next** and continue with the installation.

Subdirectories and Files

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

By default, the InstallShield installer 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 1 Participating Host & Registry Host

Subdirectories	Files
\bin\	stc_ewpsoft.dll stcpsconvert.exe
\configs\stcewgenericmonk\	psoft.def psoft3.6to4.1Rule.txt
\monk_library\	psoft.gui
\monk_library\psoft\	ps.monk psmsg-delete.monk psmsg-getrecords.monk psmsg-init.monk psmsg-readrows.monk psmsg-setrecords.monk psmsg-updaterows.monk sap_to_ps_util.monk
\stcgui\ctls\	guipsoft.ctl

By default, the InstallShield installer also installs the following files within the **\eGate\Server\registry\repository\default** tree on the Registry Host.

Table 2 Registry Host Only

Subdirectories	Files
\	stcewpsoft.ctl

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, with several options. Select the **Install e*Way** option, and follow any additional on-screen instructions.

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.*

Subdirectories and Files

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

The preceding installation procedure 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 3 Participating Host & Registry Host

Subdirectories	Files
/bin/	stc_ewpsoft.dll
/configs/stcewgenericmonk/	psoft.def psoft3.6to4.1Rule.txt
/monk_library/	psoft.gui

Table 3 Participating Host & Registry Host

Subdirectories	Files
/monk_library/psoft/	ps.monk psmsg-delete.monk psmsg-getrecords.monk psmsg-init.monk psmsg-readrows.monk psmsg-setrecords.monk psmsg-updaterows.monk sap_to_ps_util.monk
/stcgui/ctls/	guipsoft.ctl

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

Table 4 Registry Host Only

Subdirectories	Files
/	stcewpsoft.ctl

Environment Configuration

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

2.4 Installed Files

The PeopleSoft Message Agent e*Way installation process creates the following subdirectories and installs the following files within the **\eGate\client** tree on the Participating Host. These files are then committed to the *default* schema on the Registry Host.

Note that the directory paths are shown using Windows conventions. Files and directories are identical in a UNIX installation, except for the executable file extension (**.exe**).

Table 5 Installed Subdirectories and Files

Subdirectories	Files
\bin\	stc_ewpsoft.dll stcewgenericmonk.exe stcewpsoft.ctl stcpsconvert.exe
\configs\stcewgenericmonk\	psoft.def psoft3.6to4.1Rule.txt
\monk_library\	psoft.gui
\monk_library\psoft\	ps.monk psmsg-delete.monk psmsg-getrecords.monk psmsg-init.monk psmsg-readrows.monk psmsg-setrecords.monk psmsg-updaterows.monk sap_to_ps_util.monk
\stcgui\ctls\	guipsoft.ctl

2.5 Optional Example Files

The installation CD-ROM contains two sample schemas, **ps_inbound_event** and **ps_outbound_event**, located in the **samples\ewpsoft** directory. To use a schema, you must load it onto your system using the following procedure. See **Sample Schema** on page 42 for descriptions of the sample schema and instructions regarding its use.

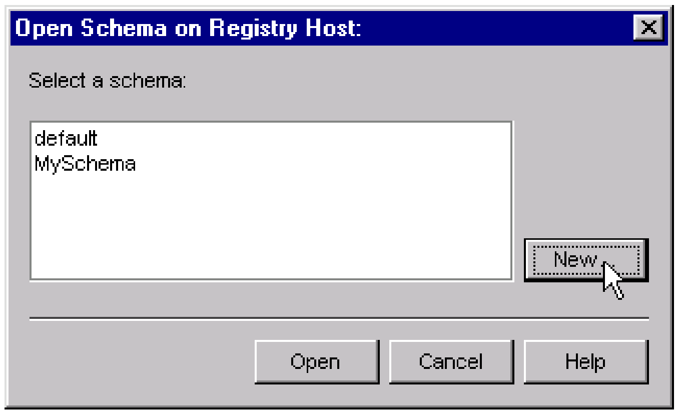
Note: *The PeopleSoft Message Agent e*Way must be properly installed on your system before you can run the sample schema.*

2.5.1 Installation Procedure

To load a sample schema

1 Invoke the **Open Schema** dialog box and select **New** (see Figure 8).

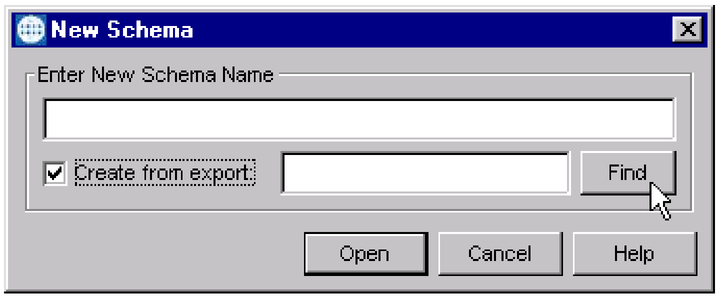
Figure 8 Open Schema Dialog



2 Type the name you want to give to the schema (for example, **inbound.Sample**)

3 Select **Create from export** and navigate to the directory containing the sample schema by clicking the **Find** button (see Figure 9).

Figure 9 New Schema Dialog



4 Select the desired archive file (***.zip**) and click **Open**.

Note: *The schema installs with the host name* **localhost** *and control broker name* **localhost_cb**. *If you want to assign your own names, copy the file* ***.zip** *to a local directory and extract the files. Using a text editor, edit the file* ***.exp**, *replacing all instances of the name* **localhost** *with your desired name. Add the edited* **.exp** *file back into the* **.zip** *file.*

2.5.2 Subdirectories and Files

The preceding procedure creates the following subdirectories and installs the following files within the **\eGate\Server\registry\repository\<SchemaName>** tree on the Registry Host, where **<SchemaName>** is the name you have assigned to the schema in step 2.

Table 6 Subdirectories and Files - ps_inbound_event

Subdirectories	Files
\	ps_inbound_event.ctl
\runtime\configs\stcewfile\	feeder.cfg feeder.sc
\runtime\configs\stcewgenericmonk\	PsPoster.cfg PsPoster.sc
\runtime\data\	PSinput.dat
\runtime\monk_scripts\common\	JD_Level0.ssc JD_WorkLocationEtc.ssc NEW_HIRE.ssc PD_Level0.ssc PD_NameAddress.ssc PD_NationalID.ssc PD_OtherPhones.ssc PsPoster-connect.monk PsPoster-finish.monk PsPoster-post.tsc PsPoster-startup.monk PsPoster-verify.monk sap_data.ssc SS_HR_ENTRY.ssc
\runtime\monk_scripts\common\datamap\	Action.map Action.map.swp ActionReasonHIR_Code.map ActionReasonTER_Code.map ActionReasonXFR_Code.map Business_Unit.map Country_Code.map Employee_RegTemp.map Employee_Status.map Employee_Type.map Form_Of_Address.map Gender.map Language_Code.map Marital_Status.map National_ID_Type.map Phone_Type.map State_DEU.map State_GBR.map State_MEX.map

Table 7 Subdirectories and Files - ps_outbound_event

Subdirectories	Files
\	ps_outbound_event.ctl
\runtime\configs\stcewfile\	eater.cfg eater.sc
\runtime\configs\stcewgenericmonk\	PsPollster.cfg PsPollster.cfg PsTrigger.cfg PsTrigger.cfg
\runtime\data\	output0.dat
\runtime\monk_scripts\common\	db_poll_init.monk EventPersJobUpdate.ssc HRMD_A02_PS.ssc JD_Level0.ssc JD_WorkLocationEtc.ssc newhire_to_hrmda02.tsc PD_Level0.ssc PD_NameAddress.ssc PD_NationalID.ssc PD_OtherPhones.ssc PSEvents.ssc psmsg-db-bind.monk psmsg-db-event.monk psmsg-events-from-queue.monk PsPollster-poll.tsc PsPollster-startup.monk PsPollster-verify.monk PSTrigger.dsc pstrigger.mrk PsTrigger-db.monk PsTrigger-events-from-queue.monk PsTrigger-startup.monk SS_PERS_JOB_UPDATE.ssc
\runtime\monk_scripts\common\data map\	ActionReasonHIR_Code.map ActionReasonPAY_Code.map ActionReasonTER_Code.map ActionReasonXFR_Code.map Bus_Area.map Cost_Center.map Country_Code.map Language_Code.map

Chapter 3

System Implementation

In this chapter we describe the procedures for implementing a working system incorporating the SeeBeyond e*Way Intelligent Adapter for PeopleSoft Message Agent. Please see the *e*Gate Integrator User's Guide* for additional information.

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 33

Creating Event Types on page 34

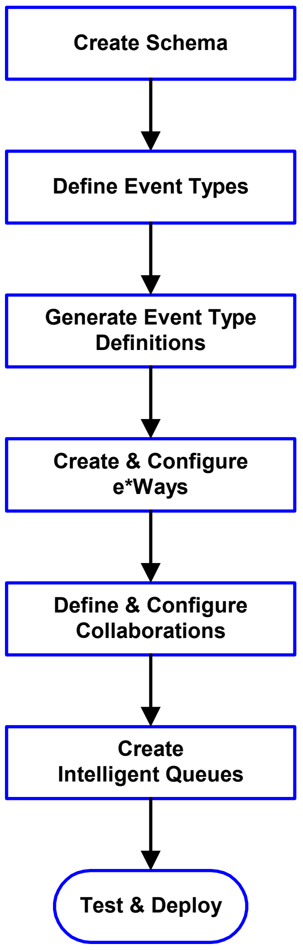
Generating Event Type Definitions on page 35

Defining Collaborations on page 40

Creating Intelligent Queues on page 41

Sample Schema on page 42

3.1.1 Implementation Sequence


Create Schema
Define Event Types
Generate Event Type Definitions
Create & Configure e*Ways
Define & Configure Collaborations
Create Intelligent Queues
Test & Deploy

1 The first step is to create a new Schema—the subsequent steps apply only to this Schema (see **Creating a Schema** on page 33).

2 The second step is to define the Event Types you are transporting and processing within the Schema (see **Creating Event Types** on page 34).

3 Third, 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 35).

4 The fourth step is to create and configure the required e*Ways (see **Chapter 4**).

5 Next is to define and configure the Collaborations linking the Event Types from step 2 (see **Defining Collaborations** on page 40).

6 Now you need to create Intelligent Queues to hold published Events (see **Creating Intelligent Queues** on page 41

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

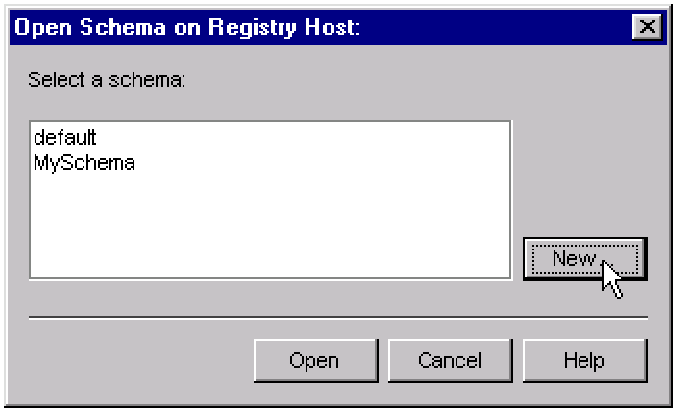
A schema is the structure that defines e*Gate system parameters and the relationships between components within the e*Gate system. Schemas can span multiple hosts.

All setup and configuration operations take place within an e*Gate schema. Therefore, 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

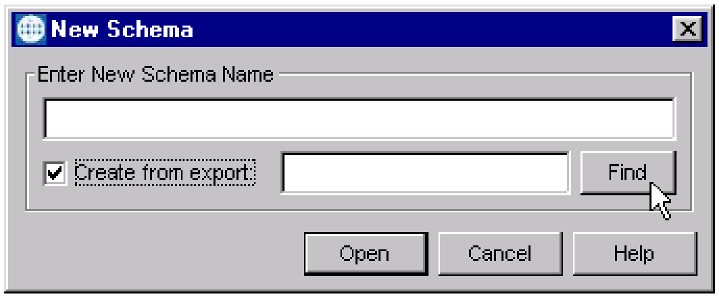
1 Invoke the **Open Schema** dialog box and **Open** an existing schema or click **New** to create a new schema.

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

3.4.1 Using the ETD Editor's Build Tool

The Event Type Definition Editor's Build tool automatically creates an Event Type Definition file based upon sample data. Use this procedure to create an Event Type Definition based upon the data your installation requires.

Note: *Be sure to set the Default Editor to* **Monk**, *from the* **Options** *menu in the e*Gate Schema Designer.*

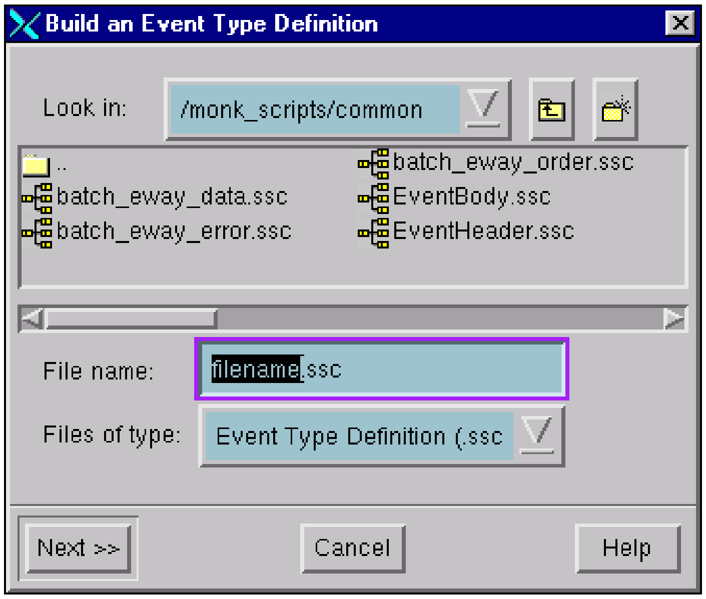
To create an Event Type Definition using the Build tool

1 Launch the ETD Editor by clicking in the e*Gate Schema Designer tool bar.

2 On the ETD Editor's tool bar, click **Build**.

The *Build an Event Type Definition* dialog box opens.

Figure 12 Build Event Type Definition Dialog

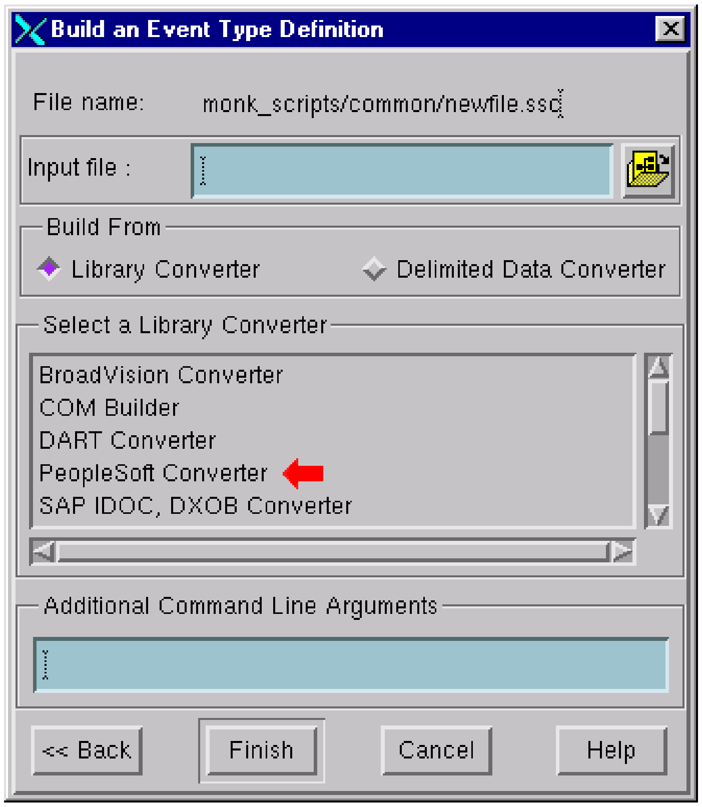


3 In the *File name* box, type the name of the ETD file you want to build.

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

4 Click **Next**. A new dialog box appears, as shown in Figure 13.

Figure 13 Building the ETD Dialog



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

6 Under *Select a Library Converter*, select **PeopleSoft Converter**.

7 In the *Additional Command Line Arguments* box, type any additional arguments, if desired.

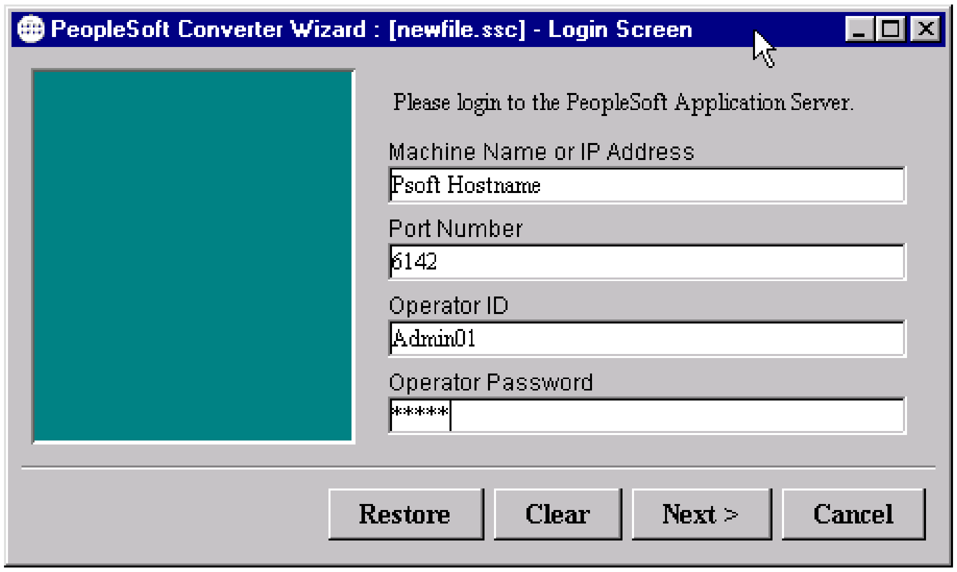
8 Click **Finish**, and the PeopleSoft Converter Wizard appears.

9 Follow the Wizard's instructions to finish building the ETD file.

The PeopleSoft Converter Wizard

The PeopleSoft Converter Wizard consists of a series of two dialog boxes, which ask for specific parameters. They are the *Login Screen* and the *Select Activity Name and Message Definition* dialog boxes.

Figure 14 Converter Wizard - Login Screen Dialog



Please login to the PeopleSoft Application Server.

Machine Name or IP Address: Psoft Hostname

Port Number: 6142

Operator ID: Admin01

Operator Password: *****

1 Enter the following information when the *Login Screen* dialog box opens:

Machine Name or IP Address—host name or IP address of the PeopleSoft application host.

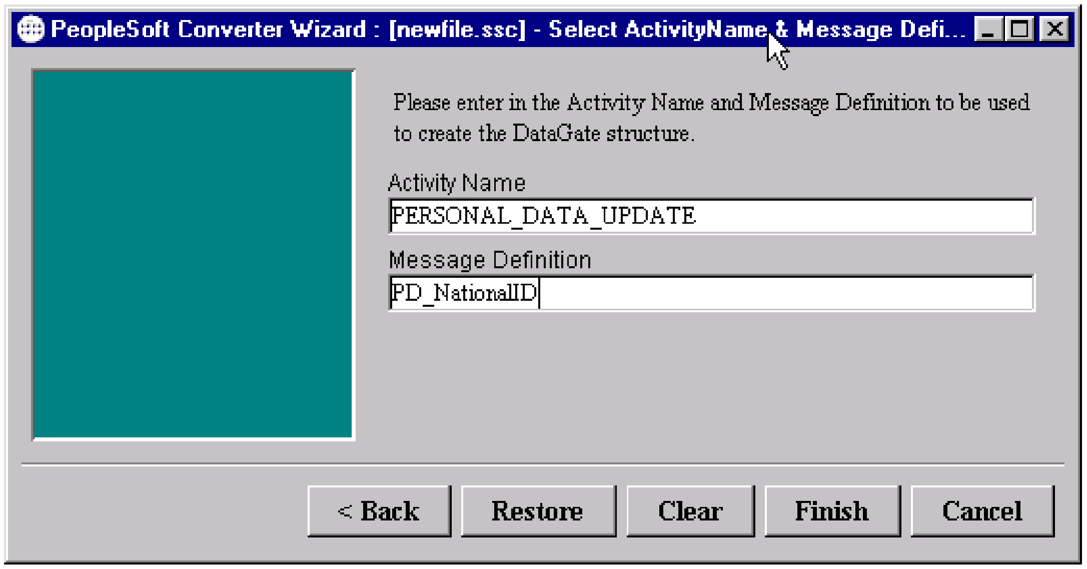
Port Number —communication port number.

Operator ID—your login name.

Operator Password—your login password.

2 After entering the information, click **Next**. The *Select Activity Name and Message Definition* dialog box appears.

Figure 15 Converter Wizard - Select Activity Name and Message Definition Dialog


PeopleSoft Converter Wizard : [newfile.ssc] - Select ActivityName & Message Defi...
< Back
Restore
Clear
Finish
Cancel

Please enter in the Activity Name and Message Definition to be used to create the DataGate structure.

Activity Name: PERSONAL_DATA_UPDATE

Message Definition: PD_NationalID

3. Enter the requested Activity Name and Message Definition.

   Activity Name—the name of the PeopleSoft Activity.

   Message Definition—the name of the PeopleSoft Message Definition to be used to generate the e*Gate Event Type Definition.

4. Click **Finish**.

The e*Gate Event Type Definition is then generated automatically and displayed in the ETD Editor.

3.4.2 Assigning ETDs to Event Types

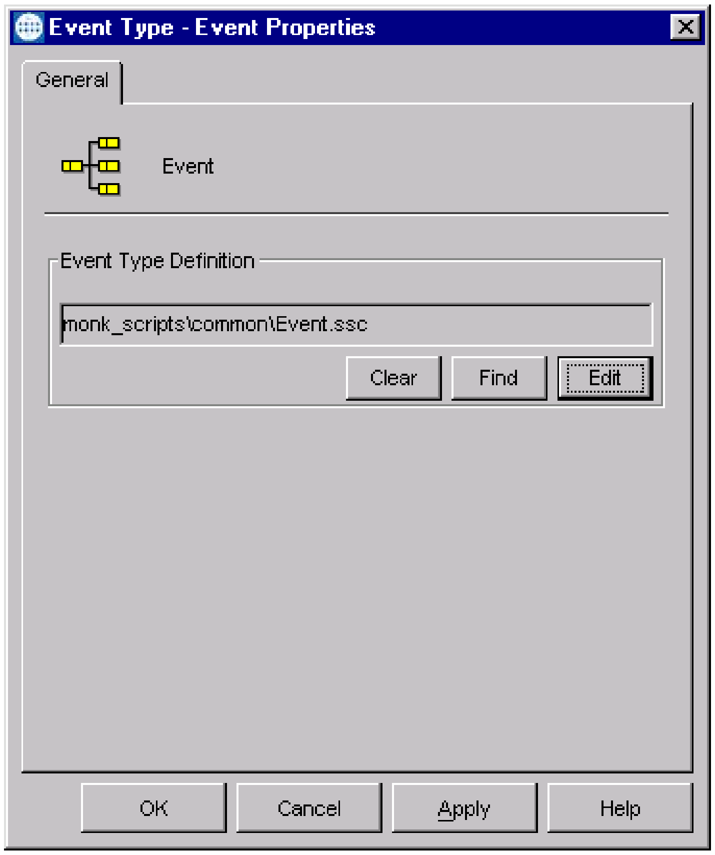
After you have created the e*Gate system's ETD files, you can assign them to Event Types you have already created.

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 16.

Figure 16 Event Type Properties Dialog Box



monk_scripts\common\Event.ssc

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.

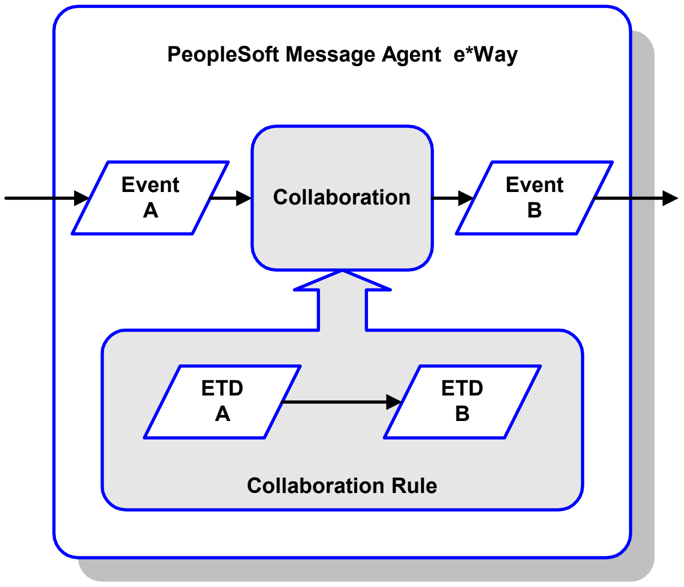
Each Event Type is now associated with the specified Event Type Definition.

3.5 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 17 Collaborations



The Collaboration is driven by a Collaboration Rule, which defines the relationship between the incoming and outgoing ETDs. You can use an existing Collaboration Rule, 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**.

3.6 Creating Intelligent Queues

The final step is to create and associate an IQ for the PeopleSoft Message Agent 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.

3.7 Sample Schema

The previous sections provided the basics for implementing a system employing the PeopleSoft Message Agent e*Way. This section describes the supplied sample schema **ps_inbound_event**, which provides an illustrative example. See **Optional Example Files** on page 27 for installation instructions.

3.7.1 Inbound Event

This sample implementation reads an input file, applies associated Collaborations and Rules, and then saves the results in a specified location. The schema consists of the following components:

e*Ways

- **Feeder**

  This e*Way reads inbound data, and performs the Collaboration that copies the data to an Intelligent Queue.

- **PsPoster**

  This e*Way transforms the data and copies it to a PeopleSoft database.

Intelligent Queue

- **SapDataQ**

  This Intelligent Queue is a standard e*Gate IQ.

Event Type

- **inputdata**

  This Event Type processes inbound data.

Collaboration Rule

- **InputdataToinputdata**

  This Collaboration Rule is associated with the **inputdata** Event Type, and passes data through the e*Ways.

Collaborations

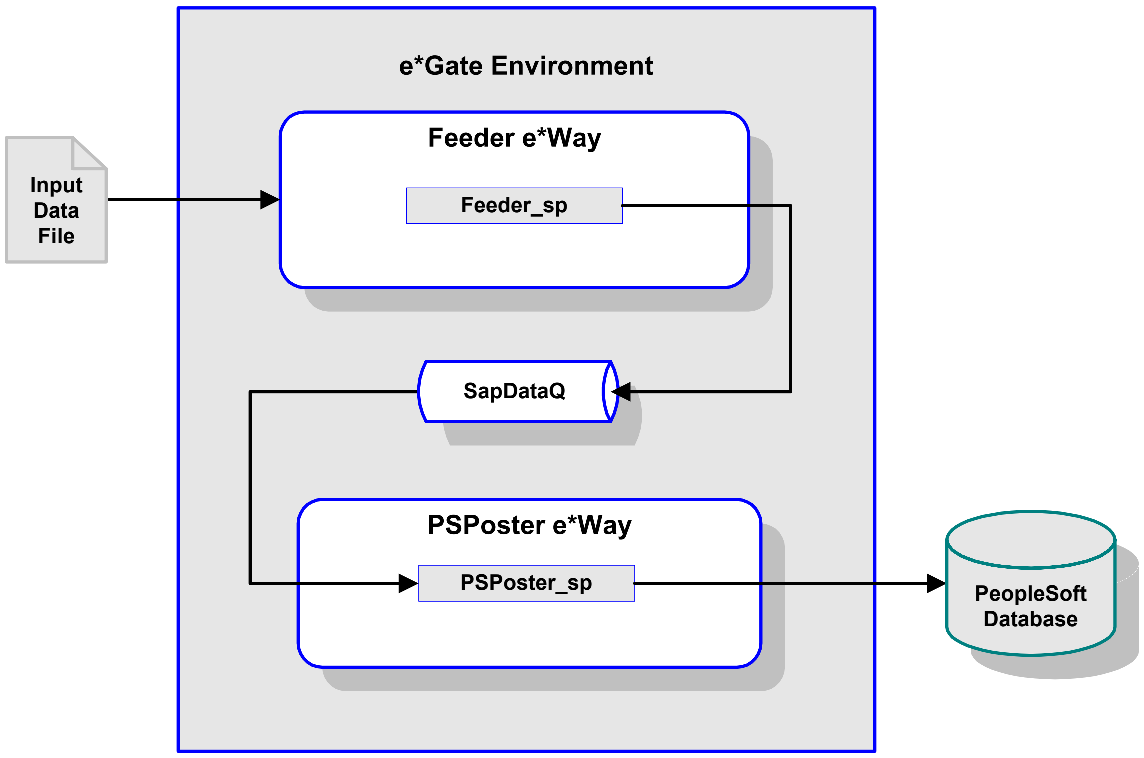
- **PsPoster_sp**

  This Collaboration is a component of the **PsPoster** e*Way, applies the **InputdataToinputdata** Rule, subscribes to the **inputdata** Event Type with the **Feeder_sp** Collaboration as its source, and publishes to a PeopleSoft database.

- **Feeder_sp**

  This Collaboration is a component of the **Feeder** e*Way, applies the **InputdataToinputdata** Rule, subscribes to the **inputdata** Event Type with an external source, and publishes to the **SapDataQ** Intelligent Queue.

Figure 18 Sample Schema Component View



It is assumed that the database is installed and configured properly. The sample schema also serves to verify that the PeopleSoft Message Agent e*Way has been properly installed and configured.

Table 8 Configuration Settings for ewPS_Inbound e*Way

Section	Parameter and Settings
General Settings	
AllowIncoming	Yes
AllowOutgoing	No
Outbound Settings	
(all)	Default
Poller Inbound Settings	
PollDirectory	/egate/data(input file folder)
InputFileExtension	*.dat (input file extension)
PollMilliseconds	Default
Remove EOL	Default
MultipleRecordsPerFile	Default
MaxBytesPerLine	Default
BytesPerLineIsFixed	Default

Table 9 Settings for ewPS_Poster e*Way

Section	Parameter and Setting
General Settings	
(all)	Default
Communication Setup	
Start Exchange Data Schedule	blank
Stop Exchange Data Schedule	blank
Exchange Data Interval	120
Down Timeout	Default
Up Timeout	Default
Resend timeout	Default
Zero Wait between successful Exchanges	Default
Monk Configuration	
Additional Path	Blank
Auxiliary Library Directories	monk_library/psoft
Monk Environment Initialization File	blank
Startup Function	monk_scripts/common/psPoster-startup.monk
Process Outgoing Message Function	monk_scripts/common/psPoster-post.tsc
Exchange Data With External Function	blank
External Connection Establishment Function	monk_scripts/common/PsPoster-connect.monk
External Connection Verification Function	monk_scripts/common/PsPoster-verify.monk
External Connection Shutdown Function	blank
Positive Acknowledgment Function	blank
Negative Acknowledgment Function	blank
Shutdown command notification function	monk_scripts/common/PsPoster-finish.monk
PeopleSoft Setup	
Host name or IP Address of Application Server:	(enter appropriate value)
Port number for the Application Server	(enter appropriate value)
Operator ID	(enter appropriate value)
Operator Password	(enter appropriate value)
Log file for failed records	(enter appropriate value)
Max. No. of failures before erroring out	(enter appropriate value)

Chapter 4

Setup Procedures

This chapter summarizes the setup procedures for the SeeBeyond e*Way Intelligent Adapter for PeopleSoft Message Agent.

4.1 Overview

After creating a schema, you must instantiate and configure the PeopleSoft Message Agent 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:

4.2 Setting Up the e*Way

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

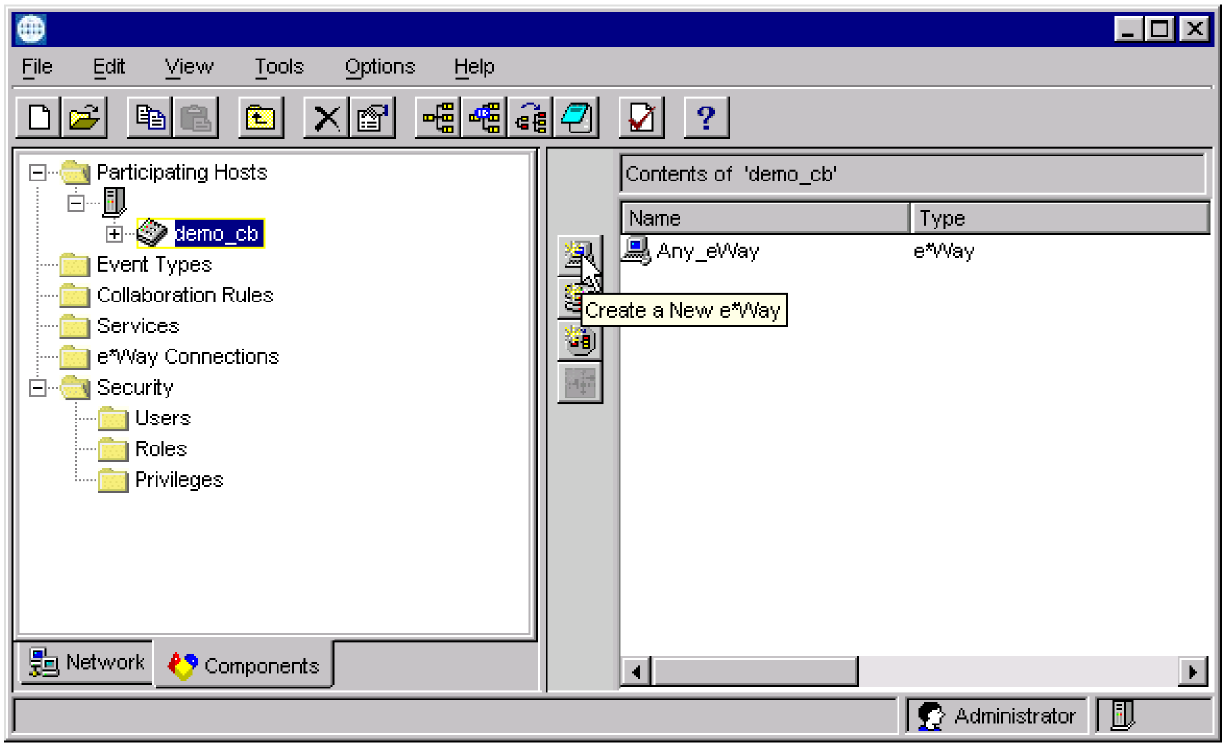
4.2.1 Creating the e*Way

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 19 e*Gate Schema Designer Window (Components View)



Contents of 'demo_cb'

Name	Type
Any_eWay	e*Way

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.

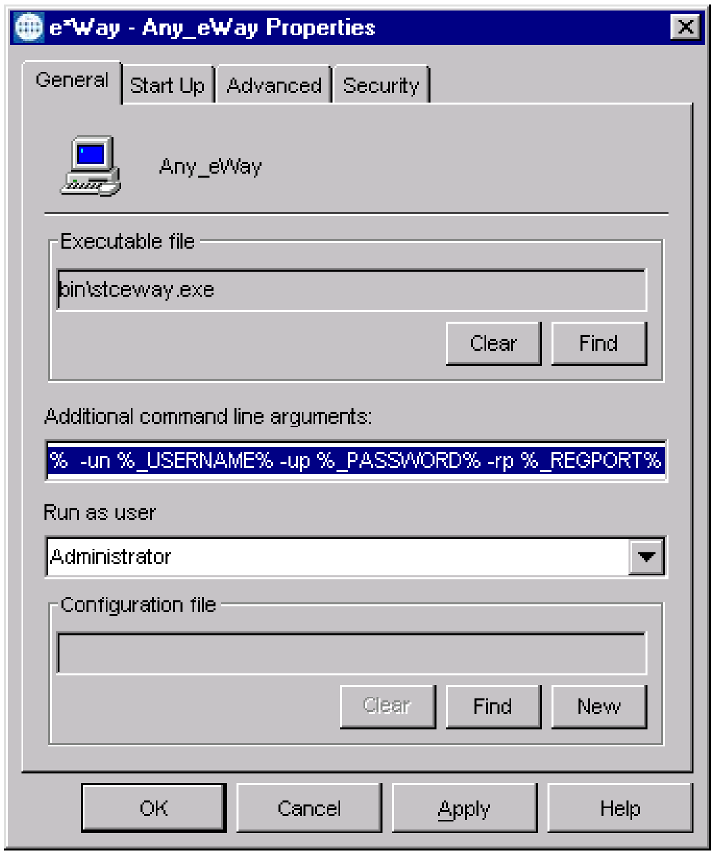
4.2.2 Modifying e*Way Properties

To modify any e*Way properties

8 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 20).

Note: *The executable files are* **stcewgenericmonk.exe** *and* **stcpsconvert.exe***.*

Figure 20 e*Way Properties (General Tab)



Executable file
bin\stceway.exe

Additional command line arguments:
% -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%

Run as user
Administrator

Configuration file

9 Make the desired modifications, then click **OK**.

4.2.3 Configuring the e*Way

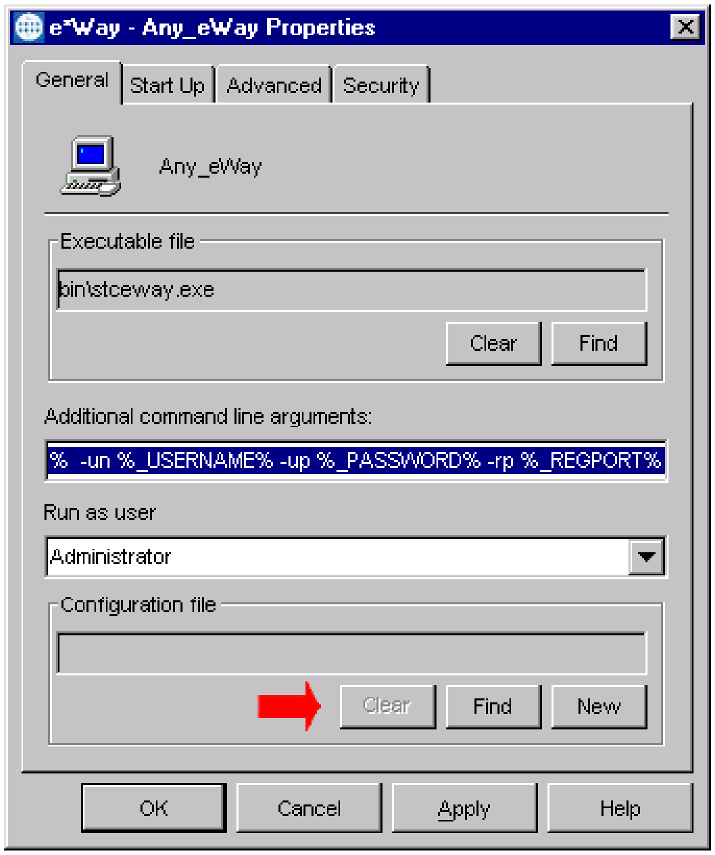
The e*Way's default configuration parameters are stored in an ASCII text file with a **.def** extension. The e*Way Editor provides a simple graphical interface for viewing and changing those parameters to create a working configuration (**.cfg**) file.

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 **psoft.def**.

Figure 21 e*Way Properties - General Tab



Executable file: bin\stceway.exe

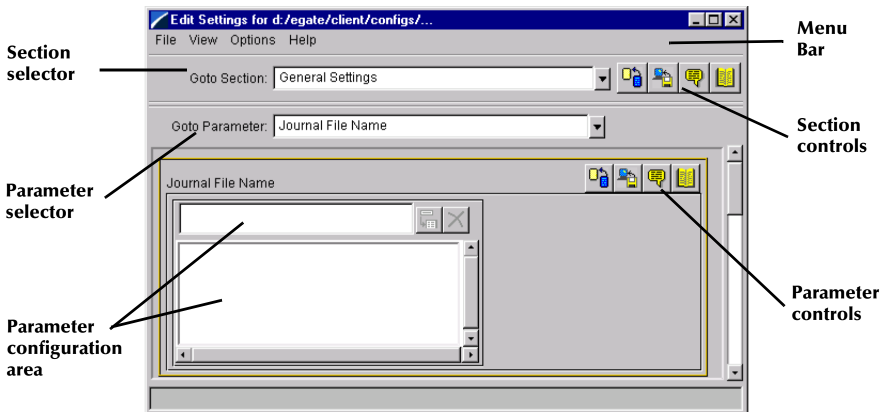
Additional command line arguments: % -un %_USERNAME% -up %_PASSWORD% -rp %_REGPORT%

Run as user: Administrator

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. Click this button to edit the currently selected file.
3. You are now in the e*Way Configuration Editor.

4.2.4 Using the e*Way Editor

Figure 22 The e*Way Configuration Editor


Edit Settings for d:/egate/client/configs/...
File View Options Help
Goto Section: General Settings
Goto Parameter: Journal File Name
Journal File Name
Section selector
Parameter selector
Parameter configuration area
Menu Bar
Section controls
Parameter controls

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 10 below.

Table 10 Parameter and Section Controls

Button	Name	Function
	Restore Default	Restores default values
	Restore Value	Restores saved values
	Tips	Displays tips
	User Notes	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 11

Table 11 Selection List Controls

Button	Name	Function
	Add to List	Adds the value in the text box to the list of available values.
✗	**Delete Items**	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*.

4.2.5 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 is to run.

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

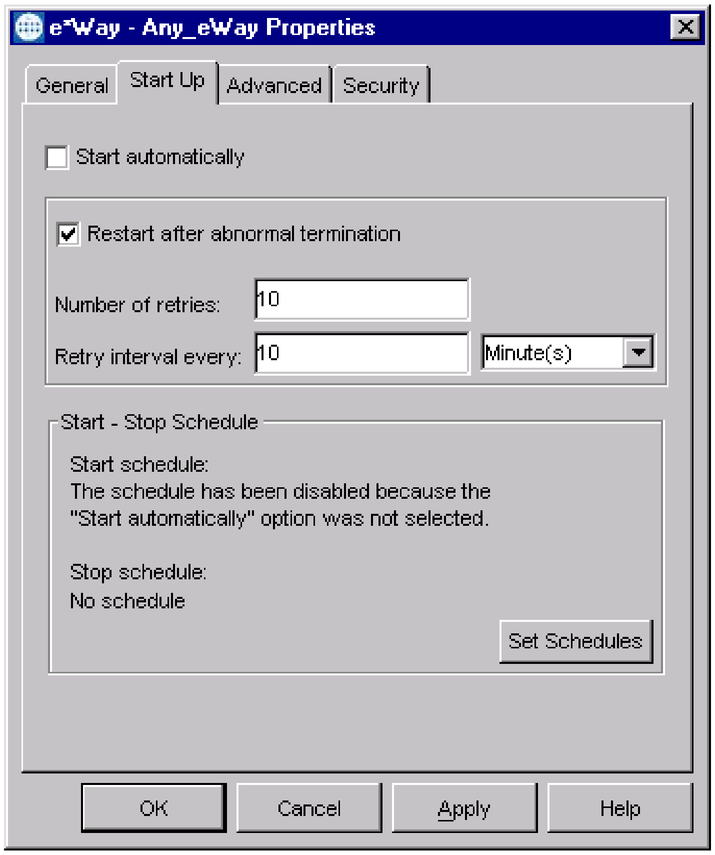
4.2.6 Setting Startup Options or Schedules

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 23). See the *e*Gate Integrator System Administration and Operations Guide* for more information about how interactive monitors can start or shut down components.

Figure 23 e*Way Properties (Start-Up Tab)



☐ Start automatically

☑ Restart after abnormal termination

Number of retries: 10

Retry interval every: 10 Minute(s)

Start - Stop Schedule

Start schedule:
The schedule has been disabled because the "Start automatically" option was not selected.

Stop schedule:
No schedule

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**.

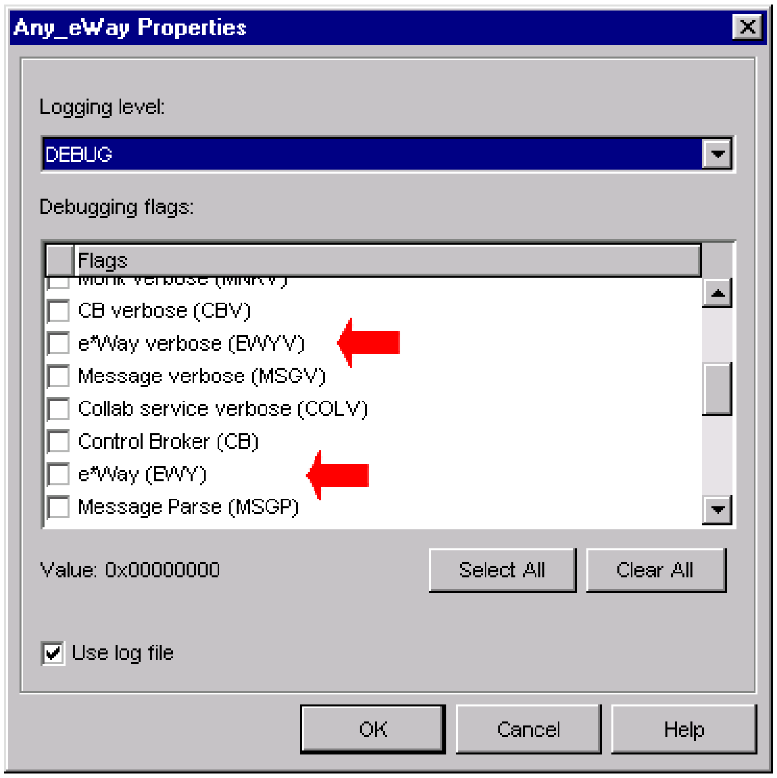
4.2.7 Activating or Modifying Logging Options

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 box appears (see Figure 24).

Figure 24 e*Way Properties (Advanced Tab - Log Option)


Any_eWay Properties

Logging level: DEBUG

Debugging flags:

Flags
- [ ] Monk verbose (MNKV)
- [ ] CB verbose (CBV)
- [ ] e*Way verbose (EWYV)
- [ ] Message verbose (MSGV)
- [ ] Collab service verbose (COLV)
- [ ] Control Broker (CB)
- [ ] e*Way (EWY)
- [ ] Message Parse (MSGP)

Value: 0x00000000

- [x] Use log file

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.

4.2.8 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. Select 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.

4.3 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.

4.3.1 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 Rules 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 Message Agent dynamically-loaded libraries. The name of this variable on the different operating systems is:
  - PATH (Windows)
  - LD_LIBRARY_PATH (Solaris)

In the PeopleSoft Application

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

4.3.2 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 Rules scripts that become evident in the course of normal operations.

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 e*Gate monitoring system to monitor operations and performance.

Chapter 5

Operational Overview

This chapter contains an overview of the interaction between the e*Way and PeopleSoft Message Agent in typical applications, and also the architecture and basic internal processes of the SeeBeyond e*Way Intelligent Adapter for PeopleSoft Message Agent.

5.1 System Operation

5.1.1 e*Gate to PeopleSoft (Event Driven)

An inbound event-driven interface to PeopleSoft involves the real-time or near-real-time processing of a single, individual transaction triggered by a user or business event in the source application. Examples include the creation of a customer sales order or a modification of an employee's personal data. Data affected by these events in the source application are passed to e*Gate for translation, and are loaded into PeopleSoft by the PeopleSoft Message Agent e*Way.

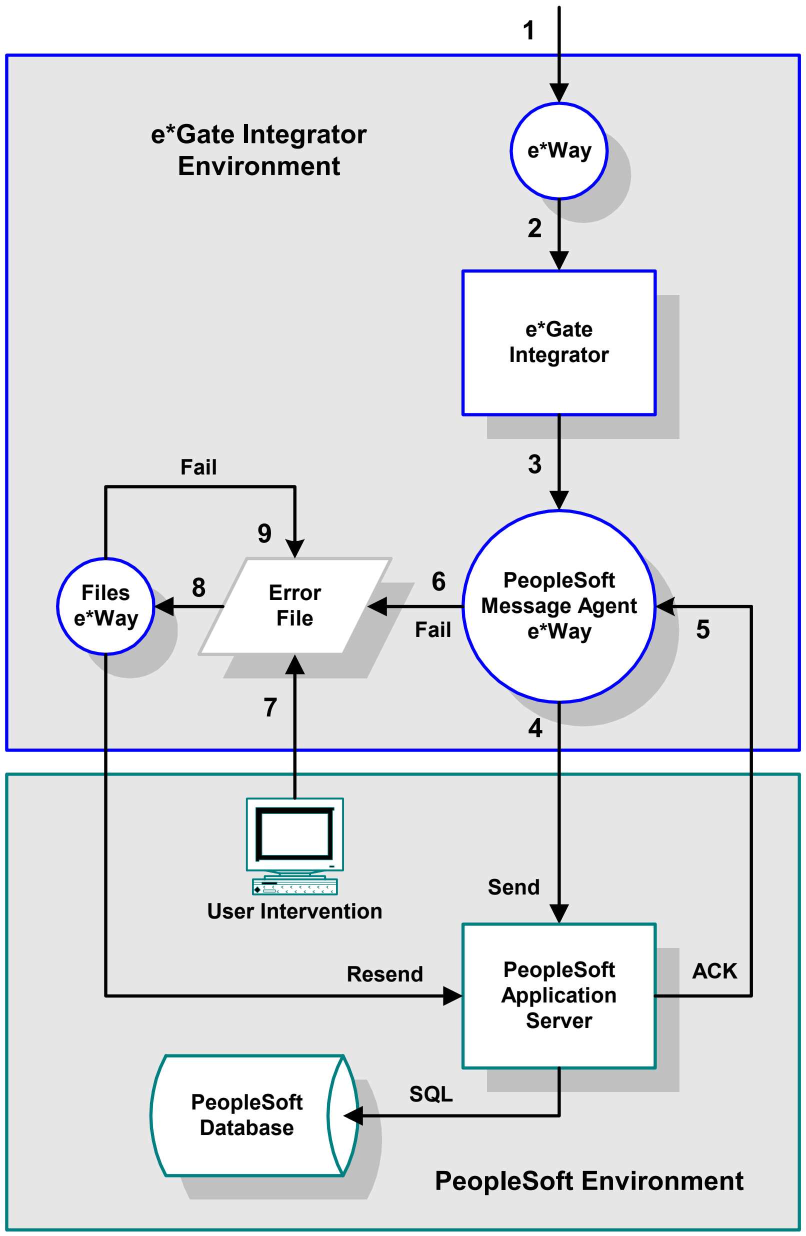
The main components of an inbound event-driven interface include:

- A source-specific e*Way, to receive the application data
- e*Gate Integrator, to transform and route the application data
- A PeopleSoft Message Agent e*Way, to load the transformed data into the PeopleSoft system
- An error-handling Files e*Way, to process failed transactions

Process Flow

1. A business event is performed in the source application and the interface is triggered.
2. The source e*Way sends the data to e*Gate Integrator in the form of a structured Event.
3. e*Gate Integrator then transforms the structure of the Event into that required by PeopleSoft, also performing any necessary data translations.

Figure 25 e*Gate-to-PeopleSoft Event-Driven Interface



Note: *The PeopleSoft inbound message structure may contain multiple substructures—corresponding to individual message definitions—to match the business event as performed in PeopleSoft. For example, an employee hire in the source system might require multiple data mappings, and therefore multiple message definitions, to duplicate the event in PeopleSoft.*

4 The transformed data are then routed to the PeopleSoft Message Agent e*Way. Once received, the e*Way processes the Event by sending the data to PeopleSoft in packets corresponding to individual message definitions, one packet at a time. The flag for each data packet initially is set to N.

5. After each data packet is processed successfully by the Message Agent, a commit is issued to the PeopleSoft database and the flag for that data packet is set to **Y**. Once all the data packets in the message have been processed successfully, an **ACK** is sent back to the PeopleSoft Message Agent e*Way, allowing the next Event to be submitted for processing.
6. If any data packet is not processed successfully, the entire inbound Event is *failed* and written to a general error file. To prevent the Event from being resent to the e*Way, an **ACK** is sent to e*Gate Integrator to remove the Event from the queue. The flag for the failed data packet remains set to **N**.
7. At this point user intervention is required to edit the error file, using a standard file editor, and correct the offending data.
8. Once the file is corrected, the error file is moved to the error handling directory where the error handling e*Way loads and reprocesses the Event. Only unprocessed data packets, with flags set to **N**, are reprocessed.
9. If an error recurs at this point, the Event is saved to the error file and the error processing is repeated.

5.1.2 e*Gate to PeopleSoft (Batch Mode)

An inbound batch interface involves the loading of a collection of transactions usually in the form of a data file into PeopleSoft. The interface can be triggered by a user or application event in the source system or by a scheduling mechanism. Examples include a nightly synchronization of inventory information with an external system or a weekly load of timecard information into PeopleSoft Payroll.

The interface data file from the source application is passed to e*Gate Integrator for translation and loaded into PeopleSoft via the Message Agent. The main components of this inbound batch interface include:

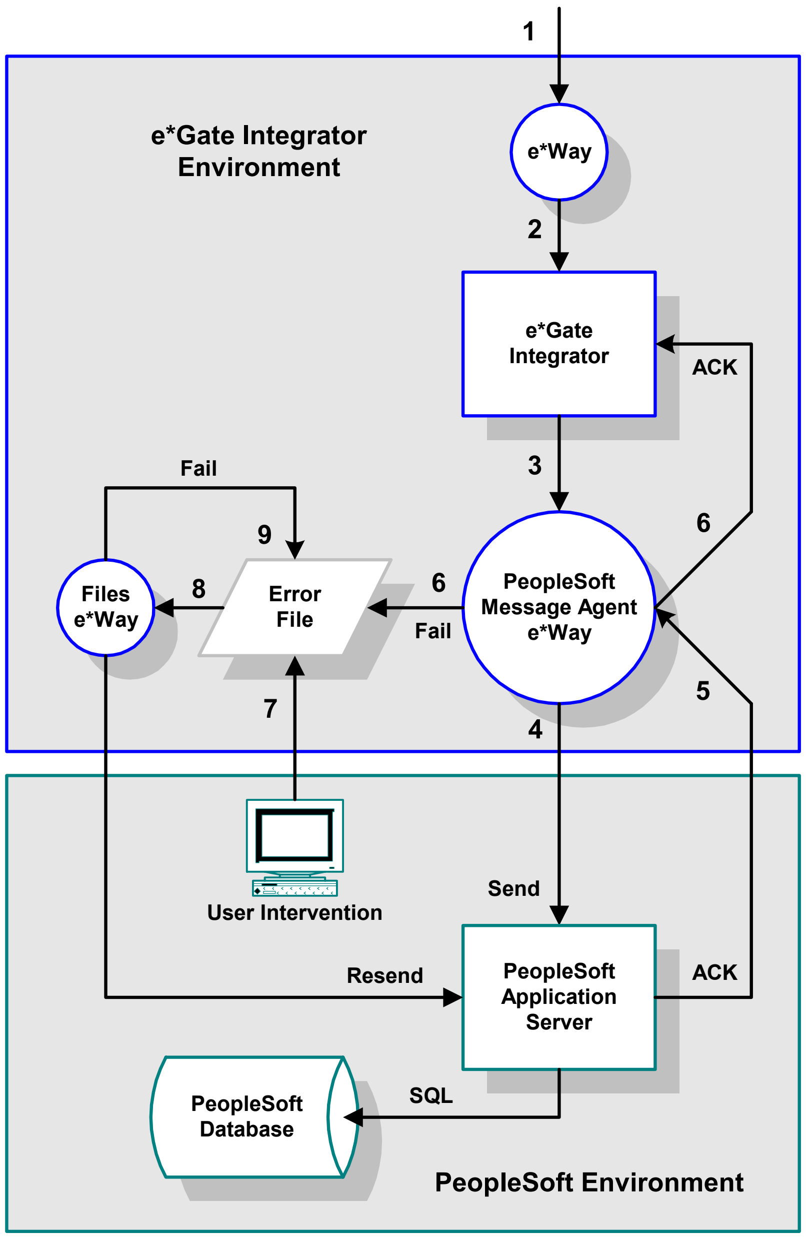
- A source-specific e*Way, to transfer a source data file
- e*Gate Integrator, to transform the data
- A PeopleSoft Message Agent e*Way, to load the transformed data into the PeopleSoft system
- An error-handling Files e*Way, to process failed transactions

Typically, batch interfaces into PeopleSoft involve the direct insertion of data into the PeopleSoft application tables. All data validation and business rules need to be identified and incorporated into the interface program. This design, on the other hand, utilizes the PeopleSoft Message Agent, which handles the PeopleCode processing and data validation.

The architecture of this interface is a scaled-up extension of the inbound event-driven interface. For both interfaces, the Message Agent is used to load the data into PeopleSoft. However, since the batch interface is designed to process a higher volume of transactions—along with multiple PeopleSoft application servers and Message Agent processes—multiple PeopleSoft Message Agent e*Ways may be required.

Process Flow

Figure 26 e*Gate-to-PeopleSoft Batch-Mode Interface



1 A trigger event or schedule in the source application initiates the creation of an interface file.

2 When the file is complete, the source application e*Way sends the entire data file to e*Gate Integrator.

3 e*Gate Integrator then transforms the structure of the Event into that required by PeopleSoft, also performing any necessary data translations.

Note: *The PeopleSoft inbound message structure may contain multiple substructures—corresponding to individual message definitions—to match the business event as performed in PeopleSoft. For example, an employee hire in the source system might require multiple data mappings, and therefore multiple message definitions, to duplicate the event in PeopleSoft.*

4. The transformed data are then routed to the PeopleSoft Message Agent e*Way. Once received, the e*Way processes the Event by sending the data to PeopleSoft in packets corresponding to individual message definitions, one packet at a time. The flag for each data packet initially is set to **N**.
5. After each data packet is processed successfully by the Message Agent, a commit is issued to the PeopleSoft database and the flag for that data packet is set to **Y**. Once all the data packets in the message have been processed successfully, an **ACK** is sent back to the PeopleSoft Message Agent e*Way, allowing the next Event to be submitted for processing.
6. If any data packet is not processed successfully, the entire inbound Event is *failed* and written to a general error file. To prevent the Event from being resent to the e*Way, an **ACK** is sent to e*Gate Integrator to remove the Event from the queue. The flag for the failed data packet remains set to **N**.
7. Depending on the error handling requirements, when an error is encountered the interface may stop processing, or bypass the error and continue processing. Rolling back all the transactions is not supported.
8. If stop on error processing is desired, the PeopleSoft Message Agent e*Way is shut down so that no more transactions are sent to PeopleSoft. Any unprocessed transactions from the interface file remain in the e*Gate Integrator queue.
9. If continue on error processing is desired, the e*Way is not shut down, and after the failed Event is written to the error file the remaining Events in the queue continue normal processing. Additional failed transactions are appended to the error file. When all of the transactions in the interface data file have been processed, the user can then correct the failed records for reprocessing.
10. At this point user intervention is required to edit the error file, using a standard file editor, and correct the offending data.
11. Once the file is corrected, the error file is moved to the error handling directory where the error handling e*Way loads and reprocesses the Event. Only unprocessed data packets, with flags set to **N**, are reprocessed.
12. For stop-on-error processing, once the failed Event has been processed successfully, the PeopleSoft Message Agent e*Way is re-started to continue processing the remaining transactions in the e*Gate Integrator queue.

5.1.3 PeopleSoft to e*Gate (Event Driven)

An outbound event-driven interface from PeopleSoft involves the real-time or near-real-time processing of a single, individual transaction triggered by a user or application business event. Examples include the creation of a customer sales order or a modification of an employee's personal data.

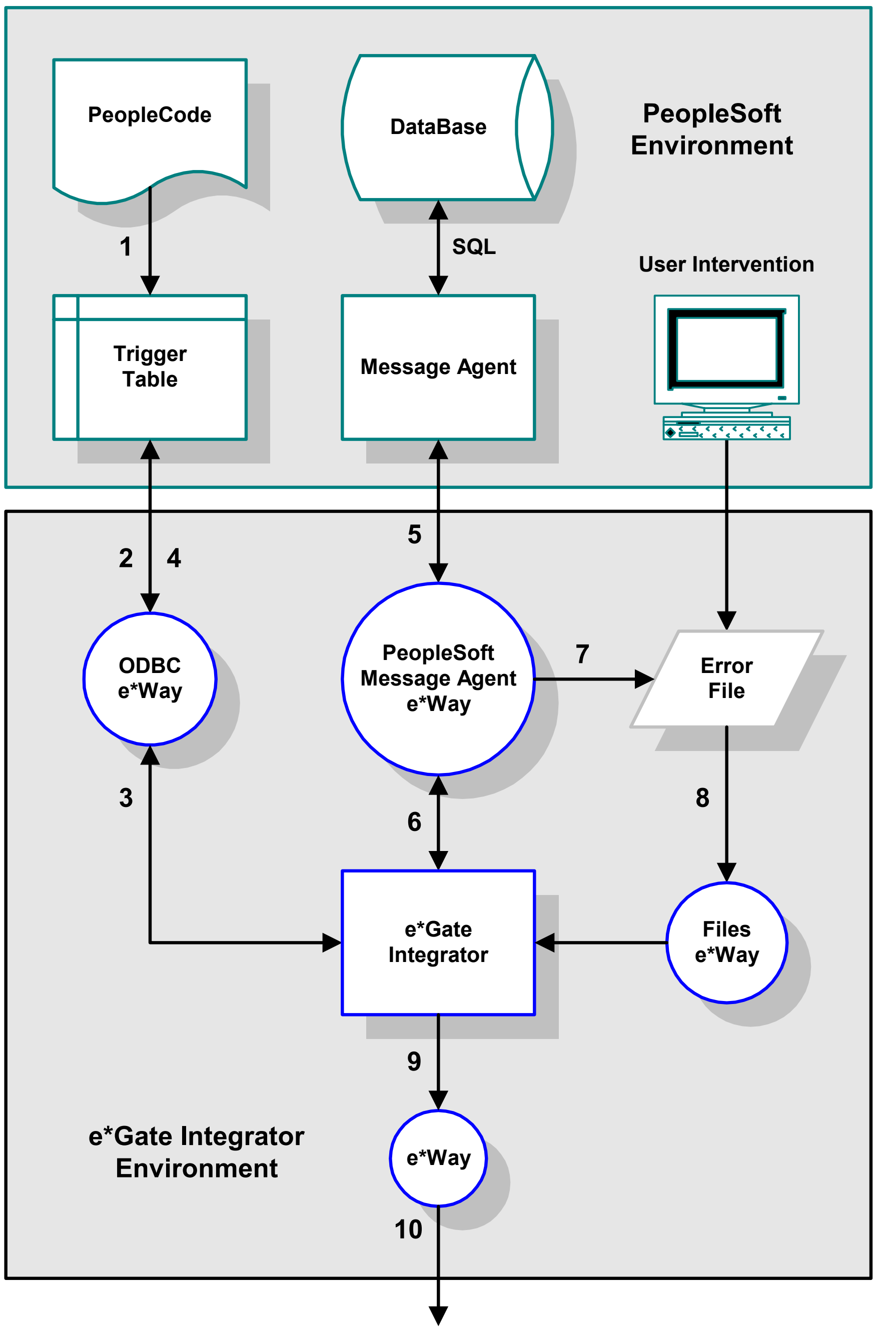
These events in the PeopleSoft source system produce a trigger which e*Gate Integrator reads, transforms, and loads into the target system. The main components of an outbound event-driven interface include:

- An ODBC e*Way, to retrieve a PeopleCode trigger from the source system
- A PeopleSoft Message Agent e*Way, to retrieve application data
- e*Gate Integrator, to transform the application data
- A target-system-specific e*Way, to load the transformed data into the target system

Process Flow

1. A business event is performed and saved in the PeopleSoft application. The PeopleCode associated with the event writes a row to the trigger table with the corresponding key fields of the event. The trigger status field is set to **N** (not processed).
2. The ODBC e*Way polls the trigger table for any unprocessed events based on the trigger status field (the polling frequency is a configurable e*Gate Integrator parameter).
3. The key fields of the business event are selected from the trigger table and sent to e*Gate Integrator's queue for subsequent routing to the PeopleSoft Message Agent e*Way. Once the key fields have been successfully delivered to e*Gate Integrator, an **ACK** is be sent back to the ODBC e*Way.
4. ODBC then updates the trigger status for the retrieved rows in the trigger table. The status is updated to **P** (processed).
5. The PeopleSoft Message Agent e*Way then uses the key fields from the trigger row to extract the data from the PeopleSoft application via the Message Agent.
6. If the data extraction is successful, the PeopleSoft Message Agent e*Way sends the extracted application data to e*Gate Integrator. e*Gate Integrator then sends an **ACK** to the PeopleSoft Message Agent e*Way to let it know that the data has been successfully received. The PeopleSoft Message Agent e*Way then sends a return **ACK** to the e*Gate Integrator that it is ready for the next set of trigger key fields.
7. If the data extraction is not successful, and the Message Agent returns a failure, the error key fields are written to an error file where a user can correct the keys with a file editor.
8. Once corrected, the error file is copied to the error handling e*Way where the key fields are loaded into the e*Gate Integrator queue. Normal processing resumes as the key fields are used by the PeopleSoft Message Agent e*Way to extract the PeopleSoft application data.

Figure 27 PeopleSoft-to-e*Gate Event-Driven Interface



9 e*Gate Integrator then routes the retrieved PeopleSoft application data to the target application e*Way.

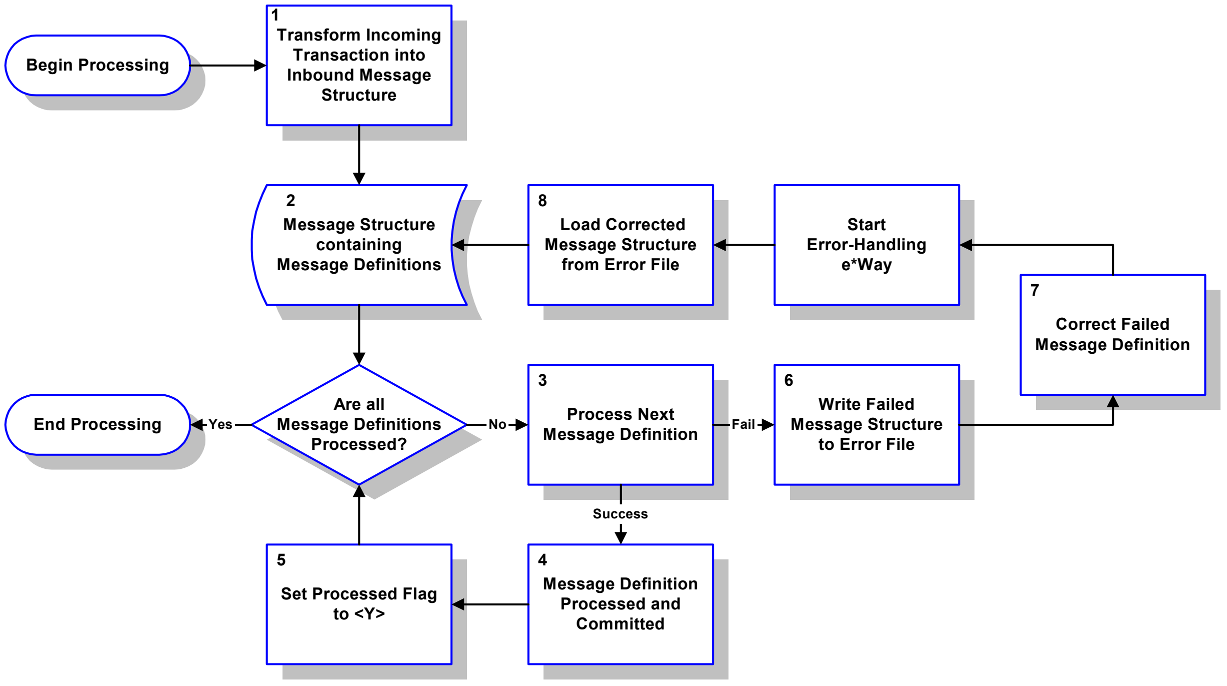
10 The target application e*Way then transforms and maps the retrieved data to the target system inbound Event Type Definition and loads the data into the target application.

5.2 Transaction Processing

5.2.1 e*Gate to PeopleSoft (Event Driven)

The interface allows the transactions to be managed down to the smallest logical unit of work-a single message definition. When an error occurs loading the data with the Message Agent, the entire message structure, which could contain multiple message definitions, is written to a general error file along with a flag that indicates the nature of the failure. This allows a user to correct the offending data and resubmit the entire message structure. The flag prevents the reprocessing of message definitions that already have been processed successfully.

Figure 28 e*Gate-to-PeopleSoft Event-Driven Processing


Begin Processing
1 Transform Incoming Transaction into Inbound Message Structure
2 Message Structure containing Message Definitions
8 Load Corrected Message Structure from Error File
Start Error-Handling e*Way
7 Correct Failed Message Definition
End Processing
Yes
Are all Message Definitions Processed?
No
3 Process Next Message Definition
Fail
6 Write Failed Message Structure to Error File
Success
5 Set Processed Flag to <Y>
4 Message Definition Processed and Committed

The processing of a message structure for a single business event by the PeopleSoft Message Agent e*Way is shown in Figure 27: Inbound Event-Driven Transaction Processing Flow. The processing steps are as follows.

1 When a business event occurs in the source system, the data from the transaction is e and routed to the PeopleSoft Message Agent e*Way.

2 The result is a Event Type Definition that comprises of one or more message definitions containing the transformed data for loading into PeopleSoft.

3 Each message definition within the message structure is submitted to the Message Agent one at a time.

4 A separate commit is issued after the successful processing of each message definition using the Message Agent **ProcessMessage** function.

5 The processed flag for the message definition is set to **Y**.

These steps are repeated until all the message definitions in the message structure are successfully processed. Once the entire message structure is processed, an **ACK** is sent to e*Gate Integrator by the PeopleSoft Message Agent e*Way to signify that it is ready for the next Event.

6 If a message definition fails, the entire message structure is failed and written to a general error file. An **ACK** is sent to e*Gate Integrator to prevent the Event from being resent to the e*Way.

7 Using a file editor, the failed message definition should be corrected.

8 Once the failed message definition is corrected, the error file is moved to the error-handling directory where the error-handling e*Way automatically loads and reprocesses the corrected error file or Event.

The processed flag indicates the message definitions that have not yet been successfully committed to the database. This prevents any message definitions that were successfully processed prior to the failure from being reprocessed in the error-handling e*Way.

5.2.2 PeopleSoft to e*Gate (Batch Mode)

The interface allows the transactions to be managed down to the smallest logical unit of work-a single message definition. When an error occurs loading the data with the Message Agent, the entire message structure, which could contain multiple message definitions, is written to a general error file along with a flag that indicates the nature of the failure. This allows a user to correct the offending data and resubmit the entire message structure. The flag prevents the reprocessing of Events that already have been processed successfully.

Stop on Error (Restart)

During the execution of a stop on error type of interface, processing stops if an error is encountered. This type of processing would be desired if there is a high level of dependency among the rows of data in the interface file. In this situation, the failure of one transaction prevents subsequent transactions from processing correctly. The following diagram illustrates the transaction processing flow for stop-on-error processing.

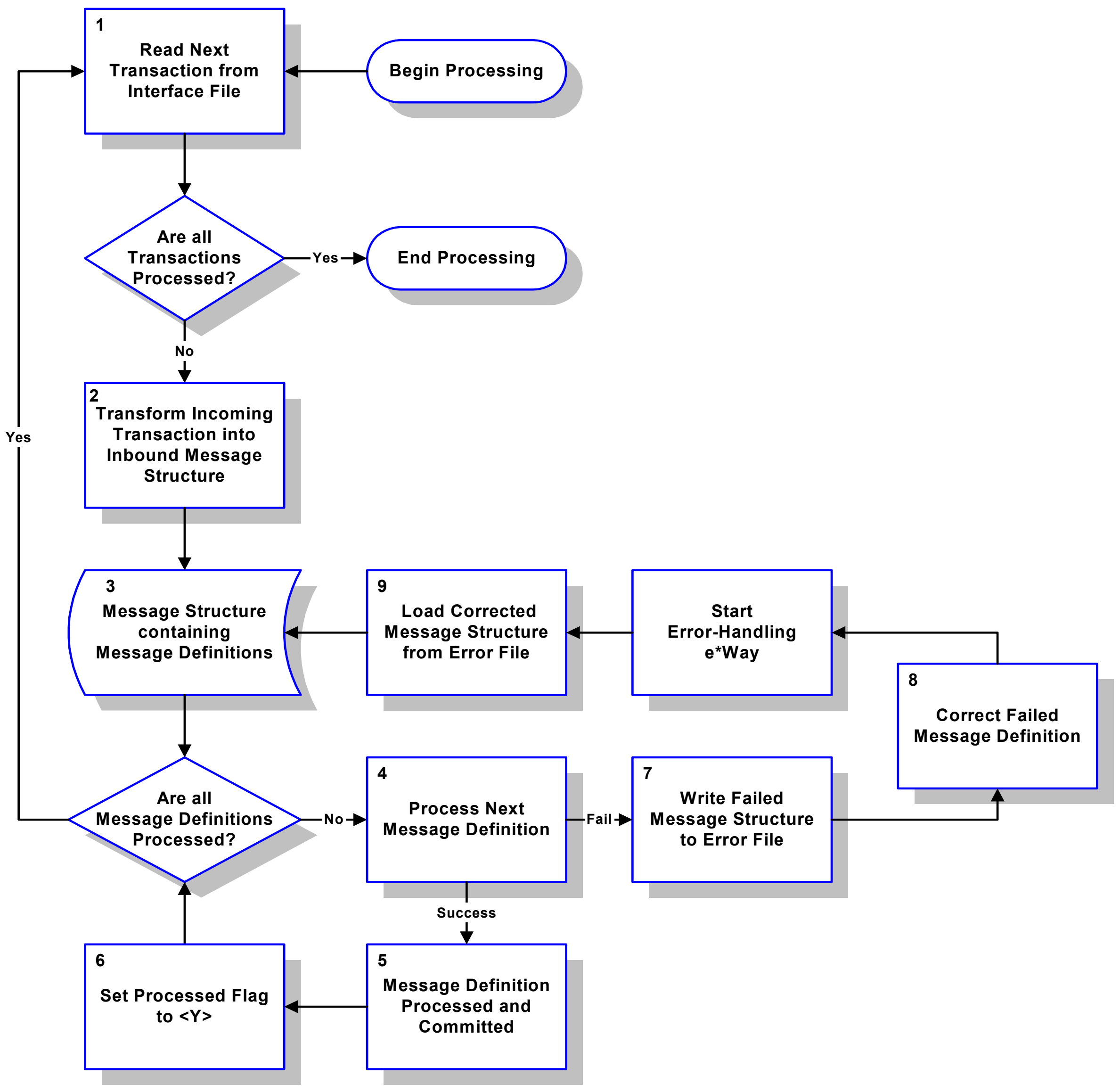
1 A row of data is read from the interface file.

2 The data from the transaction is transformed, loaded to the PeopleSoft inbound message definition, and routed to the PeopleSoft Message Agent e*Way.

3 The result is a message structure which is comprised of one or more message definitions containing the transformed data for loading into PeopleSoft.

4 Each message definition within the message structure is submitted to the Message Agent, one at a time.

Figure 29 PeopleSoft-to-e*Gate Stop-on-Error Processing


1
Read Next Transaction from Interface File
Begin Processing
Are all Transactions Processed?
Yes
End Processing
No
2
Transform Incoming Transaction into Inbound Message Structure
Yes
3
Message Structure containing Message Definitions
9
Load Corrected Message Structure from Error File
Start Error-Handling e*Way
8
Correct Failed Message Definition
Are all Message Definitions Processed?
No
4
Process Next Message Definition
Fail
7
Write Failed Message Structure to Error File
Success
6
Set Processed Flag to <Y>
5
Message Definition Processed and Committed

5 A separate commit is issued after the successful processing of each message definition using the Message Agent **ProcessMessage** function.

6 The processed flag for the message definition is set to **Y**. These steps are repeated until all the message definitions of the message structure are successfully processed. Then the next row of data from the interface file is read and the entire process is repeated until the end of file is reached.

7 If a message definition fails, the entire message structure is failed and written to a general error file with the processed flag still set to **N**.

8 The PeopleSoft Message Agent e*Way shuts down so that no more transactions are sent to PeopleSoft.

9 Using a file editor, the user corrects the failed message definition.

10 Once the failed message definition is corrected, the error-handling e*Way is started to automatically load the corrected error file or message structure into the error-handling e*Way for reprocessing.

11 Only the message definitions with a processed flags set to **N** are reprocessed. This prevents any message definitions that were successfully processed prior to the failure from being reprocessed in the error-handling e*Way.

12 Once the reprocessing successfully completes, the PeopleSoft Message Agent e*Way is restarted to continue processing the remaining rows in the file.

Continue on Error (Reprocess)

During the execution of a continue on error type of interface, when an error is encountered the failed transaction is saved to an error queue so that the remaining transactions can continue processing. Once the entire interface file has been processed, error transactions can then be corrected and resubmitted for processing. This type of processing is recommended when there is a large volume of data that must be processed within the batch window. In this scenario, the errors do not prevent other transactions from processing. The following diagram illustrates the transaction processing flow for continue-on-error processing.

During normal execution, the transaction processing flow is identical to that of the Stop on Error processing described earlier). The difference arises when an error is encountered (see step 7).

1 A row of data is read from the interface file.

2 The data from the transaction is transformed, loaded to the PeopleSoft inbound message definition, and routed to the PeopleSoft Message Agent e*Way.

3 The result is a message structure which is comprised of one or more message definitions containing the transformed data for loading into PeopleSoft.

4 Each message definition within the message structure is submitted to the Message Agent, one at a time.

5 A separate commit is issued after the successful processing of each message definition using the Message Agent **ProcessMessage** function.

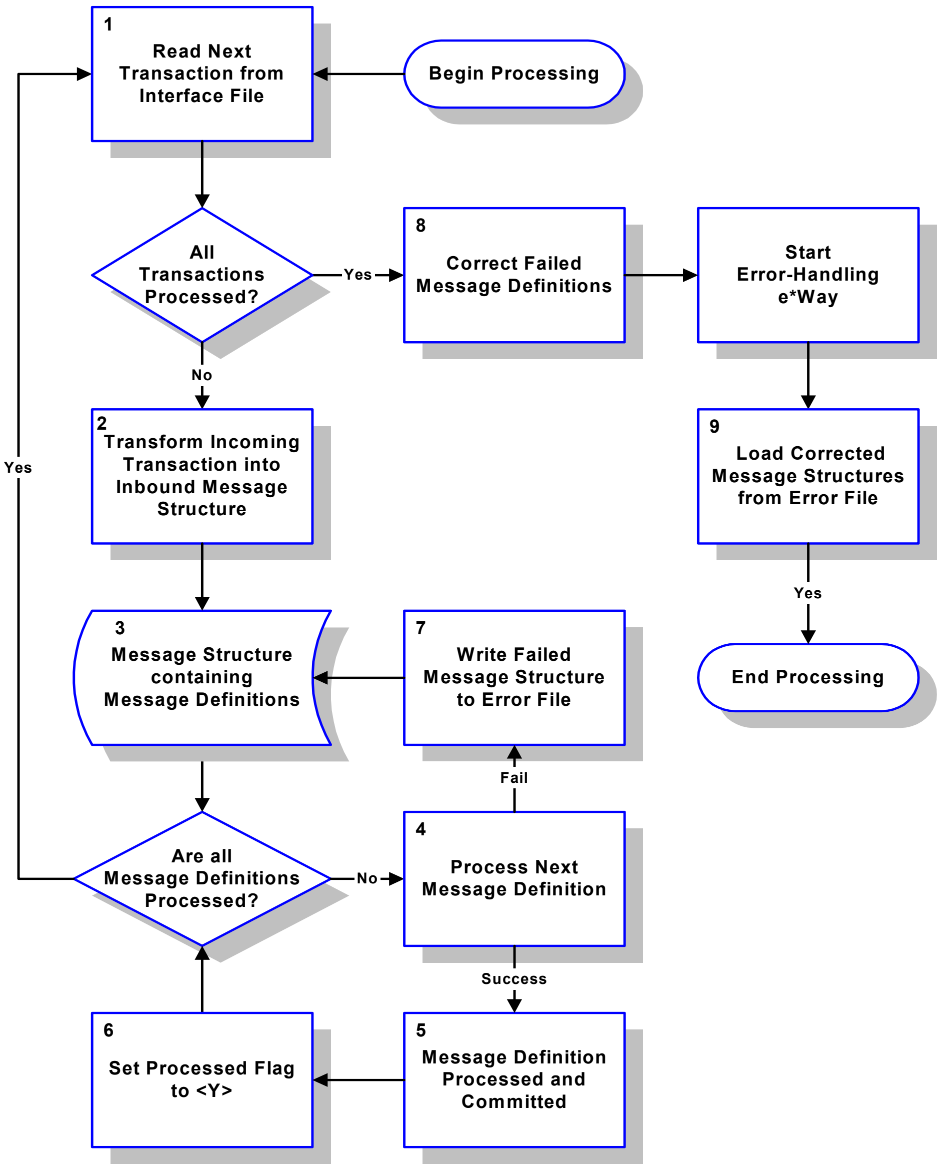
6 The processed flag for the message definition is set to **Y**. These steps are repeated until all the message definitions of the message structure are successfully processed. Then the next row of data from the interface file is read and the entire process is repeated until the end of file is reached.

7 If a message definition fails, the entire message structure is *failed* and written to a general error file. The PeopleSoft Message Agent e*Way continues to process the remaining transactions in the interface file. Other errors that are encountered are appended to the error file.

8 When all of the transactions in the interface file have been processed, the failed message definitions in the error file require user correction using a file editor.

9 Once all of the failed message definitions are corrected, the error-handling e*Way is started to automatically load the corrected error file into the error-handling e*Way for reprocessing.

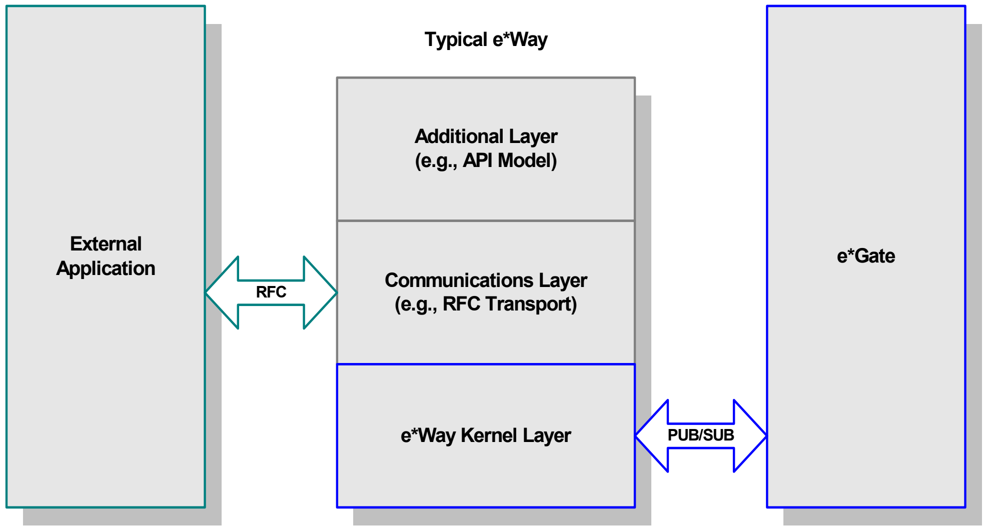
Figure 30 PeopleSoft-to-e*Gate Continue-on-Error Processing


1
Read Next Transaction from Interface File
Begin Processing
All Transactions Processed?
Yes
8
Correct Failed Message Definitions
Start Error-Handling e*Way
No
Yes
2
Transform Incoming Transaction into Inbound Message Structure
9
Load Corrected Message Structures from Error File
Yes
3
Message Structure containing Message Definitions
7
Write Failed Message Structure to Error File
End Processing
Fail
Are all Message Definitions Processed?
No
4
Process Next Message Definition
Success
6
Set Processed Flag to <Y>
5
Message Definition Processed and Committed

5.3 e*Way Architecture

Conceptually, an e*Way can be viewed as a multi-layered structure, consisting of one or more layers (see Figure 31). 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 31 Typical e*Way Architecture


Typical e*Way
External Application
RFC
Additional Layer (e.g., API Model)
Communications Layer (e.g., RFC Transport)
e*Way Kernel Layer
PUB/SUB
e*Gate

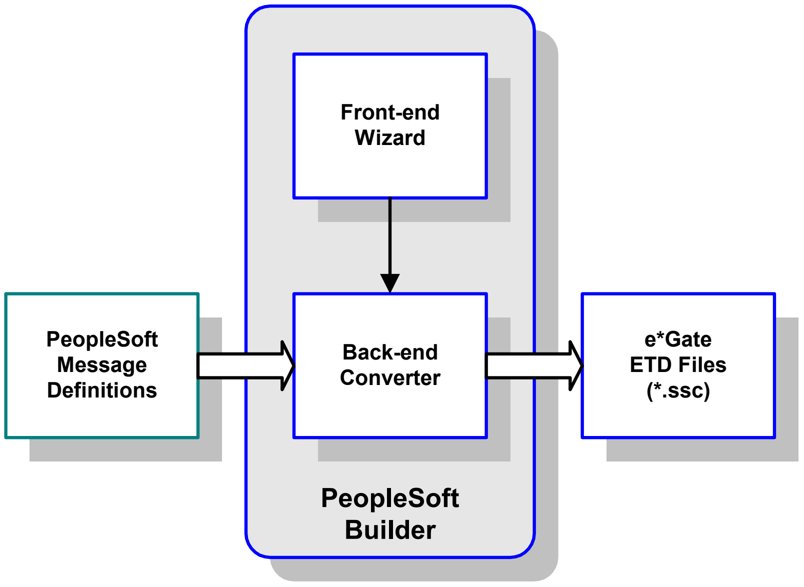
The upper layers of the e*Way use Monk functions to perform Business Process modeling and ETD mapping, package data as e*Gate *Events*, send those Events to Collaborations, and manage interaction with the external system. These layers are built upon an e*Way Kernel layer that manages the basic operations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

5.4 Event Type Definitions and Collaborations

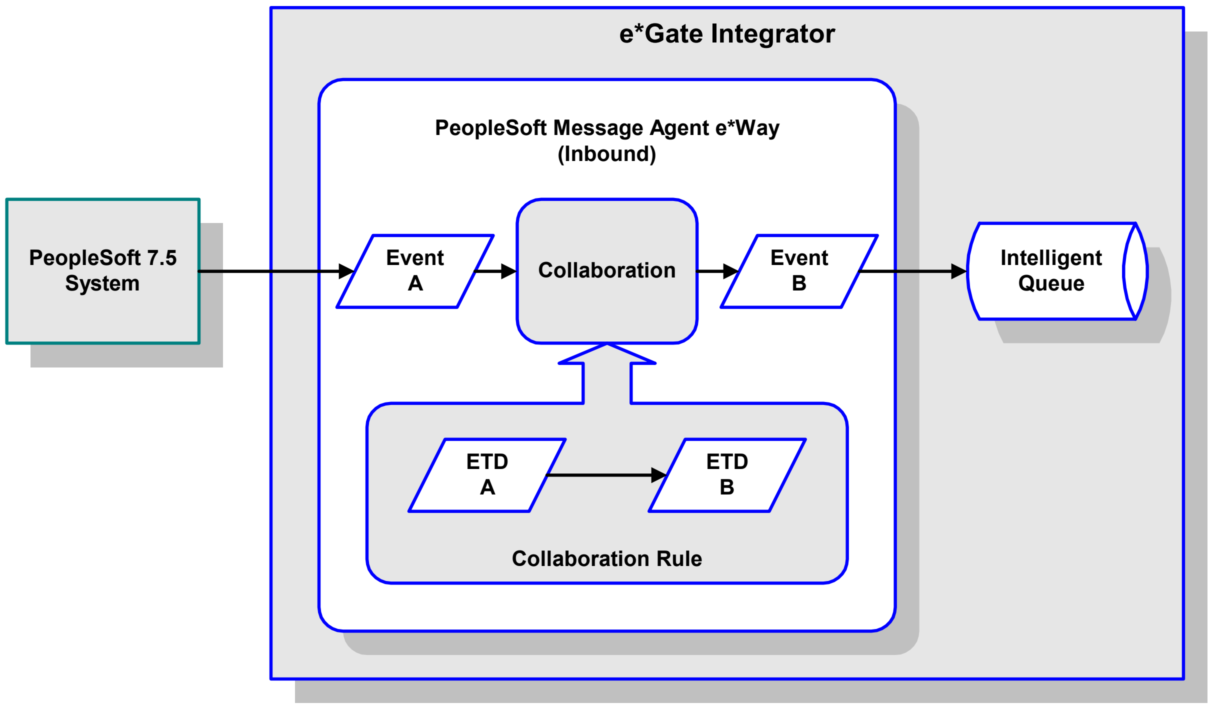
Event Type Definitions (ETDs) embody the business rules related to the associated external application. In the case of the PeopleSoft Message Agent e*Way, these ETDs are built semi-automatically by means of the PeopleSoft ETD Builder (see Figure 32). Each Collaboration uses a Collaboration Rule, which references the appropriate ETDs, and each Collaboration Rule uses a Collaboration Service.

Figure 32 PeopleSoft Builder Operation



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 33. Any Collaboration that publishes its processed Events only to an external system does *not* require *any* IQs.

Figure 33 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 6**. 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 7**. 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*.

5.5 Basic e*Way Processes

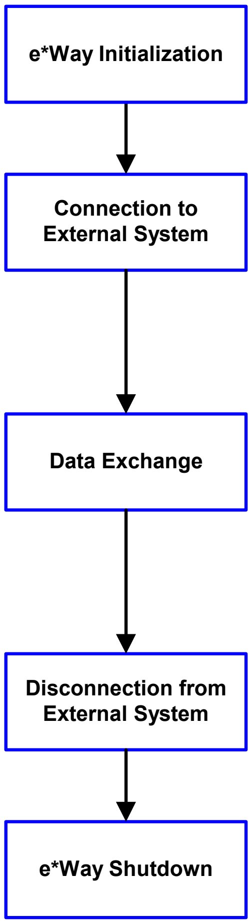
Note: *This section describes the basic operation of a typical e*Way based on the Generic e*Way Kernel. Not all functionality described in this section is used routinely by this e*Way.*

The most basic processes carried out by an e*Way are listed in Figure 34. In e*Ways based on the Generic Monk e*Way Kernel (**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 34 Basic e*Way Processes

Process

Monk Configuration Sections


e*Way Initialization
Connection to External System
Data Exchange
Disconnection from External System
e*Way Shutdown

Startup Function on page 88 (also see **Monk Environment Initialization File** on page 87)

External Connection Establishment Function on page 90
External Connection Verification Function on page 90

Event-driven Data Exchange
Process Outgoing Message Function on page 88

Schedule-driven Data Exchange
Exchange Data with External Function on page 89
Positive Acknowledgment Function on page 91
Negative Acknowledgment Function on page 92

External Connection Shutdown Function on page 91

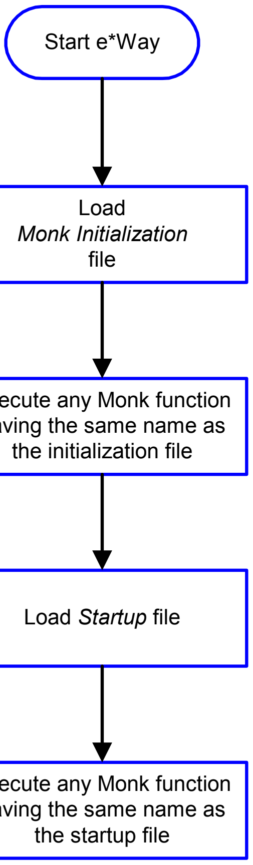
Shutdown Command Notification Function on page 93

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 6**, while the functions themselves are described in **Chapter 7**.

Initialization Process

Figure 35 illustrates the e*Way's initialization process, using the **Monk Environment Initialization File** and **Startup Function**.

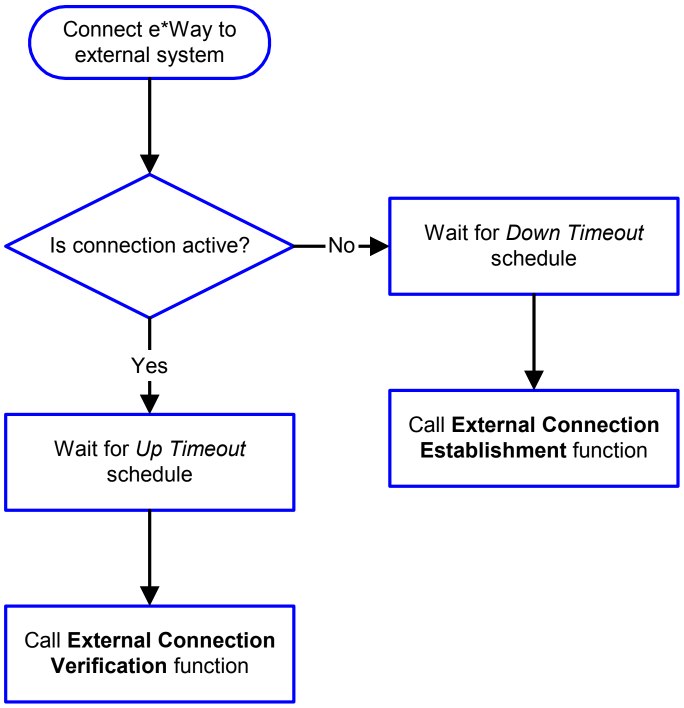
Figure 35 Initialization Process


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

Connect to External Process

Figure 36 illustrates how the e*Way connects to the external system, using the **External Connection Establishment Function** and **External Connection Verification Function**.

Figure 36 Connection Process


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

Note: *The e*Way selects the connection function based on an internal* ***up/down*** *flag rather than a poll to the external system. See* **Figure 38 on page 77** *and* **Figure 37 on page 76** *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 123 and* **send-external-down** *on page 123 for more information.*

Data Exchange Process

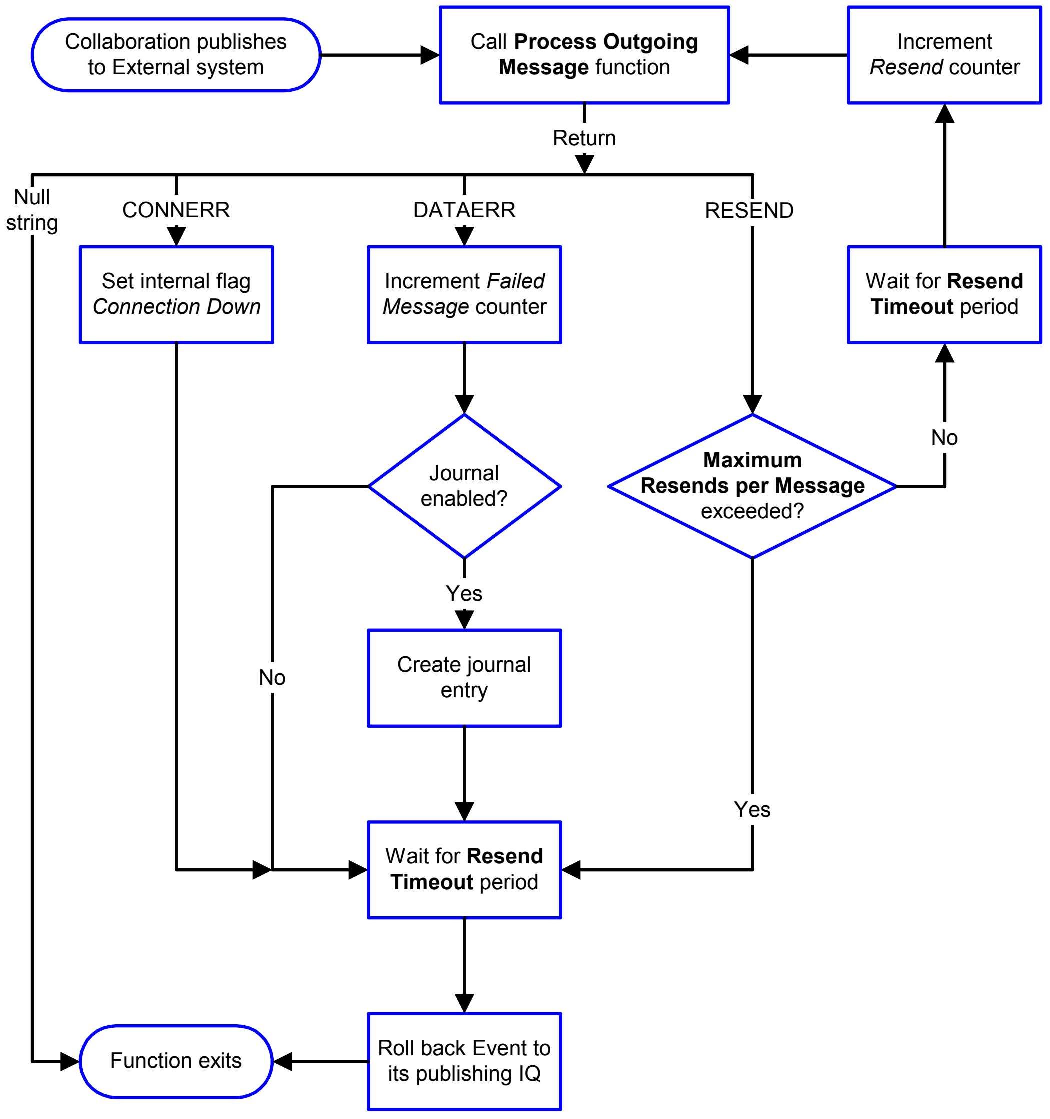
Event-driven

Figure 37 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.

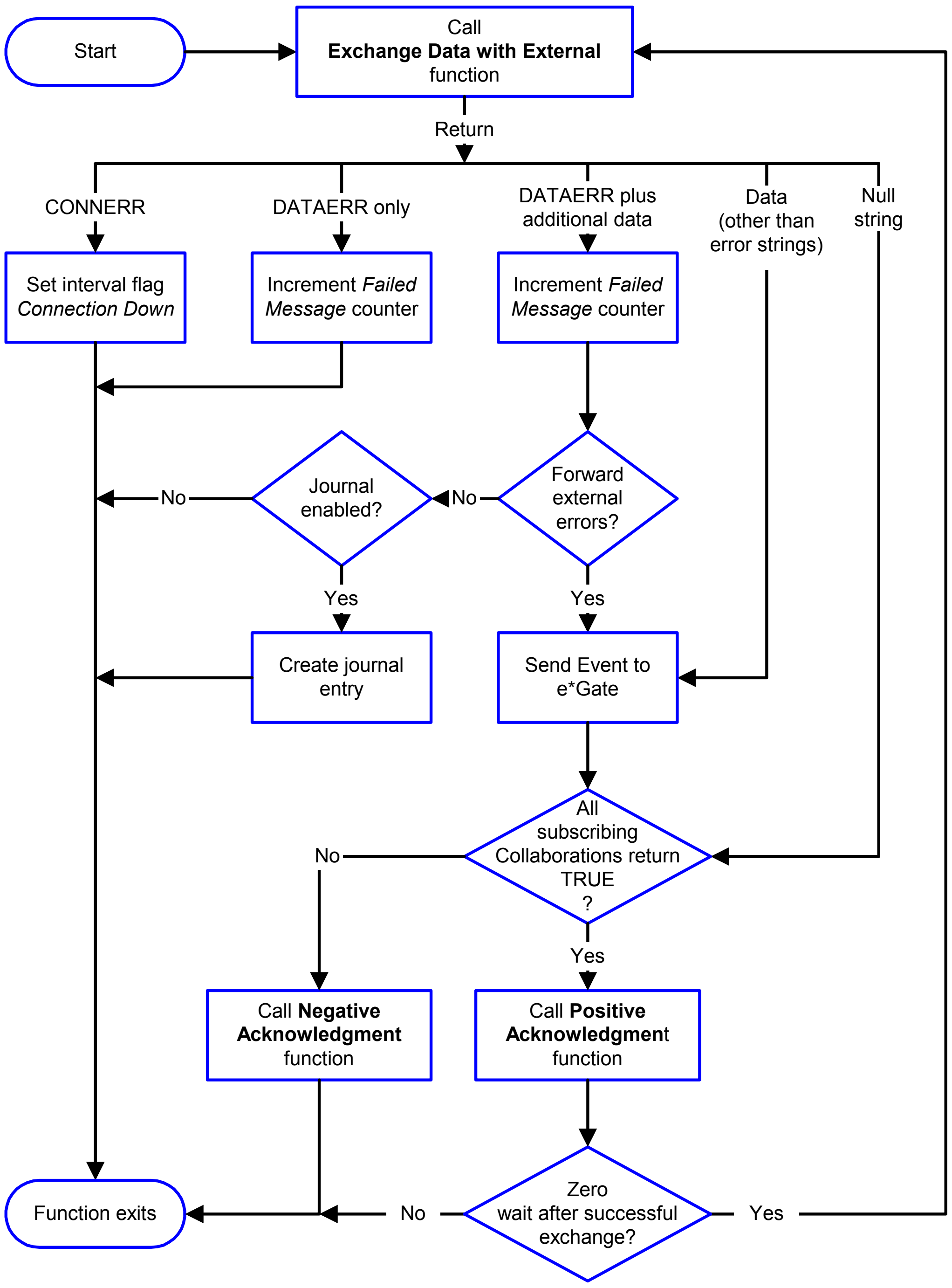
Figure 37 Event-Driven Data Exchange Process


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

Schedule-driven

Figure 38 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 38 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

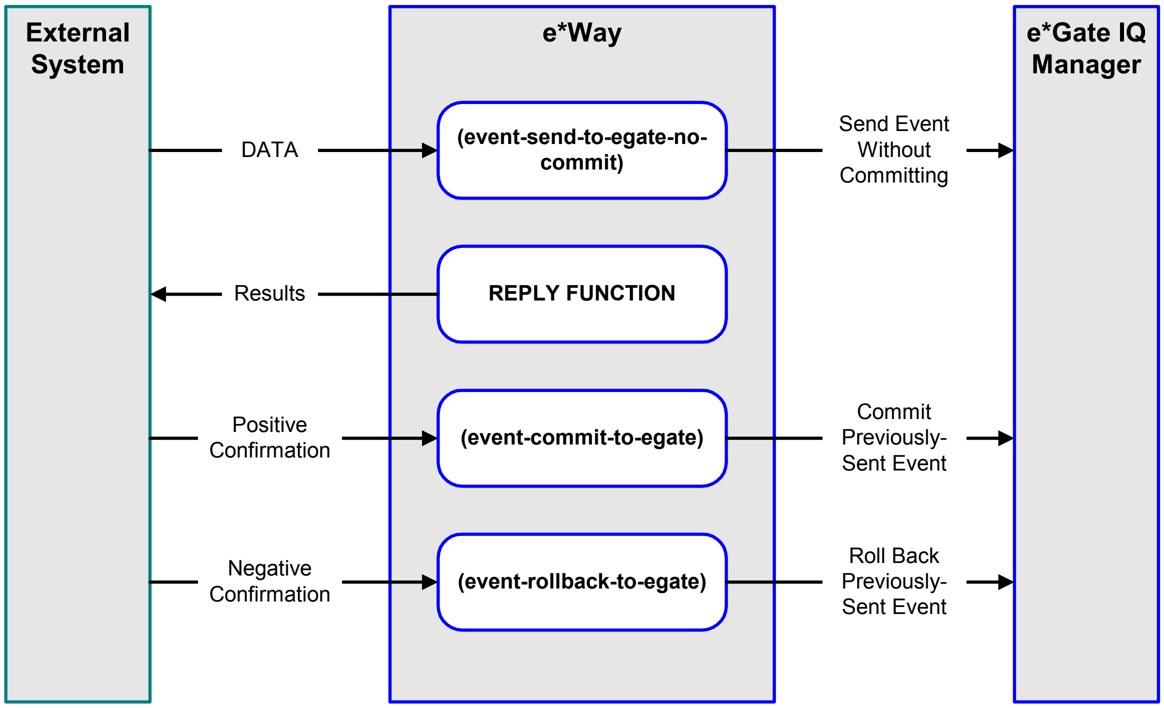
*Send Events to e*Gate* can be implemented using any of the following Monk functions:

- **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 39):

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

Figure 39 Send Event to e*Gate with Confirmation

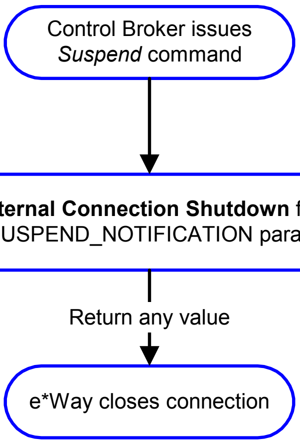

External System
e*Way
e*Gate IQ Manager
DATA
(event-send-to-egate-no-commit)
Send Event Without Committing
Results
REPLY FUNCTION
Positive Confirmation
(event-commit-to-egate)
Commit Previously-Sent Event
Negative Confirmation
(event-rollback-to-egate)
Roll Back Previously-Sent Event

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

Disconnect from External Process

Figure 40 illustrates how the e*Way disconnects from the external system, using the **External Connection Shutdown Function**.

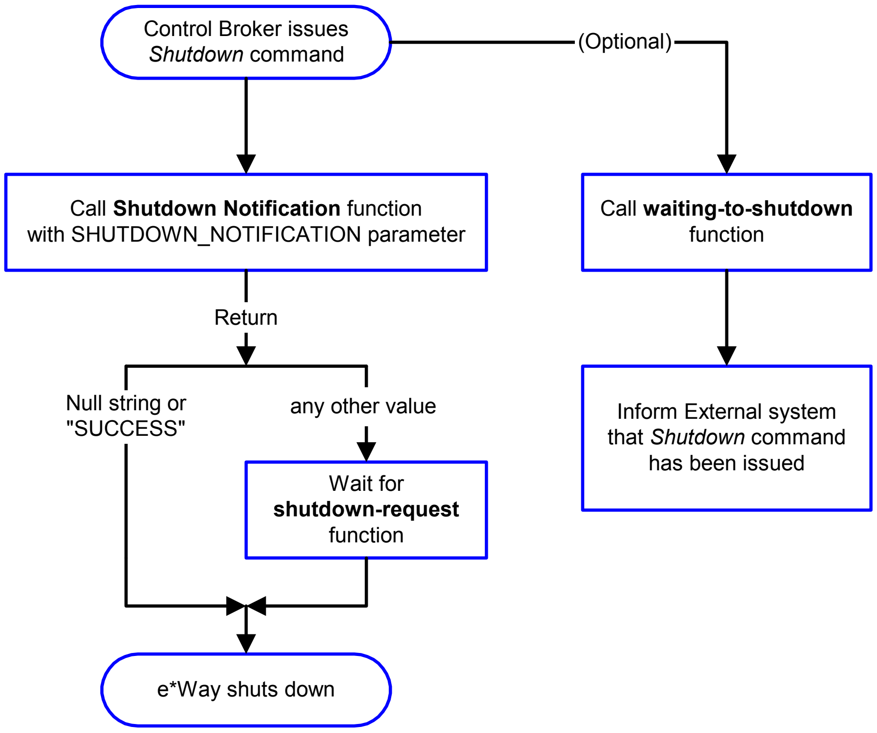
Figure 40 Disconnect Process


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

Shutdown Process

Figure 41 illustrates how the e*Way shuts itself down, using the **Shutdown Command Notification Function**.

Figure 41 Shutdown Process


Control Broker issues *Shutdown* command
(Optional)
Call **Shutdown Notification** function with SHUTDOWN_NOTIFICATION parameter
Call **waiting-to-shutdown** function
Return
Null string or "SUCCESS"
any other value
Wait for **shutdown-request** function
Inform External system that *Shutdown* command has been issued
e*Way shuts down

Chapter 6

Configuration Parameters

This chapter describes the configuration parameters for the SeeBeyond e*Way Intelligent Adapter for PeopleSoft Message Agent.

6.1 Overview

The e*Way's configuration parameters are set using the e*Way Editor; see **Configuring the e*Way** on page 48 for procedural information. The PeopleSoft Message Agent e*Way's configuration parameters are organized into the following sections. The default configuration is provided in **psoft.def**.

General Settings on page 81

Communication Setup on page 83

Monk Configuration on page 86

PeopleSoft Setup on page 94

6.2 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 journaled for the following conditions:

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

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 exits.

Required Values

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

Forward External Errors

Description

Selects whether or not error messages that are received from the external system beginning with the string **"DATAERR"** are queued to the e*Way's configured queue. See **Exchange Data with External Function** on page 89 for more information.

Required Values

Yes or **No**. The default value, **No**, specifies that error messages not to be forwarded. See **Data Exchange Process** on page 76 for more information about how the e*Way uses this function.

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

Additional Information

- If **Zero Wait Between Successful Exchanges** is set to **Yes** and the **Exchange Data with External Function** returns data, the 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, no exchange data schedule is set and the **Exchange Data with External Function** is never called

See also

Start Exchange Data Schedule on page 84

Stop Exchange Data Schedule on page 85

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**.

6.4 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. The default value is **monk_scripts/common**.

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/psoft**.

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. There is no default value for this parameter.

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 value is **ps-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. There is no default value for this parameter.

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. There is no default value for this parameter.

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 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 **ps-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. The function accepts no 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 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. There is no default value for this parameter.

Note: *This parameter is optional and may 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 **ps-on-ack**.

Note: *This parameter is **conditional** and must be supplied only if the* **Exchange Data with External Function** *is set to a non-zero value.*

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**.
- This function can return data to be queued, but the e*Way does *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.*

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 **ps-on-nak**.

Note: *This parameter is **conditional** and must be supplied only if the* **Exchange Data with External Function** *is set to a non-zero value.*

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 does *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. The default value is **ps-finish**.

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.

6.5 PeopleSoft Setup

The parameters in this section help you to set up the required information for the PeopleSoft Message Agent e*Way.

Host Name or IP Address of the Application Server

Description

Specifies the host name or the IP address of the machine where the PeopleSoft Application Server is running.

Required Values

Host name or IP address.

Port Number for the Application Server

Description

Specifies the port number to which the PeopleSoft Application Server is listening.

Required Values

An integer between **1** and **65,535**. The default is **7010**.

Operator ID

Description

Specifies the user name of the operator.

Required Values

User name.

Operator Password

Description

Specifies the password for the operator.

Required Values

Password.

Log File for Failed Records

Description

If the e*Way is not able to process a message received from the e*Gate IQ, the message is logged in this file. Specifies a file path that is relative to the e*Gate directory.

Required Values

File name and path.

Maximum Number of Failure Allowed Before Erroring Out

Description

Specifies the maximum number of failures to record before the e*Way exits with an error.

Required Values

An integer from **1** through **65,535**. The default is **1**.

Chapter 7

API Functions

This chapter describes the various Monk functions used by the SeeBeyond e*Way Intelligent Adapter for PeopleSoft Message Agent.

7.1 Overview

Architecturally, the e*Way can be viewed as a layered structure, built upon the e*Way kernel as a foundation. Each layer contains Monk scripts and/or functions, and makes use of lower-level Monk functions residing in the layers beneath. End users primarily make use of the highest-level functions, which reside in the topmost layer. These are provided to simplify the implementation process by minimizing the need for low-level programming.

The PeopleSoft Message Agent e*Way functions fall into the following categories:

- **High-level Monk Functions** on page 97
- **Low-level Monk Functions** on page 104
- **Generic e*Way Functions** on page 119

Monk functions installed with the e*Way, but not listed in any of the above categories, are used internally and should be used only by SeeBeyond personnel.

Note: *The function names beginning with* **psmsg-** *correspond to those PSMSG APIs documented in the PeopleSoft PeopleBooks documentation. For more details on PeopleSoft Message Agent APIs see PeopleBooks—PeopleTools—Integration Tools—Message Agent—Message Definition.*

7.2 High-level Monk Functions

The following Monk functions are provided to interface with the Message Agent using PeopleSoft Event Type Definitions. The first five functions listed below are used as the default values in the e*Way configuration as indicated in **Chapter 6**.

Most of these functions return **status_code**, **psmsg-ok** on success, and various numeric codes on failure. Some functions appear in groups. Call the first function and check the status code returned. If the first function succeeds, call the rest of the group to get the actual values and/or strings.

The PeopleSoft Message Structure Monk API functions are:

ps-startup on page 97

ps-finish on page 98

ps-connect on page 98

ps-on-ack on page 99

ps-on-nak on page 99

psmsg-init on page 100

psmsg-readrows on page 100

psmsg-updaterows on page 101

sap_date_conv on page 102

ps-startup

Description

Initializes the interface startup function for the PeopleSoft Message Agent e*Way, used as the default value for the **Startup Function** on page 88.

Signature

```
(ps-startup)
```

Parameters

None.

Returns

Returns a **string** representing one of the following status messages:

Name	Description
SUCCESS	Successfully started.
FAILURE	Failed to start.

Throws

None.

Notes

This function initializes the PeopleSoft Message Agent Monk Library with **psmsg-init** and connects to the PeopleSoft system specified in the e*Way configuration file. This function fails the e*Way if it fails to connect to the target PeopleSoft system.

Location

```
ps.monk
```

ps-finish

Description

Disconnects the interface from the PeopleSoft Message Agent e*Way, used as the default value for the **Shutdown Command Notification Function** on page 93.

Signature

```
(ps-finish)
```

Parameters

None.

Returns

None.

Throws

None.

Location

```
ps.monk
```

ps-connect

Description

The interface reconnect function for the PeopleSoft Message Agent e*Way, used as the default value for the **External Connection Establishment Function** on page 90.

Signature

```
(ps-connect)
```

Parameters

None.

Returns

Returns an **integer** representing one of the following status messages:

Name	Description
UP	The e*Way successfully connected to the Message Agent.
DOWN	The e*Way failed to connect to the Message Agent.

Throws

Code	Description
0x00751012	Failure to connect to Message Agent.

Location

```
ps.monk
```

ps-on-ack

Description

Acknowledges a successful connection to the interface for the PeopleSoft Message Agent e*Way, used as the default value for the **Positive Acknowledgment Function** on page 91.

Signature

```
(ps-on-ack)
```

Parameters

None.

Returns

None.

Throws

None.

Location

```
ps.monk
```

ps-on-nak

Description

Acknowledges that the PeopleSoft Message Agent e*Way has a problem with the connection to the interface, used as the default value for the **Negative Acknowledgment Function** on page 92.

Signature

```
(ps-on-nak)
```

Parameters

None.

Returns

None.

Throws

None.

Location

```
ps.monk
```

psmsg-init

Description

Initializes the PeopleSoft Message Agent Monk library.

Signature

```
(psmsg-init)
```

Parameters

None.

Returns

Undefined.

Throws

None.

Notes

This function is automatically invoked within **ps-startup**.

psmsg-readrows

Description

Reads data records from a PeopleSoft database table. This function takes a field-path to a PeopleSoft message structure as the first input parameter. It starts a PeopleSoft message with the Message Agent. It then passes to the Message Agent the name-value pairs of all the search-key fields in the SEARCHKEY segment, and of all the key fields in the first DATA segments of the structure.

After all the key fields are set, the function processes the PeopleSoft message with the Message Agent to poll the data records out of the PeopleSoft database table. With the second input parameter (Boolean) being TRUE, the value sub-nodes of all the fields in the DATA segment of the message structure are filled with the resulting data fields. This is done field by field and record by record, with each DATA segment containing a single data record in the result.

Note: *Not all the data records returned by the Message Agent contain the specified keys (the values of the search-key fields and the key fields in the first DATA segment). To filter out data records that do not contain the specified keys, set the second parameter* **get_all_records_bool** *to* **FALSE***.*

Signature

```
(psmsg-readrows node_path get_all_records_bool)
```

Parameters

Name	Type	Description
node_path	Node path	The Field-path to the message structure.
get_all_records_bool	Boolean	A process that filters out data records that do not contain the specified keys.

Returns

Returns an **integer** representing one of the following status messages:

PSMSG_OK	Success
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_ERROR	General runtime error
PSMSG_NOTLODED	Message Agent is not present.
PSMSG_BADCONTEXT	Bad connection
PSMSG_NOTFOUND	Requested definition was not found.
PSMSG_NOMATCHINGROWS	No matching rows
PSMSG_NOMOREAGENTS	No more free agents

Throws

None.

psmsg-updaterows

Description

Updates/inserts data records into PeopleSoft tables. This function takes a field-path to a PeopleSoft message structure as an input parameter. It starts a PeopleSoft message with the Message Agent and passes to the Message Agent the name-value pairs of all the search-key fields in the SEARCHKEY segment of the structure.

Then, for each of the DATA segments, the function passes to the Message Agent the name-value pairs of all the key fields and of all the non-key fields whose map sub-node has a value **Y**.

Finally, it processes the PeopleSoft message with the Message Agent. If any of the above steps fail, the function aborts and an appropriate error code is returned. Otherwise, the status code PSMSG_OK is returned.

Signature

```
(psmsg-updaterows message_struct path)
```

Parameters

Name	Type	Description
message_struct	Message structure	The PeopleSoft message structure to be updated.
path	Field path	The Field path name used to locate the subject in PeopleSoft message structure.

Returns

Returns an **integer** representing one of the following status messages:

PSMSG_OK	Success
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_ERROR	General runtime error
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Bad connection
PSMSG_NOTFOUND	Requested definition was not found.
PSMSG_NOMATCHINGROWS	No matching rows
PSMSG_NOMOREAGENTS	No more free agents

Throws

None.

sap_date_conv

Description

Converts SAP date format **yyyymmdd** to PeopleSoft date format **MM-dd-yyyy**.

Signature

```
(sap_date_conv sap_date_string)
```

Parameter

Value	Type	Description
sap_date_string	String	The Date in SAP format to be converted.

Returns

Returns a **ps_date_string** (date in PeopleSoft format).

Throws

None

Notes

This function uses the delimiter set of the **sap_data** structure. If the **sap_data** structure changes, this function may need to be modified in order to function correctly.

7.3 Low-level Monk Functions

The Monk APIs described in this section form part of the PeopleSoft Message Agent e*Way. They can be used to extend the functionality of the PeopleSoft Message Agent e*Way through the use of Monk Extension Scripts. These low-level Monk API functions are:

Note: *The functions described in this section can only be used by the functions defined within the e*Way's configuration file. None of the functions is available to Collaboration Rules scripts executed by the e*Way.*

psmsg-conn-handle?

Description

Checks if the provided Monk object is a valid handle to the PeopleSoft connection.

Signature

```
(psmsg-conn-handle? psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	Any	The Monk object provided.

Returns

Boolean

Returns **#t** (true) if the Monk object **psmsg_conn_handle** is of type *psmsg-conn-handle* and is a valid handle to a PeopleSoft Message Agent connection; otherwise, returns **#f** (false).

Throws

None.

psmsg-connect

Description

The default interface reconnection function. It tries to connect to the target PeopleSoft system.

Signature

```
(psmsg-connect serverandport operatorid operatorpasswd)
```

Parameters

Name	Type	Description
serverandport	String	The Server name and port number.
operatorid	String	The Operator ID used for connection.
operatorpasswd	String	The Operator password used for connection.

Returns

Returns a **psmsg-conn-handle** object. Upon success, it is a valid connection handle to the PeopleSoft Message Agent system; upon failure, it is an invalid handle. The caller should test the validity using the **psmsg-conn-handle?** function.

Throws

None.

psmsg-disconnect

Description

Shuts down the connection to the PeopleSoft Message Agent.

Signature

```
(psmsg-disconnect psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns an **integer** representing one of the following status messages:

PSMSC_OK	Success. The location is set to the first field.
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE
PSMSG_NOTFOUND	Output field list was empty.

Throws

None.

psmsg-findfirstfield

Description

Locates the first field in the output message field list and sets the current position there.

Signature

```
(psmsg-findfirstfield psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns an **integer** representing one of the following status messages:

PSMSC_OK	Success. The location is set to the first field.
PSMSG_ERROR	General error

PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE
PSMSG_NOTFOUND	Output field list was empty.

Throws

None.

Additional Information

After the DLL file processes a message with **psmsg-processmessage**, it builds a list of the output fields defined in the Application Designer Message Definition. The **psmsg-findfirstfield** API sets the current list position to the beginning of the list.

Use **psmsg-getfieldname** and **psmsg-getfieldvalue** to extract the field name and value, then use **psmsg-findnextfield** to move to the next field.

psmsg-findnextfield

Description

Locates the next field in the output message field list and sets the current list position to that field.

Signature

```
(psmsg-findnextfield psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns an **integer** representing one of the following status messages:

PSMSG_OK	Success
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE
PSMSG_NOTFOUND	There was no next field in the output list.

Throws

None.

Additional Information

Use **psmsg-findnextfield** in conjunction with **psmsg-findfirstfield** to visit each field in the output message field list. For each field, use **psmsg-getfieldname**, then **psmsg-getfieldvalue** to extract the field name and value.

psmsg-findnextoutputrow

Description

Locates the next row of output for a message that maps data from a level one scroll.

Signature

```
(psmsg-findnextoutputrow psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns an **integer** representing one of the following status messages:

PSMSG_OK	Success
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE passed to PSMsg.
PSMSG_NOTFOUND	There are no more rows of data in the scroll.

Throws

None.

Additional Information

The Message Agent DLL can map data to and from fields that appear inside a level 1 scroll on a panel. When a scroll includes more than one row of data, the Message Agent creates a “table” of output values. This API moves the current position to the next row in the table, enabling the user to retrieve data from the fields in that row.

When retrieving data, the initial current row is the first row in the scroll. Thus, the first use of **psmsg-findnextoutputrow** advances the current position to the second row.

This function is only available for level 1 scrolls.

Note: *For more information on level 1 scrolls, see the PeopleSoft documentation, People Books.*

psmsg-geterrorexplaintext

Description

Gets the detailed description associated with a PeopleSoft error message.

Signature

```
(psmsg-geterrorexplaintext psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns a Monk **string** that explains the last error that occurred in the PeopleSoft Monk DLL.

Throws

None.

Additional Information

When a PeopleSoft application displays an error message, the message box includes an **Explain** button. The user can select the button to view a more detailed description of the problem. This API enables the Message Agent DLL to retrieve the same detailed description. It retrieves the descriptive text associated with the first error the Message Agent encountered while processing a message.

Use **psmsg-geterrorexplaintext** in conjunction with **psmsg-geterrortext**. If the **psmsg-geterrortext** message includes a message set and message number at the end, **psmsg-geterrorexplaintext** returns the "explain" text for that set and message.

Note: *Because of a known limitation with the PeopleSoft Message Agent API, this function works correctly the first time it is invoked during a connection session, but fails to get the error text in subsequent invocations during the same session.*

psmsg-geterrorfieldname

Description

Returns the name of the field where the last error occurred.

Signature

```
(psmsg-geterrorfieldname psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns the **name** of the field where the error occurred.

Throws

None.

Note: *Because of a known limitation with the PeopleSoft Message Agent API, this function works correctly the first time it is invoked during a connection session, but fails to get the error field name in subsequent invocations during the same session.*

psmsg-geterrorrecordname

Description

Returns the name of the record where the last error occurred.

Signature

```
(psmsg-geterrorrecordname psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns the **name** of the record where the last error occurred.

Throws

None.

Additional Information

When the data entered by the Message Agent into a panel fails an online edit, use this function to determine which record definition contains the field the Message Agent was unable to update. If more than one field failed an edit, **psmsg-geterrorrecordname** retrieves the record definition name for the first failure encountered. The retrieved record name is buffered within the Message Agent DLL.

To determine which record field failed the edits, use **psmsg-geterrorfieldname**.

Note: *Because of a known limitation with the PeopleSoft Message Agent API, this function works correctly the first time it is invoked during a connection session, but fails to get the record name in subsequent invocations during the same session.*

psmsg-geterrortext

Description

Retrieves the error text for the first error encountered while processing a message.

Signature

```
(psmsg-geterrortext psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns a Monk **string** that describes the last error that occurred in the PeopleSoft Monk DLL.

Throws

None.

Additional Information

When the Message Agent DLL encounters an error, it records a textual description of the error and goes into an error state. Once in the error state, the Message Agent responds only to **startevent**, **psmsg-geterrortext**, **psmsg-geterrorexplaintext** and **psmsg-disconnect**.

Note: *Because of a known limitation with the PeopleSoft Message Agent API, this function works correctly the first time it is invoked during a connection session, but fails to get the error text in subsequent invocations during the same session.*

psmsg-getfieldname

Description

Returns the name of the current field as a Monk string.

Signature

```
(psmsg-getfieldname psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns the name of the current field as a Monk **string**.

Throws

This function throws a Monk Exception Code 0033 on failure.

Additional Information

After the Message Agent DLL processes a message with **psmsg-processmessage**, it builds a list of the output message fields defined in the Application Designer Message Definition. The **psmsg-getfieldname** API retrieves the name of the current field in that list. The name of the string is buffered within the Message Agent DLL.

psmsg-getfieldvalue

Signature

```
(psmsg-getfieldvalue psmsg_conn_handle)
```

Description

Returns the value of the current field as a Monk string.

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns the value of the current field as a Monk **string**.

Throws

None.

Additional Information

After the Message Agent DLL processes a message with **psmsg-processmessage**, it builds a list of the output message fields defined in the Application Designer Message Definition.

psmsg-getreplyoption

Description

Returns the reply option set by **psmsg-processmessage**.

Note: *To use this API, you must first call* **psmsg-processmessage**.

Signature

```
(psmsg-getreplyoption psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns an **integer** representing one of the following status messages (reply_option_num):

PSMSG_NOREPLY	No action
PSMSG_REPLY	Send confirmation to the originator of the message.
PSMSG_FORWARD	Return a copy of the new field values to the originator.
PSMSG_MONKERROR	A Monk error occurred.

Throws

None.

Additional Information

The message reply option indicates how PeopleSoft expects the calling program to respond to the originator of the message.

As shown in the table above, the Message Definition specifies a reply option of **No Reply**, **Reply**, or **Forward**.

The expected response is: **Reply** which means to send a confirmation to the original sender. **Forward** means to forward the field values that were actually sent back to the sender.

psmsg-processmessage

Description

Triggers message processing after all input message fields have been set.

Signature

```
(psmsg-processmessage psmsg_conn_handle)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Returns

Returns an **integer** representing one of the following status messages:

PSMSG_OK	Success
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE passed to PSMsg function.
PSMSG_NOMATCHINGROWS	Message Agent attempted to locate a record in Update/Display mode, but could not locate it.

Throws

None.

Additional Information

After you have set all input message fields with **psmsg-setfield**, **psmsg-processmessage** opens the panel, sets the values for the input fields, saves the panel data, and prepares a list of output values (if any).

Use **psmsg-getreplyoption** to determine the reply option set by **psmsg-processmessage**.

If an error occurs, the Message Agent records the text of the error and enters an error state.

psmsg-setfield

Description

Sets the value of a message field to be mapped into a panel buffer when the message is processed.

Signature

```
(psmsg-setfield psmsg_conn_handle fieldname fieldvalue)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.
fieldname	String	The name of the message field to be mapped.
fieldvalue	String	The value of the message field to be set.

Return Values

Returns an **integer** representing one of the following status messages:

PSMSG_OK	Success
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE passed to PSMsg function.
PSMSG_NOTFOUND	The requested field was not found.

Throws

None.

Additional Information

The Message Definition contains a mapping between message input fields and panel record fields. To update a panel record field, the PeopleSoft Message Agent e*Way passes message field names and values to the Message Agent. The Message Agent buffers these values until you use **psmsg-processmessage**.

psmsg-setoptions

Description

Takes a PsMsg Handle and an integer, which is a Message Agent Option flag.

Signature

```
(psmsg-setoptions psmsg_conn_handle PSMSG_OPT_*)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.

Name	Type	Description
PSMSG_OPT_*	constant	Defined in the psmsg-init.monk, and passed to psmsg-setoptions.

Returns

Returns an **integer** (Status Code), as defined in **psmsg-init**, that represents one of the following status messages:

PSMSG_OK	Success
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE passed to PSMsg function.
PSMSG_NOMATCHINGROWS	Message Agent attempted to locate a record in Update/Display mode, but could not locate it.

Throws

None.

Additional Information

The PSMSG_OPT_* constants defined in **psmsg-init** that can be used to pass the second parameter to **psmsg-setoptions** are:

Message Agent Option Flag Constants	Description
PSMSG_OPT_RESET	Resets all options.
PSMSG_OPT_CHANGEMODEUPDATE	Changes panel mode from add to update, if level 0 record was found.
PSMSG_OPT_NOGUI	Disables the GUI.
PSMSG_OPT_NOFIELDEDITS	Disables Field edits. This stops the checking for valid values for fields, so use it when you are reading validated values and pushing it back into application tables.
PSMSG_OPT_RESET	Orders the output lists by how they appear in Field Mapping.

Note: *Multiple Option Flags can be* **OR***ed and passed to* **psmsg-setoptions**.

psmsg-startmessage-verified

Description

Notifies the Message Agent that the caller is ready to send and receive data for a defined message.

Signature

```
(psmsg-startmessage-verified psmsg_conn_handle activity_name
    message_def_name)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg-conn-handle	The Message Agent connection handle used for connection.
activity_name	String	The name of the activity to occur.
message_def_name	String	The name of the defined message to be sent.

Returns

Returns an **integer** representing one of the following status messages:

PSMSG_OK	The Message Agent opened the Message Definition.
PSMSG_ERROR	General error
PSMSG_MONKERROR	A Monk error occurred.
PSMSG_NOTLOADED	Message Agent is not present.
PSMSG_BADCONTEXT	Invalid PSMSGHANDLE passed to PSMsg function.
PSMSG_NOTFOUND	The requested Message Definition was not found.

Throws

None.

Additional Information

The **psmsg-startmessage-verified** function tells the Message Agent DLL status messages which Application Designer Message Definition to use for mapping fields. Use this function before setting the input message fields with **psmsg-setfield** or processing the message with **psmsg-processmessage-verified**. The user can start multiple messages during a single connection to a Message Agent.

If the Message Agent cannot open the Message Definition, the Message Agent goes into an error state until the next message is specified.

When someone creates a Message Definition in Application Designer, they have the option of requiring that the calling program verify the originator of the message by means of an electronic signature or some other method. If a message requires originator verification, but there is no way for the calling program to verify the originator, the Message Agent does not process the message.

psmsg-startmessage-unverified

Description

Allows the Message Agent to send and receive data for a defined message whether or not the caller is ready.

Signature

```
(psmsg-startmessage-unverified psmsg_conn_handle activity_name
    message_def_name)
```

Parameters

Name	Type	Description
psmsg_conn_handle	psmsg_conn_handle	The Message Agent connection handle used for connection.
activity_name	String	The name of the activity to occur.
message_def_name	String	The name of the defined message to be sent.

Returns

Returns an **integer** representing one of the following:

PSMSG_OK	Success
PSMSG_MONKERROR	A Monk error occurred.

Throws

None.

7.4 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 119

event-rollback-to-egate on page 120

event-send-to-egate on page 120

event-send-to-egate-ignore-shutdown on page 121

event-send-to-egate-no-commit on page 121

get-logical-name on page 122

insert-exchange-data-event on page 122

send-external-up on page 123

send-external-down on page 123

shutdown-request on page 124

start-schedule on page 124

stop-schedule on page 125

waiting-to-shutdown on page 125

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 121

event-send-to-egate-no-commit on page 121

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 120

event-send-to-egate-no-commit on page 121

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 119

event-rollback-to-egate on page 120

event-send-to-egate on page 120

event-send-to-egate-ignore-shutdown on page 121

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 83

Zero Wait Between Successful Exchanges on page 83

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.

Index