SeeBeyond ICAN Suite

Siebel EAI e*Way Intelligent Adapter User's Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

Monk Version

SEEBEYOND®

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.

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

Version 20050406040424.

Contents

Preface 8

Intended Reader 8

Organization 8

Nomenclature 9

Online Viewing 9

Writing Conventions 9

Additional Information 10

Chapter 1

Introduction 11

Overview 11

Communicating with Siebel EAI 12

e*Way Operation 13

e*Gate-to-Siebel (HTTP) Mode 13

Siebel-to-e*Gate (MUX) Mode 14

SeeBeyond Workflow Templates 15

e*Way Components 16

Supported Operating Systems 17

Chapter 2

Installation 18

System Requirements 18

External System Configuration 18

External System Requirements 19

Installing the e*Way 20

Installation Procedure 20

Subdirectories and Files 22

Environment Configuration 23

Web Server Setup 24

Installing the Siebel Web Server Extension 24
Verifying SWSE Operation 26
Setting Up the MUX ASP 27

Optional Example Files 28
Installation Procedure 29
Subdirectories and Files 30

Chapter 3

Configuration Parameters 31

Overview 31

General Settings (HTTP) 32
Journal File Name 32
Max Resends Per Message 32
Max Failed Messages 32
Forward External Errors 33

Communication Setup 34
Exchange Data Interval 34
Zero Wait Between Successful Exchanges 34
Start Exchange Data Schedule 35
Stop Exchange Data Schedule 35
Down Timeout 36
Up Timeout 36
Resend Timeout 36

Monk Configuration 37
Specifying Function or File Names 37
Specifying Multiple Directories 37
Load Path 37
Additional Path 37
Auxiliary Library Directories 38
Monk Environment Initialization File 38
Startup Function 39
Process Outgoing Message Function 39
Exchange Data with External Function 40
External Connection Establishment Function 41
External Connection Verification Function 42
External Connection Shutdown Function 42
Positive Acknowledgment Function 43
Negative Acknowledgment Function 43
Shutdown Command Notification Function 44

HTTP Configuration 46
Request 46
Timeout 46
URL 46
User Name 47
Encrypted Password 47
Agent 47
Content-type 47
Request-content 47
Accept-type 48

HTTP Proxy Configuration 49
Use Proxy Server 49
User Name 49
Encrypted Password 49

Server Address 49
Port Number 50

HTTP SSL Configuration 51
Trusted CA Certificated Directory 51
Use Client Certificate Map 51
Client Certificate Map File 51

Siebel Configuration 53
SWEExtSource 53
SWEExtCmd 53
User Name 53
Encrypted Password 53

General Settings (MUX) 54
Request Reply IP Port 54
Push IP Port 54
Rollback if No Clients on Push Port 54
Wait for IQ Ack 54

Chapter 4

e*Way Setup 56

Overview 56

Setting Up the e*Way 57
Creating the e*Way 57
Modifying e*Way Properties 58
Configuring the e*Way 59
Using the e*Way Editor 60
Section and Parameter Controls 61
Parameter Configuration Controls 61
Command-line Configuration 62
Getting Help 62
Changing the User Name 63
Setting Startup Options or Schedules 63
Activating or Modifying Logging Options 65
Activating or Modifying Monitoring Thresholds 66

Troubleshooting the e*Way 67
Configuration Problems 67
System-related Problems 68

Chapter 5

Operational Overview 69

Siebel EAI e*Way Architecture 69
e*Gate-to-Siebel (HTTP) Mode 69
Siebel-to-e*Gate (MUX) Mode 70

Collaborations 71

Basic e*Way Processes (HTTP Mode Only) 72
Initialization Process 73
Connect to External Process 74

Data Exchange Process 75
Disconnect from External Process 78
Shutdown Process 78

Chapter 6

System Implementation 79

Contents 79
Pre-Implementation Tasks 79

Overview 80
General Sequence 80
e*Gate to Siebel 81
Siebel to e*Gate 82
Viewing e*Gate Components 82

SeeBeyond Workflow Templates 83
Overview 83
Siebel XML Messages 88
Format 88
Examples 88
Importing SeeBeyond Workflow Templates 90
Modifying SeeBeyond Workflow Templates 92
Setting Up SeeBeyond Workflow Processes 93

Creating a Schema 97

Generating the Integration Object DTD 98

Verifying the Integration Object DTD 98

Creating Event Type Definitions 101
Using the ETD Editor's Build Tool 101
Assigning ETDs to Event Types 103

Defining Collaborations 104

Creating Intelligent Queues 105

Using the e*Way 106
Connecting to Siebel 106
Specifying the Business Service 107
The Siebel Workflow Process 107
e*Gate-to-Siebel Example 108
Siebel-to-e*Gate Example 109

Sample Schema 110
e*Gate-to-Siebel Example 110
Collaborations 112
Siebel-to-e*Gate Example 115
Collaborations 116

Chapter 7

API Functions 117

Overview 117

Siebel HTTP Monk Functions 118

siebel-http-ack 118
siebel-http-connect 119
siebel-http-exchange 119
siebel-http-init 120
siebel-http-nack 121
siebel-http-notify 122
siebel-http-outgoing 122
siebel-http-process 123
siebel-http-shutdown 124
siebel-http-startup 125
siebel-http-verify 125

Generic e*Way Functions 127

event-commit-to-egate 127
event-rollback-to-egate 128
event-send-to-egate 128
event-send-to-egate-ignore-shutdown 129
event-send-to-egate-no-commit 129
get-logical-name 130
insert-exchange-data-event 130
send-external-up 131
send-external-down 131
shutdown-request 132
start-schedule 132
stop-schedule 133
waiting-to-shutdown 133

e*Gate API Kit Functions 134

Index 135

Preface

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

P.1 Intended Reader

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

- Operation and administration of the appropriate operating systems (see **Supported Operating Systems** on page 17)
- Windows-style GUI operations
- Siebel 2000 and Siebel EAI concepts and operation

P.2 Organization

This User's Guide is organized into two parts. The first part consists of Chapters 1-4, which introduce the e*Way and describe the procedures for installing and implementing a working system incorporating the e*Way. Chapter 6 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 that 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 architecture and internal functionality of the e*Way. 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 Siebel EAI is frequently referred to as the Siebel EAI e*Way, or simply the e*Way.

P.4 Online Viewing

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, or attributes within a function signature, are set in italics 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 Information

Many of the procedures included in this User's Guide are described in greater detail in the *e*Gate Integrator User's Guide*.

You may want to refer to the following Siebel documents prior to installation:

- *Siebel System Requirements and Supported Platforms*
- *Client Installation and Administration Guide*
- *Server Installation Guide*
- *Upgrade Guide*

Chapter 1

Introduction

This chapter provides a brief introduction to the SeeBeyond Monk e*Way Intelligent Adapter for Siebel EAI.

1.1 Overview

The Siebel EAI e*Way is a software interface that enables the e*Gate system to exchange Events (messages) with Siebel EAI via a Web server. The e*Way communicates with Siebel via open standards such as HTTP and XML, and incorporates elements of two other SeeBeyond e*Ways:

- HTTP e*Way
- e*Gate API Kit

The e*Gate-to-Siebel component of the Siebel EAI e*Way uses the HTTP e*Way to forward Siebel XML messages, while the Siebel-to-e*Gate component uses the MUX ASP from the API Kit to receive Siebel XML messages. Common elements of both e*Ways are installed automatically as part of the Siebel EAI e*Way installation.

1.2 Communicating with Siebel EAI

A traditional Siebel infrastructure is composed of four basic components:

- A **Siebel Database** to hold the data.
- A **Siebel Gateway Server** to store enterprise configuration.
- At least one **Siebel Application Server** to manage components of Siebel Applications.
- A **Siebel Client** to interface with the user.

To make use of the Web, Siebel adds another component: the Siebel Web Server Extension (SWSE). This is a shared library that runs inside a Web server to direct user requests to appropriate Application Object Manager service via the Siebel Web Engine (SWE). The Application Object Manager is a component in the Siebel Server, which passes Siebel object definitions and data between the database and the SWE. These object definitions provide the application logic and enable the user to interact with the database.

Communication with Siebel EAI also involves the **Microsoft Internet Information Service (IIS)**. The Siebel Web Engine application installs the Siebel Web Server Extension (SWSE) into the IIS.

Internally, Siebel EAI executes the Transport, Business Service and Workflow in the Business Integration Manager (BIM). BIM provides the development and run-time tools to configure and deploy integration between the Siebel EAI system and other applications. It includes the following components, which are used by the Siebel EAI e*Way in the manner indicated:

- **Siebel Integration Objects** (to generate the ETD)
- **Transport Adapters** (to send and receive messages)
- **Business Service** (to start the workflow)
- **Workflow Process Designer** (to convert XML messages and update Siebel)
- **EAI Siebel Adapter** (to populate the Siebel database)

The workflow process uses two Siebel EAI Toolkit components: **EAI XML Converter** and **EAI Siebel Adapter**. The EAI XML Converter uses the **XML to Property Set** method to convert the Siebel XML message to a property set format that can be used by EAI Siebel Adapter to query, insert, update, or delete the Business Object. In case of a query, the EAI XML Converter converts the property set back to an XML message.

1.3 e*Way Operation

1.3.1 e*Gate-to-Siebel (HTTP) Mode

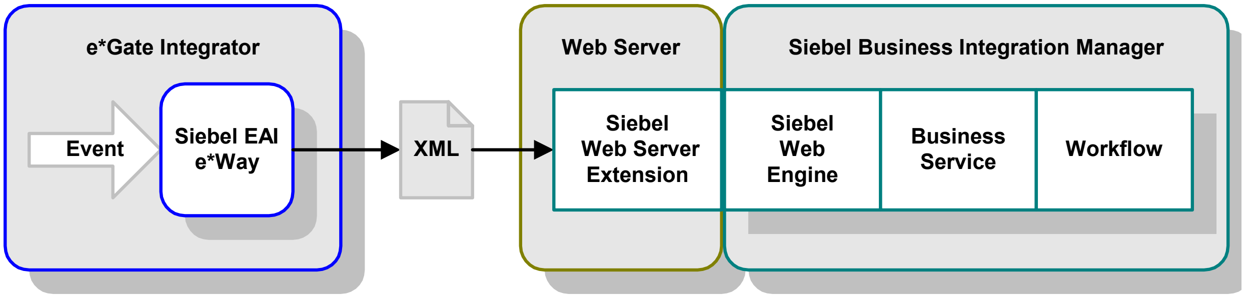
The Siebel EAI e*Way acts as a Web browser, and uses HTTP to forward a Siebel XML-formatted Event to Siebel. It also specifies one of the following actions to be performed on the XML message:

- Delete
- Insert/Update
- Query

The result is that a corresponding Workflow is executed to process the message. A Siebel Workflow is a customized business application for managing and enforcing business processes.

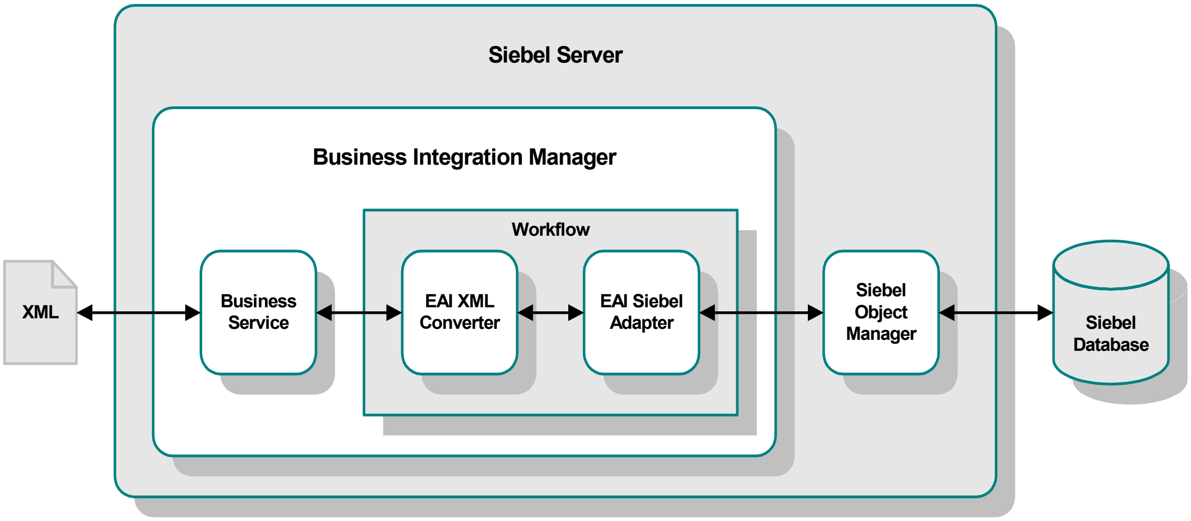
The Siebel EAI e*Way receives an Event, which originated in some external application, from the e*Gate system. The e*Way passes the Event via HTTP to the Web server as a Siebel XML Message. Siebel Web Server Extension invokes the specified Business Service which, in turn, starts an internal Workflow. Figure 1 illustrates the process.

Figure 1 e*Gate-to-Siebel Data Flow



The Workflow invokes the Siebel EAI XML Converter, which converts the information from XML into the Siebel internal format and presents it to the Siebel EAI Adapter. The information then is sent to the Siebel Server via the Siebel Object Manager (see Figure 2).

Figure 2 Siebel Internal Processing (In)

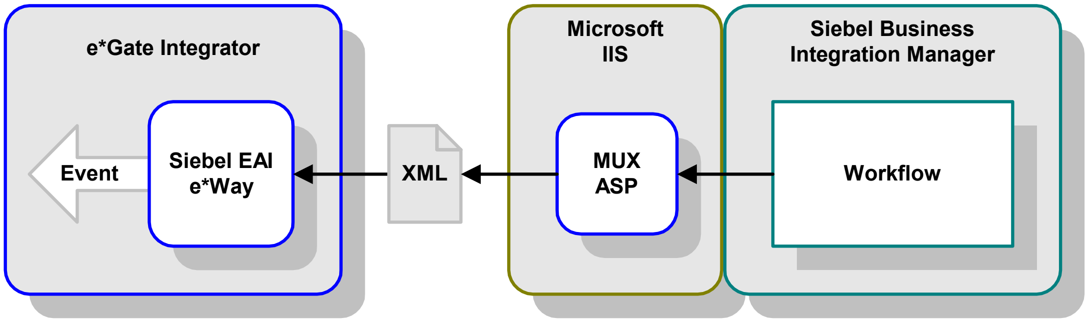


If there is data to be returned, the EAI Siebel Adapter can pass the result to the EAI XML Converter and send the data back to the e*Way as a Siebel XML message.

1.3.2 Siebel-to-e*Gate (MUX) Mode

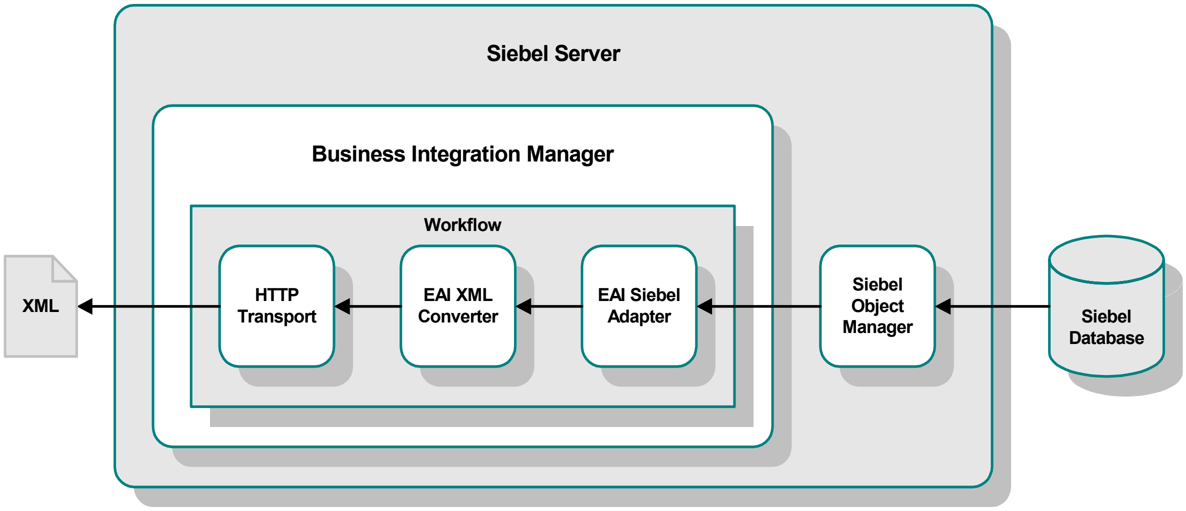
The Siebel EAI e*Way also allows Siebel to send a Siebel XML message to e*Gate via HTTP. The data flow within Siebel is shown in Figure 3. This process is event-driven, and can be initiated, for example, by a feature added to the user interface of the Siebel application.

Figure 3 Siebel-to-e*Gate Data Flow



When a Siebel client initiates a data transfer, the Siebel Object Manager retrieves an Event from the Siebel database and starts a Workflow that resembles the Siebel-inbound workflow in reverse. The EAI Siebel Adapter relays the Event to the EAI XML Converter, which hands it off to a HTTP Transport module. The resulting Siebel XML Message is then sent to the Microsoft IIS, which sends it to e*Gate via the MUX ASP.

Figure 4 Siebel Internal Processing (Out)



1.3.3 SeeBeyond Workflow Templates

A set of SeeBeyond Workflow Templates is included with the Siebel EAI e*Way. These Workflow Templates invoke the necessary Workflow Processes to map the data directly to or from the Siebel database.

Additional information can be found in **SeeBeyond Workflow Templates** on page 83. The referenced section also includes instructions on setting up the Business Service to execute the Workflows (see **Setting Up SeeBeyond Workflow Processes** on page 93). If you are using Siebel 2000 (Japanese), also see the information included in **Using the e*Way** on page 106.

1.4 e*Way Components

The Monk e*Way Intelligent Adapter for Siebel EAI incorporates the following components:

- The Generic e*Way executable, **stcewgenericmonk.exe**, for e*Gate-to-Siebel operation (installed with e*Gate Integrator)
- The e*Gate API Kit executable, **stcewipmp.exe**, for Siebel-to-e*Gate operation
- Ancillary e*Way executable files, associated with the e*Gate API Kit:
    - **stcxipmptest.exe**
    - **stcewmuxipserver.exe**
- Accompanying dynamic load libraries:
    - **stc_ewipmpclnt.dll** (supports C and Java clients for e*Gate API Kit)
    - **stc_ewipmpclntperl.dll** (supports Perl clients for e*Gate API Kit)
    - **stc_form2ssc.exe** (required for HTTP e*Way)
    - **stc_monkhttp.dll** (required for HTTP e*Way)
    - **stc_xipmpclnt.dll** (supports ActiveX clients for e*Gate API Kit)
- Configuration definition files, which you customize as discussed in **Chapter 3**:
    - **stcewsiebelhttp.def** (for e*Gate-to-Siebel operation)
    - **stcewipmp.def** (for Siebel-to-e*Gate operation)
- Monk function scripts, discussed in **Chapter 7**
- Monk Event Type Definition (ETD) Builder, a tool that builds a Monk Event Type Definition from a Siebel Object Definition file
- Example schemas, discussed in **Chapter 6**

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

1.5 Supported Operating Systems

The e*Way Intelligent Adapter for Siebel EAI currently supports the following combinations of operating systems and Siebel versions.

Table 1 English-language Version

Operating System	Siebel 2000	Siebel 7.0.3	Siebel 7.0.4
Windows 2000, Windows XP, and Windows Server 2003	X	X	X
IBM AIX 5.1L and 5.2	-	-	X
Sun Solaris 8 and 9	-	X	X

Table 2 Japanese-language Version

Operating System	Siebel 2000	Siebel 7.0.3	Siebel 7.0.4
Windows 2000, Windows XP, and Windows Server 2003	X	-	X

Table 3 Korean-language Version

Operating System	Siebel 2000	Siebel 7.0.3	Siebel 7.0.4
Windows 2000, Windows XP, and Windows Server 2003	-	-	X

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 6**.

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

2.1 System Requirements

To use the e*Way Intelligent Adapter for Siebel EAI, you need the following:

1. An e*Gate Participating Host.
2. A TCP/IP network connection.
3. Sufficient free disk space on both the Participating Host and the Registry Host to accommodate e*Way files (not including sample schemas):
   - Approximately 2.6 MB on Windows 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, and any external applications performing the processing.*

2.1.1 External System Configuration

Most installations of Siebel applications require some customization of the Data Model to meet the client's specific requirements. This guide assumes that any customization of Siebel required for your implementation has been performed prior to the installation of e*Gate. No special configuration of the Siebel application is required to interface with e*Gate.

2.1.2 External System Requirements

The following software must be installed on the client(s) prior to installation of the e*Way:

- Siebel 2000
  - Siebel Client
  - Siebel Tools

The following software must be installed on the server prior to installation of the e*Way:

- Siebel 2000
  - Gateway Server
  - Siebel Server 6.0
  - Database Server
  - Siebel Web Engine
  - Siebel Tools
- For Windows platforms:
  - Microsoft Internet Information Server 5.0
  - Libraries **stdole2.tlb** and **stdole32.tlb**
  - Active Server Pages (ASP) must be enabled

Refer to **Web Server Setup** on page 24 for information regarding the Web server.

2.2 Installing the e*Way

Note: *It is not necessary to install the e*Gate components on the Siebel Application server; however, the e*Way must have access to the Siebel file system.*

Installation Procedure

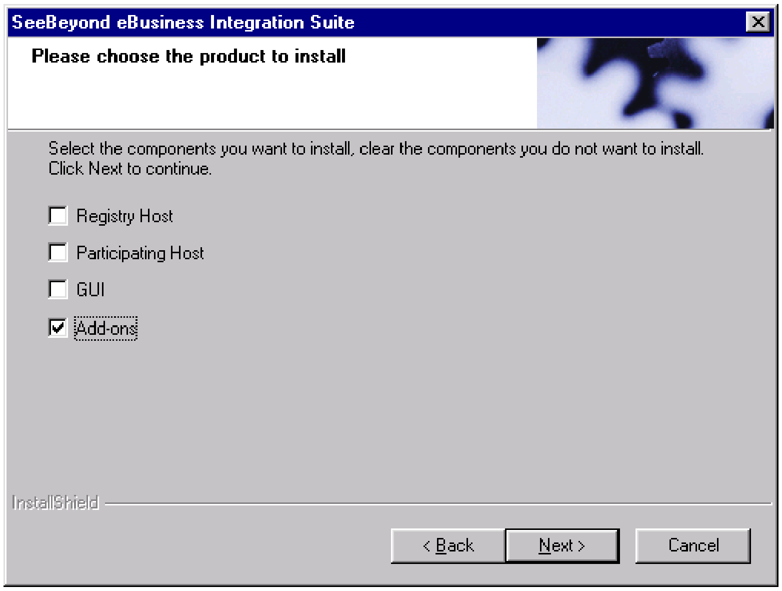
Note: *The installation utility detects and suggests the appropriate installation directory. Use this directory unless advised otherwise by SeeBeyond.*

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



SeeBeyond eBusiness Integration Suite

Please choose the product to install

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

□ Registry Host

□ Participating Host

□ GUI

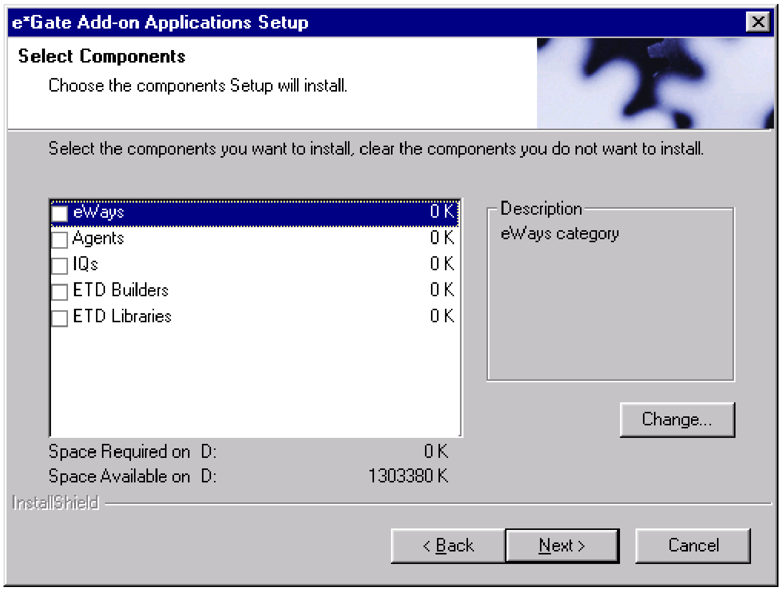
☑ Add-ons

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

Figure 6 Select Components Dialog



e*Gate Add-on Applications Setup

Select Components

Choose the components Setup will install.

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

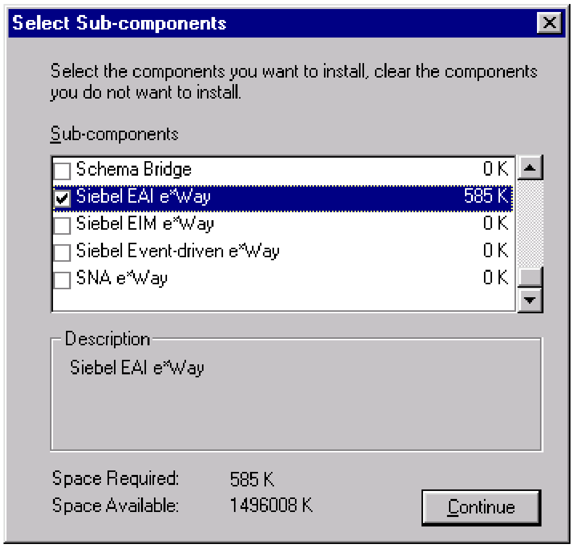
☐ eWays	0 K
☐ Agents	0 K
☐ IQs	0 K
☐ ETD Builders	0 K
☐ ETD Libraries	0 K

Description: eWays category

Space Required on D: 0 K
Space Available on D: 1303380 K

6 When the **Select Sub-components** dialog box appears (see Figure 8), check the **Siebel EAI e*Way**.

Figure 7 Select e*Way Dialog



Select Sub-components

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

Sub-components

☐ Schema Bridge	0 K
☑ Siebel EAI e*Way	585 K
☐ Siebel EIM e*Way	0 K
☐ Siebel Event-driven e*Way	0 K
☐ SNA e*Way	0 K

Description: Siebel EAI e*Way

Space Required: 585 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 Siebel EAI 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 4 Participating Host & Registry Host

Subdirectories	Files
\bin\	stc_ewipmpclnt.dll stc_ewipmpclntperl.dll stc_form2ssc.exe stc_monkhttp.dll stc_xipmpclnt.dll stcewgenericmonk.exe stcewipmp.exe stcewmuxipserver.exe stcxipmptest.exe
\configs\stcewgenericmonk\	stcewsiebelhttp.def
\configs\stcewipmp\	stcewipmp.def
\configs\stcewmuxipserver\	stcewmuxipserver.def
\monk_library\ewsiebelhttp\	siebel-http-ack.monk siebel-http-connect.monk siebel-http-exchange.monk siebel-http-init.monk siebel-http-nack.monk siebel-http-notify.monk siebel-http-outgoing.monk siebel-http-process.monk siebel-http-shutdown.monk siebel-http-startup.monk siebel-http-verify.monk
\monk_library\xml\	event-xml.monk
\monk_scripts\common\	reply.ssc request.ssc sample.ssc siebel-http-outgoing-delete.dsc siebel-http-outgoing-delete-sjis.dsc siebel-http-outgoing-execute.dsc siebel-http-outgoing-execute-sjis.dsc siebel-http-outgoing-insert.dsc siebel-http-outgoing-insert-sjis.dsc

Note that three sample Monk Collaboration Rules script files are included in the installation:

- **siebel-http-outgoing-delete.dsc**
- **siebel-http-outgoing-execute.dsc**
- **siebel-http-outgoing-insert.dsc**

If you are using Siebel 2000 (Japanese), you should use the following alternate files instead:

- **siebel-http-outgoing-delete-sjis.dsc**
- **siebel-http-outgoing-execute-sjis.dsc**
- **siebel-http-outgoing-insert-sjis.dsc**

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

Table 5 Registry Host Only

Subdirectories	Files
\	ewsiebelhttp.ctl

Environment Configuration

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

2.3 Web Server Setup

2.3.1 Installing the Siebel Web Server Extension

To install Siebel Web Server Extension (SWSE)

1. From the Siebel installation media, run **eappweb****setup.exe**, which invokes the installation wizard.
2. Follow the instructions presented by the wizard. Use the naming conventions for your Siebel EAI Application Server.
3. For **Connection Protocol**, specify the default port for an HTTP server, which is **80**.
4. Do *not* use any encryption or compression methods.
5. For **Anonymous Employee** and **Anonymous Contact** login and password, use **SADMIN**.
6. For **Error Level for Logging**, enter **All Errors and Warnings**. You can change this once correct system operation has been verified.
7. In the **bin** directory where you have installed the Siebel Web Server Extension, open the **eapps.cfg** file and note the following (typical values are shown):

   ```
   [defaults]
   AnonUserName = SADMIN
   AnonPassword = SADMIN
   AnonUserPool = 10
   StatsPage = _stats.swe

   [/eai]
   ConnetString = siebel.TCPIP.none.none://MyGatewayServer:3230/
        MyEnterpriseServer/eaiObjMgr/MyAppServer
   EnableExtServiceOnly = TRUE
   ```
8. In the **bin** directory where you have installed the Siebel Server, open the corresponding application configuration file (for example, **eai.cfg**).
9. In the [Server] section, comment out the following line:

   ```
   ;SecurityAdapter = LDAP
   ```

   and set

   ```
   ContactLogin = FALSE
   ```
10. In the [SWE] section, comment out the following lines:

    ```
    ;UserSWFName =
    ;ContactLogin = TRUE
    ```

11 If LDAP is not used, comment out all of the following lines:

```
;[SecurityAdapters]
;LDAP = LDAP

;[LDAP]
;DllName = sscfldap.dll
;ServerName =
;Port = 389
;BaseDN =
;UsernameAttributeType = uid
;PasswordAttributeType = userPassword
;CredentialsAttributeType = credentials
;RolesAttributeType = roles
;SslDatabase =
```

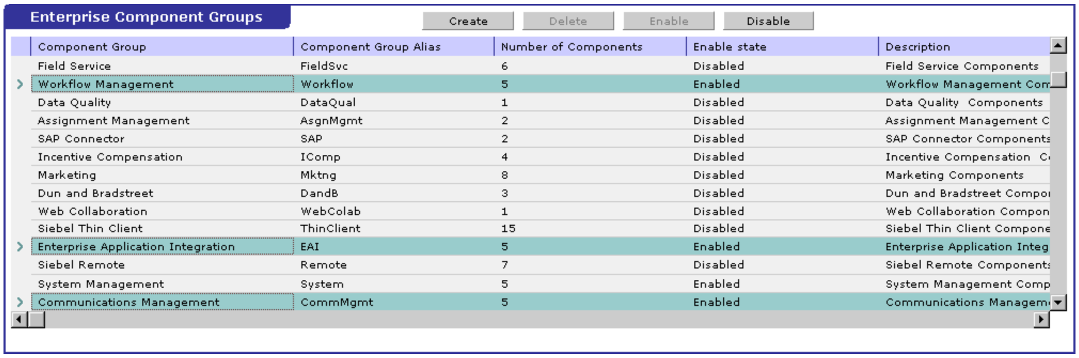
12 After modifying these files, stop and then restart the following services:

- Siebel Server
- Web Publishing Service

13 Log in to Siebel Sales 6.0 and follow the **Screens** menu path:

Server Administration > Enterprise Config > Enterprise Component Groups

Figure 8 Enterprise Component Groups



Component Group	Component Group Alias	Number of Components	Enable state	Description
Field Service	FieldSvc	6	Disabled	Field Service Components
Workflow Management	Workflow	5	Enabled	Workflow Management Com
Data Quality	DataQual	1	Disabled	Data Quality Components
Assignment Management	AsgnMgmt	2	Disabled	Assignment Management C
SAP Connector	SAP	2	Disabled	SAP Connector Components
Incentive Compensation	IComp	4	Disabled	Incentive Compensation C
Marketing	Mktng	8	Disabled	Marketing Components
Dun and Bradstreet	DandB	3	Disabled	Dun and Bradstreet Compo
Web Collaboration	WebColab	1	Disabled	Web Collaboration Compon
Siebel Thin Client	ThinClient	15	Disabled	Siebel Thin Client Compone
Enterprise Application Integration	EAI	5	Enabled	Enterprise Application Integ
Siebel Remote	Remote	7	Disabled	Siebel Remote Components
System Management	System	5	Enabled	System Management Comp
Communications Management	CommMgmt	5	Enabled	Communications Managem

14 Select the following items, and click **Enable**:

- Enterprise Application Integration
- Workflow Management
- Communication Management

15 Open the browser and type:

```
<yourservername>/<module> (for example,10.1.3.135/eai)
```

and then click **Enter**.

Make sure that your Siebel Server ODBC data source is configured. You can verify which one you are using by examining the Siebel Server log directory—it contains a file listing all the parameters.

2.3.2 Verifying SWSE Operation

To verify proper operation of the Siebel Web Server Extension (SWSE)

1. Verify that the **Server Request Processor** is running without error. You may need to synchronize the Server Request Components with the Gateway Server—follow the menu/command path:

   Enterprise Configuration > **Batch Components Admin** > **Synchronize**

2. Verify that the **EAI Object Manager** is running without error. Follow the path:

   Server Admin > **Servers** > **Server Components**

   Under **Assignment Components**, select **EAI Object Manager**.

3. Verify that the **.swe** file is associated with **sweiis.dll** in the web site. Use the following procedure to set the association:

   C Run the IIS 4.0 **Management Console** application.

   D Right-click on your Web site entry in the tree display, and select **Properties**.

   E Select the **Home Directory** tab.

   F In the **Application Settings** box, select **Configuration**.

   G On the **App Mappings** tab, select **Add**.

   H Type the following line:

   ```
   .make association swe - sweiis.dll
   ```

   I Select **All Siebel apps**.

4. Verify that the configuration files are set up properly. If LDAP is not used, comment out all LDAP-related parameters in the configuration files of the corresponding application (see step 11 in the previous section).

2.3.3 Setting Up the MUX ASP

For **Siebel-to-e*Gate** operation, additional setup is required to enable operation of the MUX ASP. Note that the existing Active Server Page file, **Mux.asp**, serves as a template that you must modify to suit your system.

To Register the ActiveX Client

- If your e*Gate installation is co-located with your Siebel 2000 application, launch a command shell and, at the command prompt, type:

  ```
  regsvr32 <drive>:\egate\client\bin\stc_xipmpclnt.dll <enter>
  regsvr32 <drive>:\egate\client\bin\stc_common.dll <enter>
  regsvr32 <drive>:\egate\client\bin\stc_ewipmpclnt.dll <enter>
  ```

  A message box appears after each entry, confirming that the command was performed successfully; no messages are displayed in the command window itself.

- If your e*Gate installation is located on a different platform from your Siebel 2000 application:

  A Copy the ***.dll** files from the **\eGate\Client\bin** directory to a directory on your Siebel host.

  B Launch a command shell and, at the command prompt, type:

  ```
  regsvr32 <drive>:\<path>\stc_xipmpclnt.dll <enter>
  regsvr32 <drive>:\<path>\stc_common.dll <enter>
  regsvr32 <drive>:\<path>\stc_ewipmpclnt.dll <enter>
  ```

  where <path> is the path to your chosen directory location.

  A message box appears after each entry, confirming that the command was performed successfully; no messages are displayed in the command window itself.

To Install the Active Server Page File

1. Locate the file **Mux.asp**, which is contained in the following sample schema file on the installation CD-ROM:

   ```
   <cdrom>\setup\samples\ewsiebelhttp\ewsiebelhttpsample.zip
   ```

2. Copy the file to the directory of the Web Server that you want to access.
3. Modify the following line:

   ```
   const strHost = "localhost"
   ```

   using the IP address or host name corresponding to your MUX server. If your MUX server uses a port number other than the default, also change the value of **dwPort**.

2.4 Optional Example Files

The installation CD-ROM contains two sample schemas, **SiebelHTTPSample**, and **SiebelHTTPSample2**, located in the **samples\ewsiebelhttp** directory. Note that sample schema for the Java version of this e*Way are located in the same directory. These are described in the User's Guide for the Java version.

- **ewsiebelhttpsample.zip** ((e*Gate-to-Siebel schema example)
- **ewsiebelhttpsample2.zip** (Siebel-to-e*Gate schema example)

Table 6 e*Gate CD-ROM Directory Structure

Subdirectory	Files	Description
\Siebel2000\	ewsiebelhttpsample.zip	Monk example.
	ewsiebelhttpsample2.zip	Monk example.
	JavaSiebelOutbound.zip	Java example for Siebel 2000—ignore.
	SiebelAccount.xml	
\Siebel2000\inputdata\ewsiebelhttpsample\	input.fin	Input data for Monk example.
\Siebel2000\inputdata\JavaSiebelOutbound\	sample.xml	Input data for Java example—ignore.
\Siebel7\	JavaSiebelInbound.zip	Java Inbound example for Siebel 7—ignore.
	JavaSiebelOutbound.zip	Java Outbound example for Siebel 7—ignore.
\Siebel7\inputdata\JavaSiebelOutbound\	sample.xml	Input data for Java Outbound example—ignore.

To use a schema, you must load it onto your system using the following procedure. See **Sample Schema** on page 110 for information on the schemas.

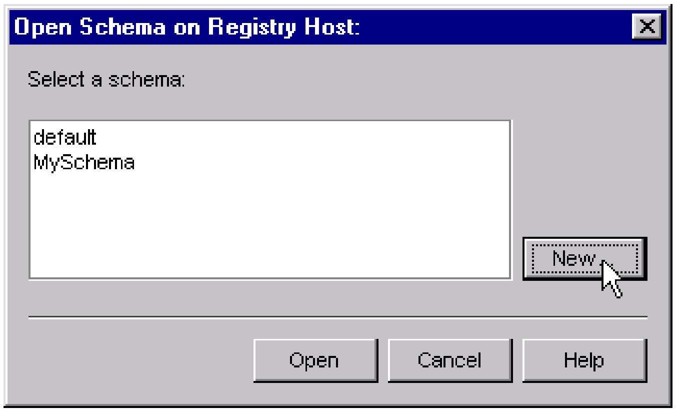
Note: *Each schema must be imported individually, and the appropriate Siebel EAI e*Way (HTTP or MUX) must have been installed and configured.*

2.4.1 Installation Procedure

To load a sample schema

1 Invoke the **Open Schema** dialog box and select **New** (see Figure 9).

Figure 9 Open Schema Dialog



Open Schema on Registry Host:

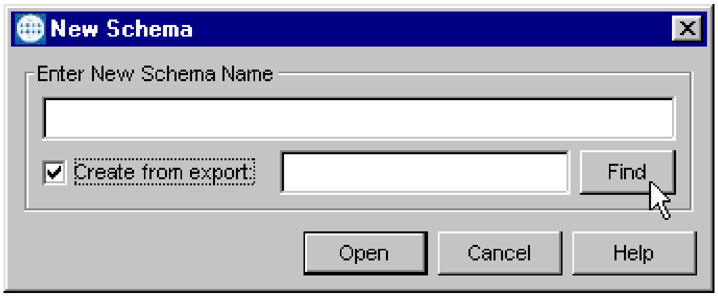
Select a schema:

default
MySchema

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

3 Select **Create from export** and navigate to the directory containing the sample schema by clicking the **Find** button (see Figure 10).

Figure 10 New Schema Dialog



New Schema

Enter New Schema Name

☑ Create from export:

4 Navigate to 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.*

2.4.2 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 7 Subdirectories and Files - SiebelHTTPSample

Subdirectories	Files
\	SiebelHTTPSample.ctl
\runtime\configs\stcewfile\	siebelhttpsample_eater.cfg siebelhttpsample_eater.sc siebelhttpsample_feeder.cfg siebelhttpsample_feeder.sc
\runtime\configs\stcewgenericmonk\	siebelhttpsample_http.cfg siebelhttpsample_http.sc
\runtime\monk_scripts\common\	intObj.ssc qput.isc sample.ssc SetIntObj.tsc siebel-http-outgoing-delete.dsc siebel-http-outgoing-execute.dsc siebel-http-outgoing-insert.dsc xmlinput.ssc

Table 8 Subdirectories and Files - SiebelHTTPSample2

Subdirectories	Files
\	SiebelHTTPSample2.ctl
\runtime\configs\stcewfile\	SiebelHTTPSample2_file.cfg SiebelHTTPSample2_file.sc
\runtime\configs\stcewipmp\	SiebelHTTPSample2_mux.cfg SiebelHTTPSample2_mux.sc
\runtime\monk_scripts\common\	stripMuxHeader.tsc

Chapter 3

Configuration Parameters

This chapter describes the configuration parameters for the Siebel EAI e*Way.

3.1 Overview

The e*Way's configuration parameters are set using the e*Way Editor; see **Configuring the e*Way** on page 59 for procedural information. The Siebel EAI e*Way's configuration parameters are divided into two groups, depending upon the direction of data flow.

e*Gate-to-Siebel Mode (HTTP)

This e*Way's configuration parameters are organized into the following sections. The default configuration is provided in **stcewsiebelhttp.def**.

General Settings (HTTP) on page 32

Communication Setup on page 34

Monk Configuration on page 37

HTTP Configuration on page 46

HTTP Proxy Configuration on page 49

HTTP SSL Configuration on page 51

Siebel Configuration on page 53

Siebel-to-e*Gate Mode (MUX)

This e*Way's configuration parameters are found in the following section. The default configuration is provided in **stcewsiebelipmp.def**.

General Settings (MUX) on page 54

3.2 General Settings (HTTP)

These General Settings control top level operational parameters for e*Gate-to-Siebel operation.

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.

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 between **1** and **1,024**. The default is **5**.

Max Failed Messages

Description

Specifies the maximum number of failed messages 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 between **1** and **1,024**. The default is **3**.

Forward External Errors

Description

Selects whether or not error messages that begin with the string **“DATAERR”** that are received from the external system is queued to the e*Way’s configured queue. See **Exchange Data with External Function** on page 40 for more information.

Required Values

Yes or **No**. The default value, **No**, specifies that error messages are not to be forwarded. See **Data Exchange Process** on page 75 for more information about how the e*Way uses this function.

3.3 Communication Setup

The Communication Setup parameters control the schedule by which the e*Way obtains data from the external system.

Note: *The schedule that 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 is exchanged. Be sure that 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 **Exchange Data Interval** setting is ignored and the e*Way invokes the **Exchange Data with External Function** immediately.

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

Zero Wait Between Successful Exchanges

Description

Selects whether or not 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**.

If this parameter is set to **Yes**, the e*Way immediately invokes the **Exchange Data with External Function** if the previous exchange function returned an data.

If this parameter is set to **No**, the e*Way always waits the number of seconds specified by **Exchange Data Interval** between invocations of the **Exchange Data with External Function**.

Start Exchange Data Schedule

Description

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

Required Values

One of the following:

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

Also required: If you set a schedule using this parameter, you must also define all three of the following:

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

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

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** and **Negative Acknowledgment Function**) and whether or not 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.

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

See also **Zero Wait Between Successful Exchanges** on page 34 for more information.

Stop Exchange Data Schedule

Description

Establishes the schedule to stop data exchange.

Required Values

One of the following:

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

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

Down Timeout

Description

Specifies the number of seconds that the e*Way waits between calls to the **External Connection Establishment Function**.

Required Values

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

Up Timeout

Description

Specifies the number of seconds the e*Way waits between calls to the **External Connection Verification Function**.

Required Values

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

Resend Timeout

Description

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

Required Values

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

3.4 Monk Configuration

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

Specifying Function or File Names

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

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

- .monk
- .tsc
- .dsc

Specifying Multiple Directories

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

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

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

Load Path

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

Additional Path

Description

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

Required Values

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

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

Additional information

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

Auxiliary Library Directories

Description

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

Required Values

A pathname, or a series of paths separated by semicolons. The default value is **monk_library/ewsiebelhttp**.

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

Monk Environment Initialization File

Description

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

Required Values

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

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

Returns

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

Additional information

- Use this feature to initialize the e*Way's Monk environment (for example, to define Monk variables that are used by the e*Way's function scripts); it is good practice to initialize any global Monk variables that may be used by any other Monk Extension scripts
- The internal function that loads this file is called once when the e*Way first starts up
- The e*Way loads this file and try to invoke a function of the same base name as the file name

Startup Function

Description

Specifies a Monk function that the e*Way loads and invokes upon startup or whenever the e*Way's configuration is reloaded. It is called after the e*Way loads the specified **Monk Environment Initialization File** and any files within the specified **Auxiliary Library Directories**. This function accepts no input, and must return a string.

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

Required Values

The name of a Monk function or the name of a file containing a Monk function. The default is **siebel-http-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

Three template scripts are supplied with the e*Way:

- monk_scripts/common/siebel-http-outgoing-delete.dsc
- monk_scripts/common/siebel-http-outgoing-execute.dsc
- monk_scripts/common/siebel-http-outgoing-insert.dsc

The default is **monk_scripts/common/siebel-http-outgoing-insert.dsc.**

These scripts should be modified and renamed for use in your production system. To return data to the e*Gate system, the script should call **event-send-to-egate**.

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

Returns

- A *null string* ("") indicates that the Event was published successfully to the external system
- A string beginning with **RESEND** indicates that the Event should be resent

- A string beginning with **CONNERR** indicates that there is a problem with the connection to the external system, and causes a rollback of the Event
- A string beginning with **DATAERR** indicates that there is a problem with the message (Event) data itself, and causes a rollback of the Event
- A string beginning with **SHUTDOWN** indicates that the e*Way must exit immediately
- If any string other than one of the preceding is returned, the e*Way creates an entry in the log file indicating that an attempt has been made to access an unsupported function

Additional Information

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

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

Exchange Data with External Function

Description

Specifies a Monk function that initiates the transmission of data from the external system to the e*Gate system and forwards that data as an inbound Event to one or more e*Gate Collaborations. This function is invoked automatically by the **Start Exchange Data Schedule** or manually by the **start-schedule** Monk function, and is responsible for either sending data to or receiving data from the external system. If this function returns data, it is queued to e*Gate in an inbound Collaboration. The e*Way must have at least one Collaboration configured suitably to process the inbound Event, as well as any required IQs.

Required Values

The name of a Monk function or the name of a file containing a Monk function. The default value is **siebel-http-exchange**.

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

Returns

- A *null string* (““) indicates that the data exchange was completed successfully, but with no resultant data sent back to the e 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 **siebel-http-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. The default is **siebel-http-verify**.)

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

Returns

- **"SUCCESS"** or **"UP"** indicates that the connection was established successfully
- Any other string (including the null string) indicates that the attempt to establish the connection failed

Additional Information

If this function is not specified, the e*Way executes the **External Connection Establishment Function** in its place. This latter function also is called when the e*Way has determined that its connection to the external system is down.

External Connection Shutdown Function

Description

Specifies a Monk function that the e*Way calls to shut down the connection to the external system. This function is invoked only when the e*Way receives a *suspend* command from a Control Broker.

Required Values

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

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

Input

A string indicating the purpose for shutting down the connection.

- **"SUSPEND_NOTIFICATION"** - the e*Way is being suspended or shut down
- **"RELOAD_NOTIFICATION"** - the e*Way is being reconfigured

Returns

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

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

Positive Acknowledgment Function

Description

This function is loaded during the initialization process and is called when all data received from the external system has been processed and enqueued successfully.

Required Values

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

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

Required Input

A string, the inbound Event to e*Gate.

Returns

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

Additional Information

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

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

Negative Acknowledgment Function

Description

This function is loaded during the initialization process and is called when the e*Way fails to process or enqueue data received from the external system successfully.

Required Values

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

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

Required Input

A string, the inbound Event to e*Gate.

Returns

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

Additional Information

- This function is called only during the processing of inbound Events. After the **Exchange Data with External Function** returns a string that is transformed into an inbound Event, the Event is handed off to one or more Collaborations for further processing. The e*Way executes this function if the Event's processing is not completed successfully by *all* the Collaborations to which it was sent; otherwise, the e*Way executes the **Positive Acknowledgment Function**.
- This function can return data to be queued, but the e*Way will *not* acknowledge the data with an **ACK** or **NAK**.

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

Shutdown Command Notification Function

Description

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

Required Values

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

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

Input

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

Returns

- A *null string* or **"SUCCESS"** indicates that the shutdown can occur immediately
- Any other string indicates that shutdown must be postponed; once postponed, shutdown does not proceed until the Monk function **shutdown-request** is executed

Additional Information

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

3.5 HTTP Configuration

The parameters in this section furnish the required HTTP variables.

Request

Description

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

Required Values

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

Timeout

Description

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

Required Values

A number between **1** and **864,000**; the default value is **50,000**.

URL

Description

The URL to **GET** or **POST**.

Required Values

A full URL, for example:

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

Note: *The default URL is set to* **http://<Web Server>/eai/start.swe**. *You must replace* **<Web Server>** *with a valid web server name or IP address.*

Additional Information

If using **GET**, you can provide parameters using the **application/x-www-form-urlencoded** notation, for example:

```
http://www.<sitename>.com/search?p1=<x>&p2=<y>&p3=<z>
```

where **<sitename>** is replaced with a valid site name, **p1=<x>**, **p2=<y>** and **p3=<z>** are replaced with valid search information in the given format, such as **p1=john**, **p2=james**, and **p3=doe**.

User Name

Description

The username for authentication when connecting to an HTTP server.

Required Values

A string; the default is **Anonymous**.

Encrypted Password

Description

The password for authentication when connecting to the HTTP server.

Required Values

A valid password string.

Agent

Description

The agent name to pass to the HTTP server. This is usually used to specify the type of client used, for example, browser.

Required Values

String; the default is **e*Gate HTTP e*Way**.

Content-type

Description

Specifies the content type.

Required Values

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

Additional Information

Normally, this should not be changed from the default value.

Request-content

Description

Sets up request content if using the **POST** method (this parameter is ignored for **GET**).

Required Values

String (see below).

Additional Information

The content normally is in the **application/x-www-form-urlencoded** format of name/value pairs, for example:

```
p1=seebeyond&p2=snoopy
```

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

Accept-type

Description

Provides the parameters for the Accept Type request header.

Required Values

A string, for example:

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

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

3.6 HTTP Proxy Configuration

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

Use Proxy Server

Description

Specifies whether or not to use a Proxy server.

Required Values

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

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

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

User Name

Description

The user name for authentication when connecting to a Proxy server.

Required Values

A valid username string.

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

Encrypted Password

Description

The password for authentication when connecting to a Proxy server.

Required Values

A valid password string.

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

Server Address

Description

Specifies the proxy server to be used.

Required Values

A proxy server URL, for example:

```
http://myproxy
```

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

Port Number

Description

The port number the Proxy server is listening on.

Required Values

A number between **1** and **864,000**; the default value is **8080**.

3.7 HTTP SSL Configuration

The parameters in this section furnish the required variables for an SSL connection via HTTP.

Trusted CA Certificated Directory

Description

Specifies the directory located on the Repository where all of the CA certificates are located. These certificates are used to verify a trust relationship between you and the CA.

Required Values

A directory string; the default is **pkicerts/trustedcas**.

Use Client Certificate Map

Description

Enables or disables the selection of client certificates based on a certificate map.

Required Values

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

Additional Information

A certificate map is a text file that maps a base URL to a client certificate file and client private key file.

Client Certificate Map File

Description

Specifies the directory and file name of the text file containing the client certificate map.

Required Values

A directory string; the default is **pkicerts/client/certmap.txt**.

Additional Information

The file contains four columns, separated by a '|' bar sign, representing the following information:

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

Example:

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

If there is an asterisk (*) in the first column replacing the base URL, it means *all others*.

If there is a pound sign (#) in the first column, the line is treated as a comment.

Lines in the file are processed from top to bottom and the first base URL match found is the one used.

3.8 Siebel Configuration

The parameters in this section furnish the required variables for executing service and logging onto the Siebel Server.

SWEExtSource

Description

Specifies the service the Siebel Web Engine is going to call. This should match one of the services that is listed under the section [HTTP Services] in the file **eai.cfg**.

Required Values

A string.

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

SWEExtCmd

Description

Specifies the command Siebel Web Engine uses to execute the service.

Required Values

A string; the default is **Execute**.

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

User Name

Description

Specified the username for authentication when connecting to the Siebel server.

Required Values

A string; the default is **SADMIN**.

Encrypted Password

Description

Specified the password for authentication when connecting to the Siebel server.

Required Values

A valid password string; the default is a null string.

3.9 General Settings (MUX)

These General Settings control top level operational parameters for Siebel-to-e*Gate operation, allowing multiple request/reply connections from an external system, such as a WWW server.

Request Reply IP Port

Description

This is the IP port number to which the e*Way listens (binds) for client connections (for Request/Reply behavior). This parameter is required.

Required Values

A number between **1** and **65,536**. The default is **26,051**.

Push IP Port

Description

If this parameter is set to a non-zero value, the e*Way allows an external system to connect and receive unsolicited events (i.e., without submitting a request).

Required Values

A number between **1** and **65,536**. The default is **0**.

Additional Information

Any event received by the e*Way that has zero values for all fields in the 24-byte MUX header is sent to all callers of the Wait.

Rollback if No Clients on Push Port

Description

If set to **Yes**, the event continually rolls back if there are no push clients connected.

Required Values

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

Wait for IQ Ack

Description

If this parameter is set to **Yes**, the send client function does **not** return until the event is committed to the IQ.

Required Values

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

Additional Information

This parameter should be set if the data must be committed to the IQ on every transaction before the API returns to the client.

If normal request/reply type transactions are being sent or received, and the data can be recreated at the client, this parameter should not be set.

Note: *Setting this parameter to* **Yes** *has a significant negative impact on performance.*

Chapter 4

e*Way Setup

This chapter describes procedures for customizing the Siebel EAI e*Way to operate with your Siebel EAI system.

4.1 Overview

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

The topics discussed in this chapter include the following:

Setting Up the e*Way

Creating the e*Way on page 57

Modifying e*Way Properties on page 58

Configuring the e*Way on page 59

Changing the User Name on page 63

Setting Startup Options or Schedules on page 63

Activating or Modifying Logging Options on page 65

Activating or Modifying Monitoring Thresholds on page 66

Troubleshooting the e*Way

Configuration Problems on page 67

System-related Problems on page 68

4.2 Setting Up the e*Way

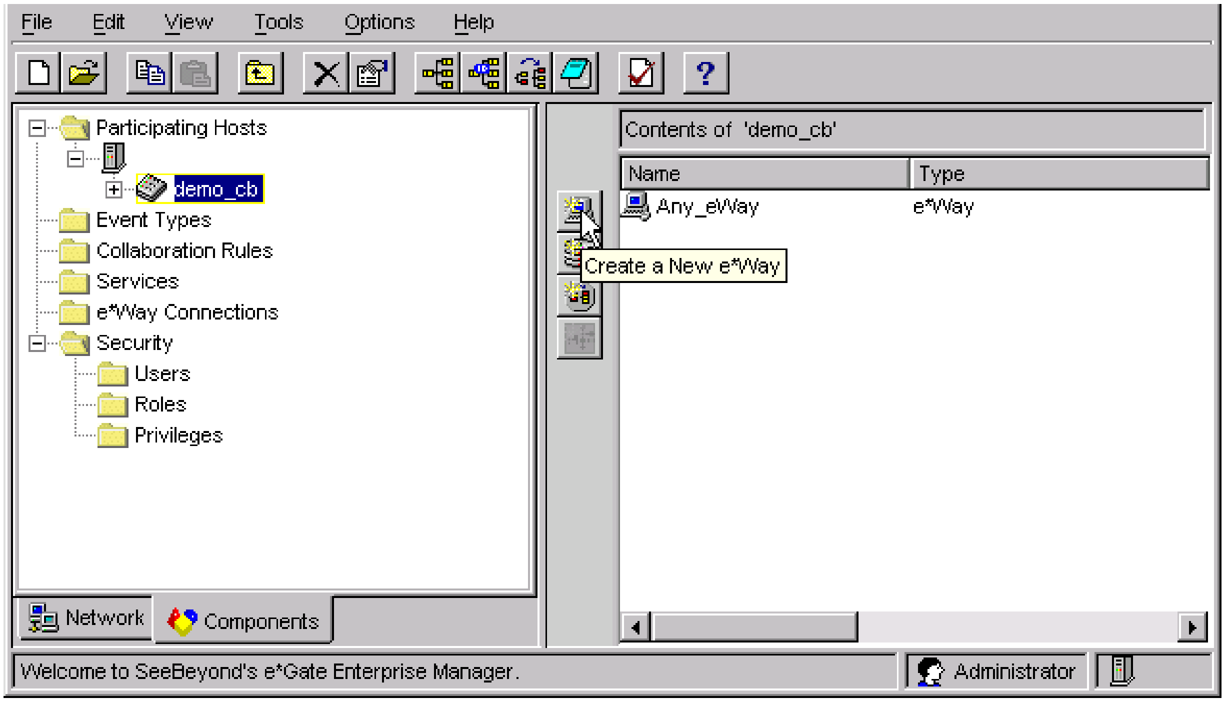
4.2.1 Creating the e*Way

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 11 e*Gate Schema Designer Window (Components View)



Contents of 'demo_cb'

Name	Type
Any_eWay	e*Way

Welcome to SeeBeyond's e*Gate Enterprise Manager.

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.

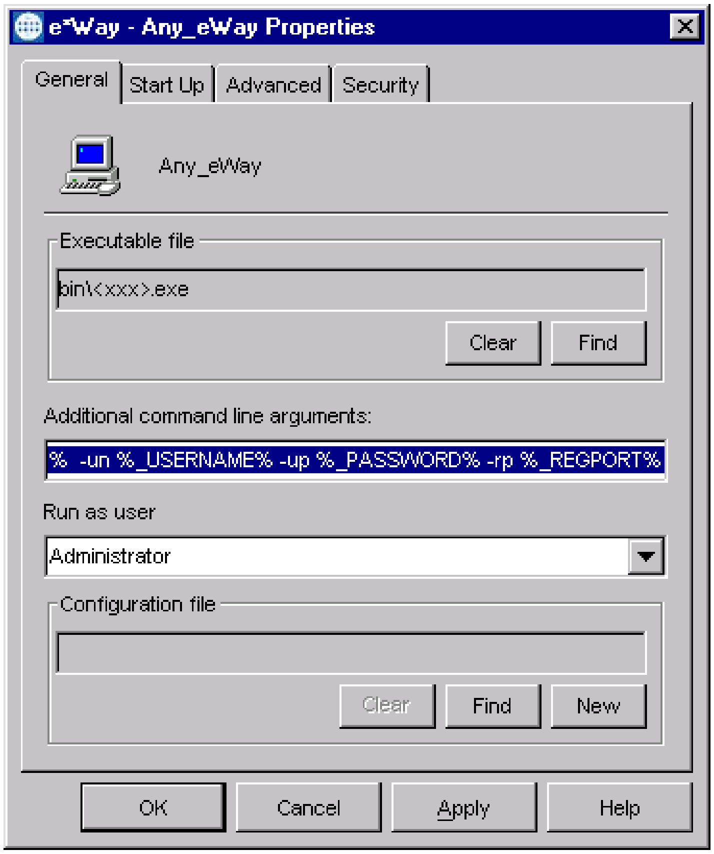
4.2.2 Modifying e*Way Properties

To modify any e*Way properties

1. Right-click on the desired e*Way and select **Properties** to edit the e*Way's properties. The properties dialog opens to the **General** tab (shown in Figure 12).

Note: *The executable and default configuration files used by this e*Way are listed in* **e*Way Components** *on page 16.*

Figure 12 e*Way Properties (General Tab)



Any_eWay

Executable file: bin\<xxx>.exe

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

Run as user: Administrator

Configuration file:

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

4.2.3 Configuring the e*Way

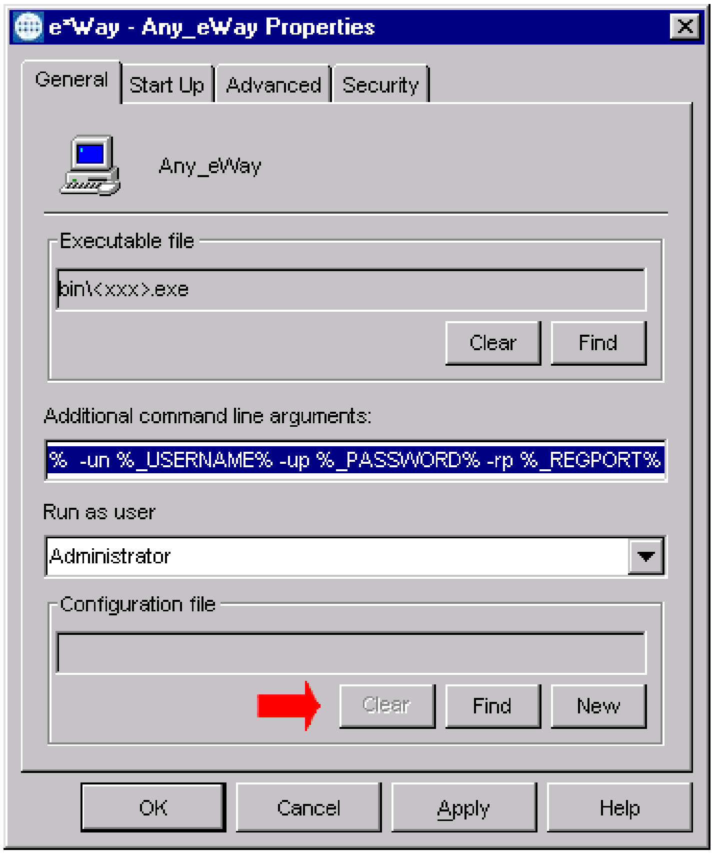
The e*Way's default configuration parameters are stored in an ASCII text file with a **.def** extension. The e*Way Editor provides a simple graphical interface for viewing and changing those parameters to create a working configuration (**.cfg**) file.

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 executable and default configuration files used by this e*Way are listed in* **e*Way Components** *on page 16.*

Figure 13 e*Way Properties - General Tab



e*Way - Any_eWay Properties

Any_eWay

Executable file: bin\<xxx>.exe

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

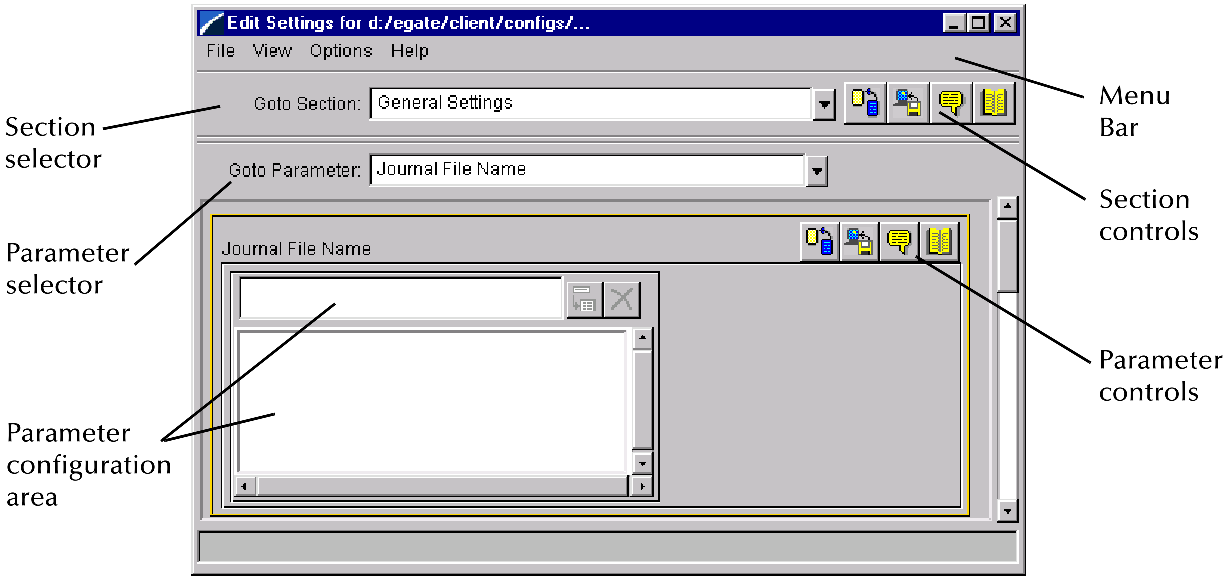
Run as user: Administrator

Configuration file:

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

Using the e*Way Editor

Figure 14 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 9 below.

Table 9 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 10

Table 10 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 Help system that launches from the Toolbar's Help menu. 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*.

4.2.4 Changing the User Name

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

To change the user name

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

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

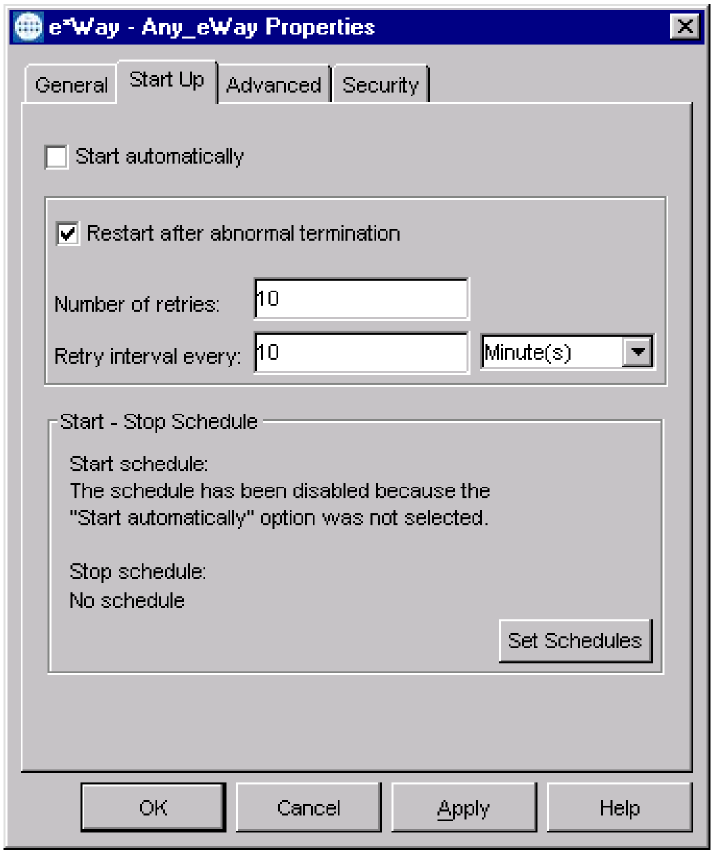
4.2.5 Setting Startup Options or Schedules

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

Figure 15 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**.

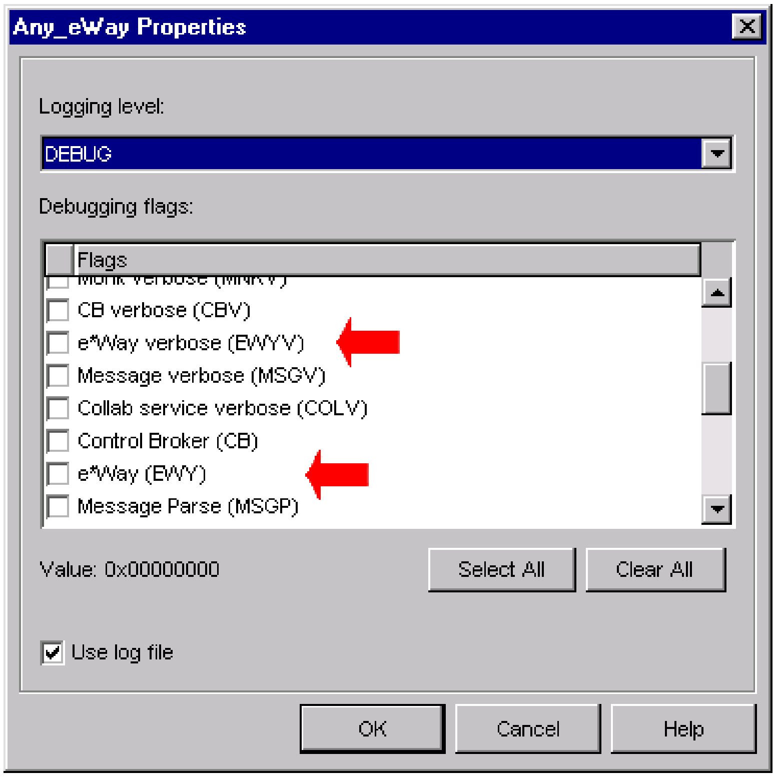
4.2.6 Activating or Modifying Logging Options

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**. The dialog window appears as in Figure 16.

Figure 16 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

☑ 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 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.

4.2.7 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 is routed 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 provide information to this e*Way properly configured, and are they sending the appropriate Events correctly?
- Are all the components to which this e*Way sends information 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 environmental variable includes a path to the Siebel EAI dynamically-loaded libraries.

In the Siebel 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 Schema Manager system to monitor operations and performance.

Chapter 5

Operational Overview

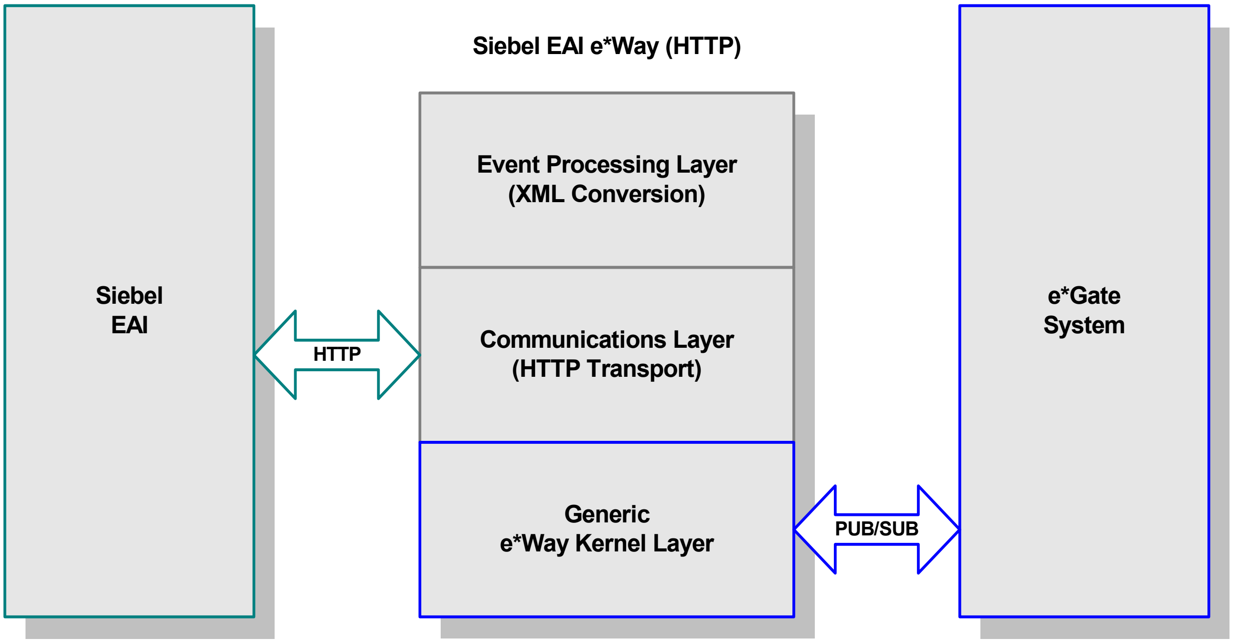
This chapter contains an overview of the architecture and basic internal processes of the Siebel EAI e*Way.

5.1 Siebel EAI e*Way Architecture

5.1.1 e*Gate-to-Siebel (HTTP) Mode

Conceptually, theSiebel EAI e*Way (in HTTP mode) can be viewed as a three-layered structure, consisting of an Event Processing layer that handles XML conversion, a Communications layer that handles HTTP transport communication with the Siebel application, and an e*Way Kernel layer that manages the processing of data and subscribing or publishing to other e*Gate components (see Figure 17).

Figure 17 Siebel EAI e*Way (HTTP) Architecture


Siebel EAI e*Way (HTTP)
Siebel EAI
HTTP
Event Processing Layer (XML Conversion)
Communications Layer (HTTP Transport)
Generic e*Way Kernel Layer
PUB/SUB
e*Gate System

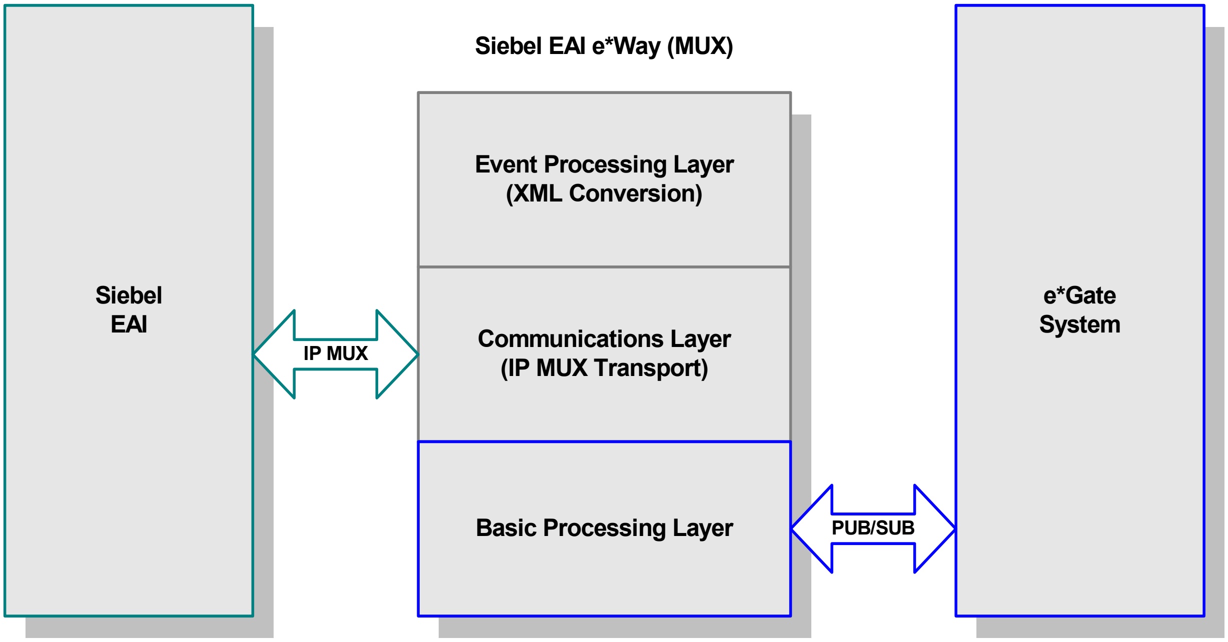
The upper layers of the e*Way use Monk functions to perform Business Process modeling and ETD mapping, package data as e*Gate *Events*, send those Events to Collaborations, and manage interaction with the external system. These layers are built upon an e*Way Kernel layer that manages the basic operations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

5.1.2 Siebel-to-e*Gate (MUX) Mode

Conceptually, the Siebel EAI e*Way operating in MUX mode can be viewed as a three-layered structure, consisting of an Event Processing layer that handles XML conversion, a Communications layer that handles IP MUX transport communication with the Siebel application, and an e*Way Kernel layer that manages the processing of data and subscribing or publishing to other e*Gate components (see Figure 18).

Figure 18 Siebel EAI e*Way (MUX) Architecture



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

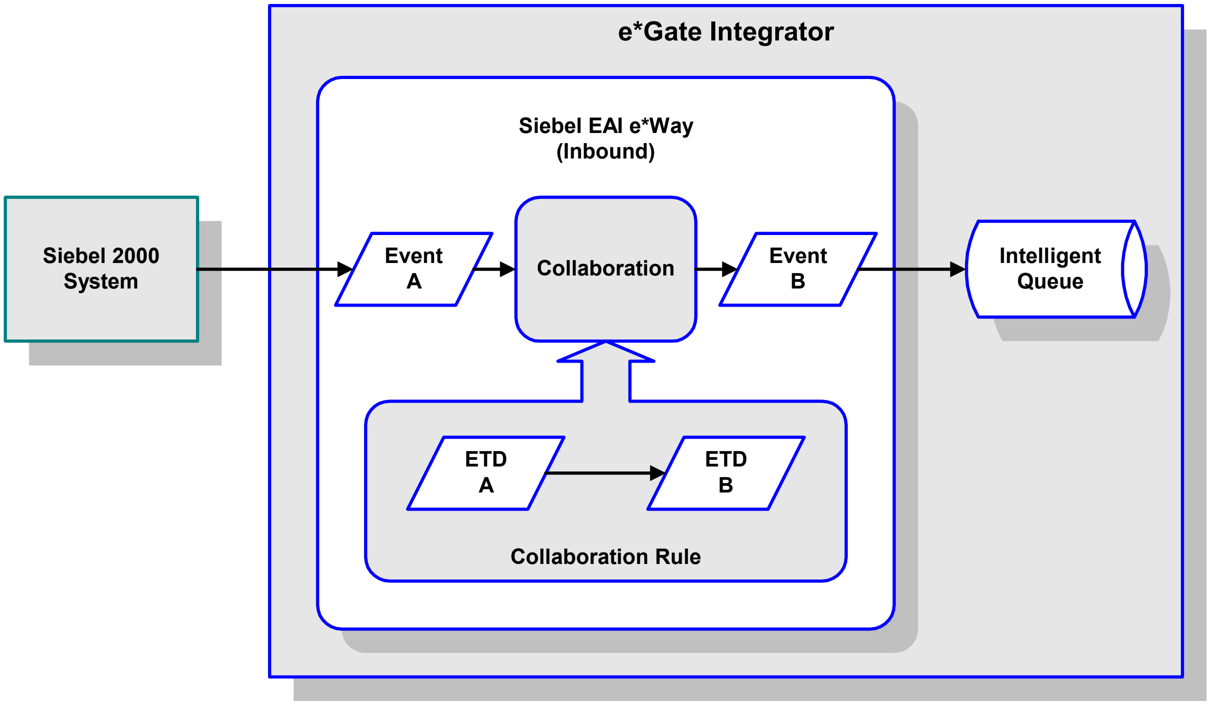
The communication layers of the e*Way are single-threaded. Functions run serially, and only one function can be executed at a time. Processing layers are multi-threaded, with one executable thread for each Collaboration. Each thread maintains its own

Monk environment; therefore, information such as variables, functions, path information, and so on cannot be shared between threads.

5.2 Collaborations

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

Figure 19 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 3**. You can create and modify these functions using the SeeBeyond Collaboration Rules Editor or a text editor (such as *Microsoft Word* or *Notepad*). 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.

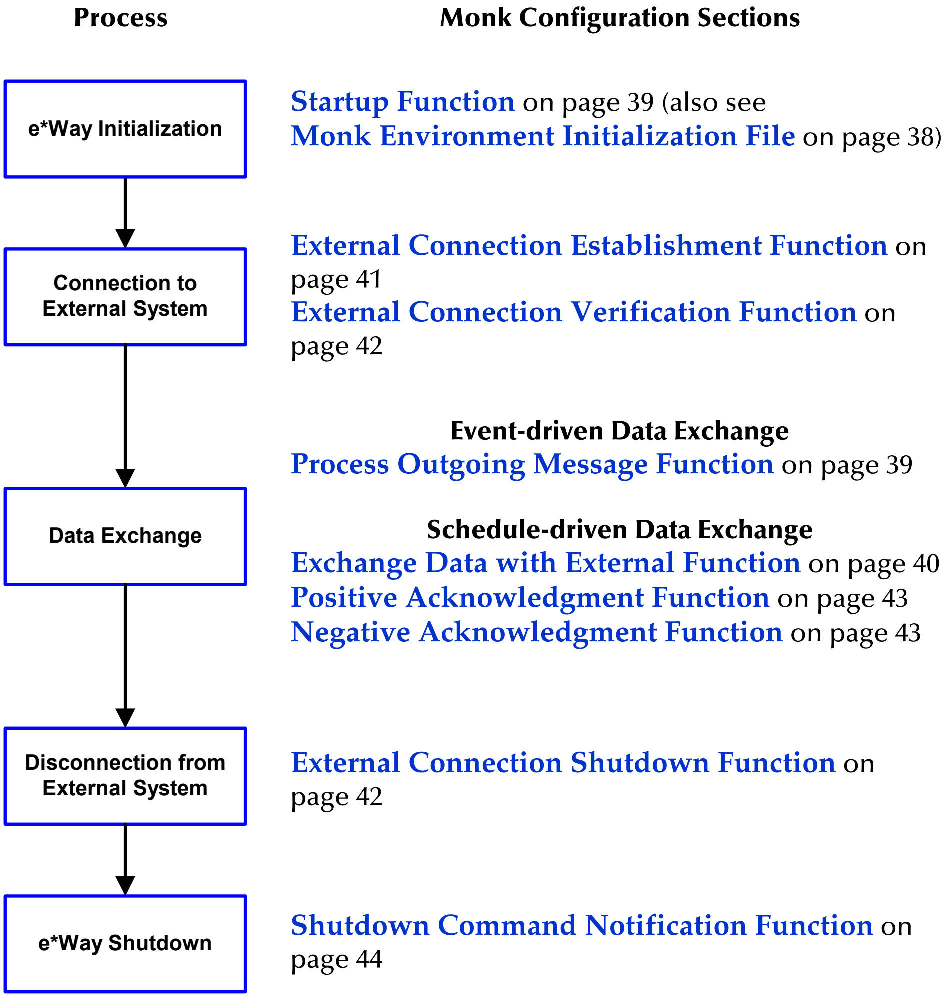
For more information on defining Collaborations, defining IQs, assigning Collaborations to e*Ways, or configuring Collaborations to publish Events, see the *e*Gate Integrator User's Guide*.

5.3 Basic e*Way Processes (HTTP Mode Only)

Note: *This section describes the basic operation of a typical e*Way based on the Generic e*Way Kernel, and applies only to the Siebel EAI e*Way in HTTP mode. Not all functionality described in this section is used routinely by the Siebel EAI e*Way.*

The most basic processes carried out by an e*Way are listed in Figure 20. In e*Ways based on the Generic Monk e*Way Kernel (using **stcewgenericmonk.exe**), these processes are controlled by the listed Monk functions. Configuration of these functions is described in the referenced sections of this User's Guide.

Figure 20 Basic e*Way Processes



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

Monk Configuration Sections

Startup Function on page 39 (also see **Monk Environment Initialization File** on page 38)

External Connection Establishment Function on page 41
External Connection Verification Function on page 42

Event-driven Data Exchange
Process Outgoing Message Function on page 39

Schedule-driven Data Exchange
Exchange Data with External Function on page 40
Positive Acknowledgment Function on page 43
Negative Acknowledgment Function on page 43

External Connection Shutdown Function on page 42

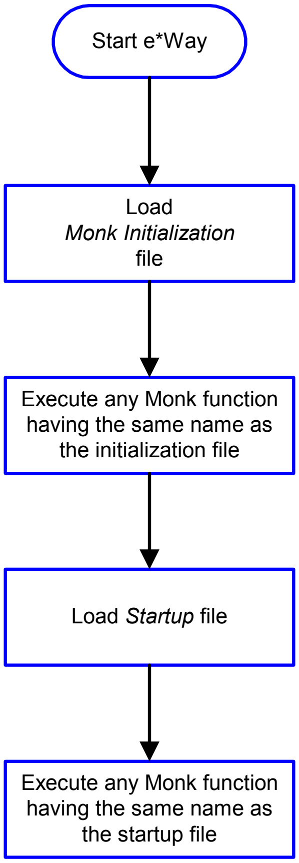
Shutdown Command Notification Function on page 44

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

Initialization Process

Figure 21 illustrates the e*Way's initialization process, using the **Monk Environment Initialization File** and **Startup Function**.

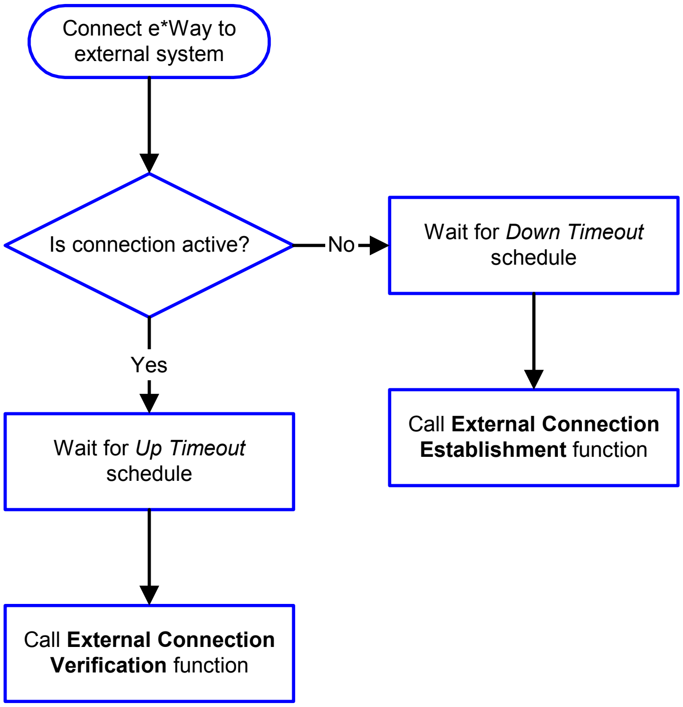
Figure 21 Initialization Process


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

Figure 22 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 24 on page 76** *and* **Figure 23 on page 75** *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 131 and* **send-external-down** *on page 131 for more information.*

Data Exchange Process

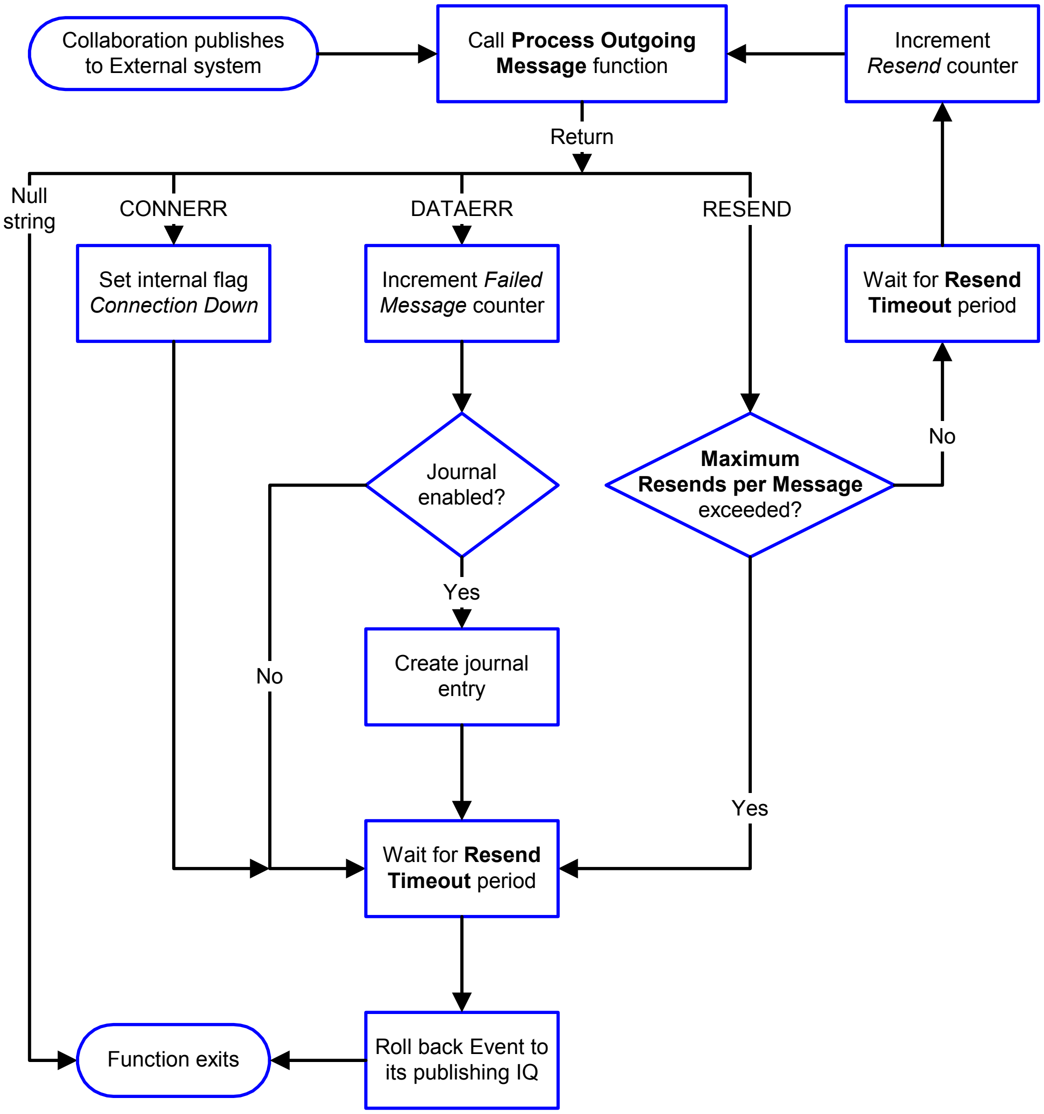
Event-driven

Figure 23 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.

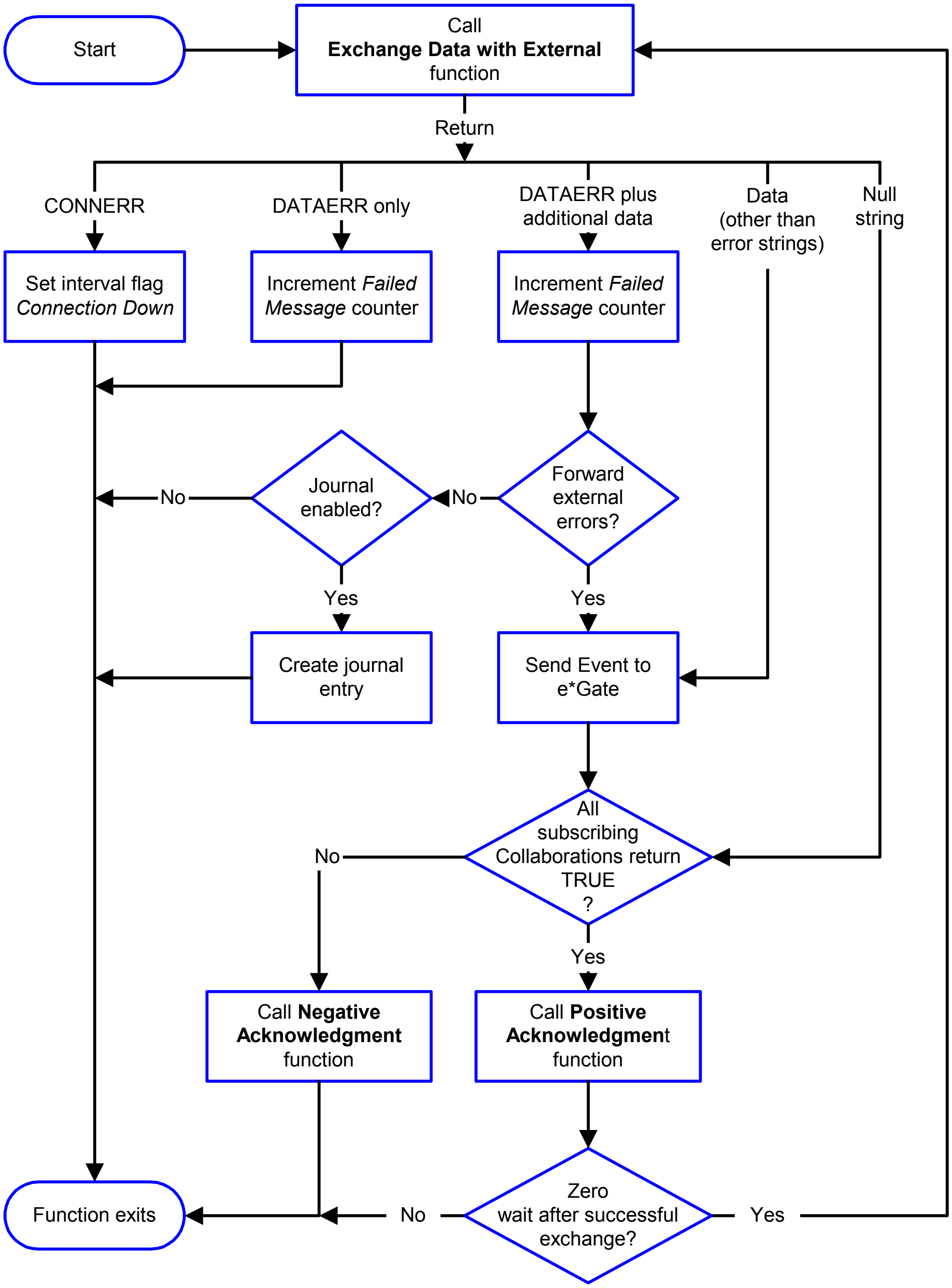
Figure 23 Event-Driven Data Exchange Process


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 24 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 24 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

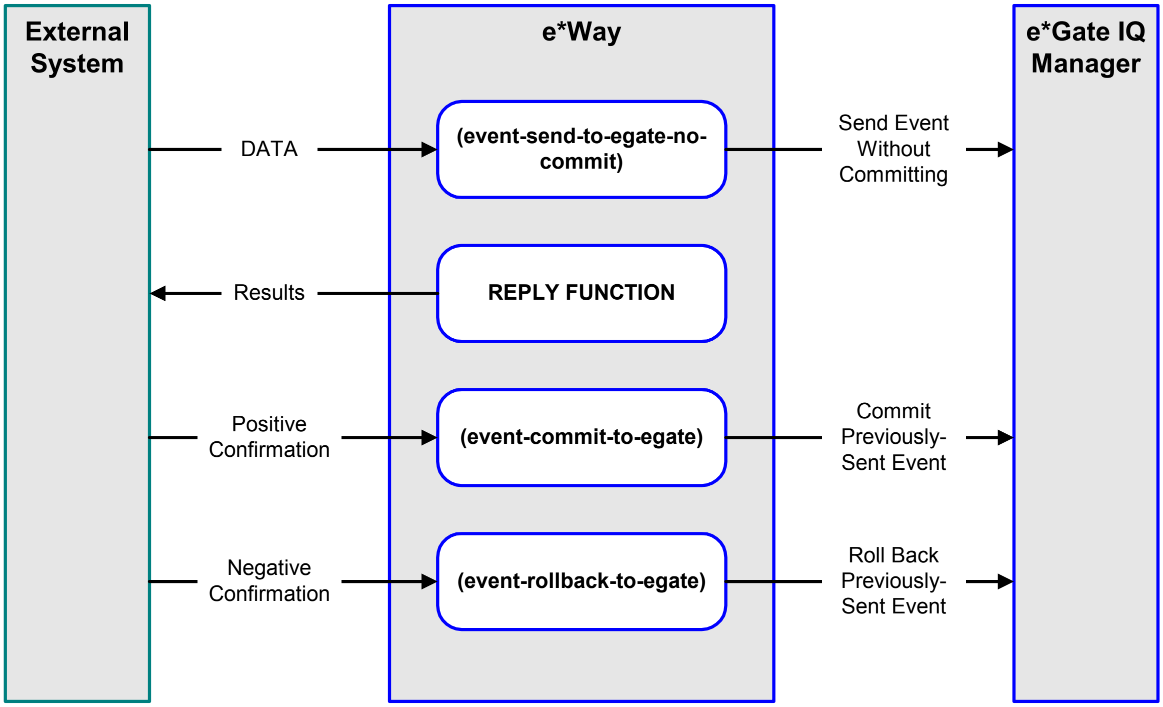
*Send Events to e*Gate* can be implemented using any of the following Monk functions:

- **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 25):

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

Figure 25 Send Event to e*Gate with Confirmation

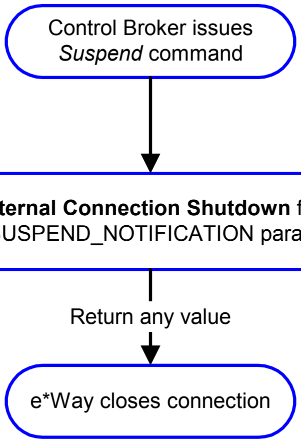

External System
e*Way
e*Gate IQ Manager
DATA
(event-send-to-egate-no-commit)
Send Event Without Committing
Results
REPLY FUNCTION
Positive Confirmation
(event-commit-to-egate)
Commit Previously-Sent Event
Negative Confirmation
(event-rollback-to-egate)
Roll Back Previously-Sent Event

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

Disconnect from External Process

Figure 26 illustrates how the e*Way disconnects from the external system, using the **External Connection Shutdown Function**.

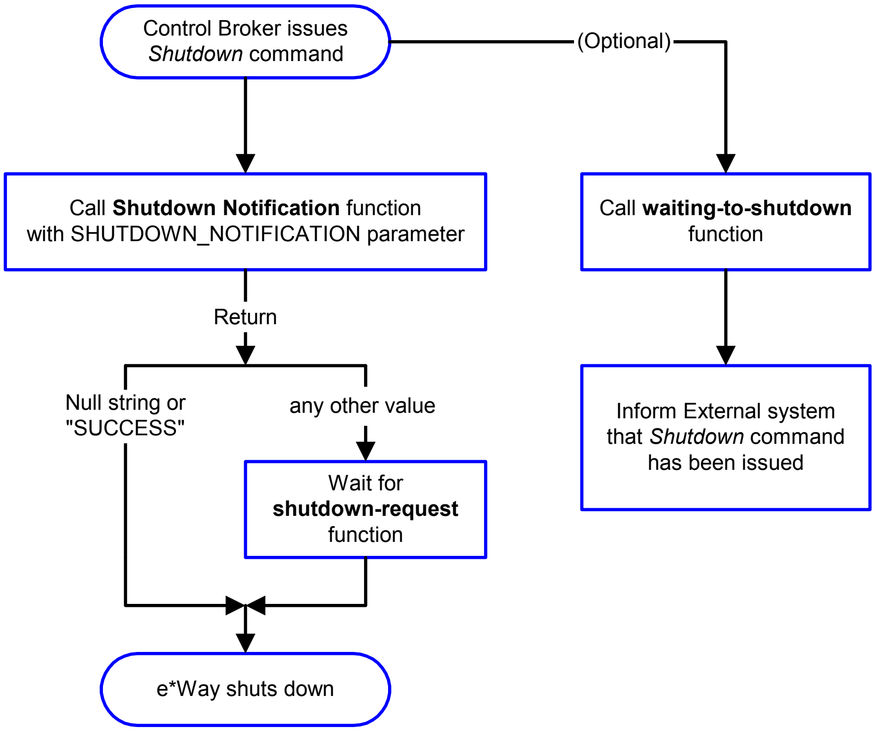
Figure 26 Disconnect Process


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

Shutdown Process

Figure 27 illustrates how the e*Way shuts itself down, using the **Shutdown Command Notification Function**.

Figure 27 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

System Implementation

This chapter describes the procedures for creating a functional Siebel-e*Gate system incorporating the Siebel EAI e*Way. Please refer to the *e*Gate Integrator User's Guide* for additional information regarding the e*Gate Integrator system.

6.1 Contents

This e*Way provides a specialized transport component for incorporation in 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.

One or more sample schema, included in the software package, are described at the end of this chapter. These can be used to test your system following installation and, if appropriate, as a template that you can modify to produce your own schema.

6.1.1 Pre-Implementation Tasks

Install the SeeBeyond Software

The first task is to install the SeeBeyond software as described in **Installing the e*Way** on page 20.

Import the Sample Schema

If you want to use the sample schema supplied with the e*Way, the schema files must be imported from the installation CD-ROM (see **Optional Example Files** on page 28).

Note: *It is highly recommended that you make use of the sample schemas to familiarize yourself with e*Way operation, test your system, and use as templates for your working schemas.*

Configure the Siebel EAI System

Follow the procedure described in **Web Server Setup** on page 24.

Set up the Siebel EAI e*Way

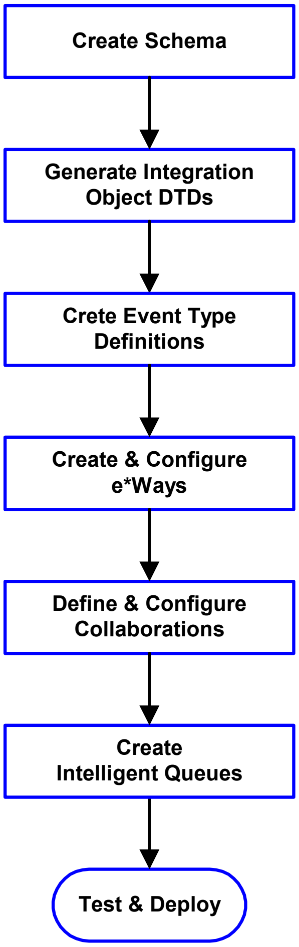
Follow the procedures described in **e*Way Setup** on page 56.

6.2 Overview

6.2.1 General Sequence

The high-level implementation sequence for a system incorporating the Siebel EAI e*Way is depicted below.

General Implementation Sequence


Create Schema
Generate Integration Object DTDs
Crete 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 97).

2 The second step is to generate and verify the Integration Object DTD in Siebel Tools (see **Generating the Integration Object DTD** on page 98).

3 Next you need to create Event Type Definitions (ETDs) derived from the Integration Object DTDs (see **Creating Event Type Definitions** on page 101).

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

5 Next you need to define and configure the Collaborations between Event Types (see **Defining Collaborations** on page 104).

6 Now you need to create Intelligent Queues to hold published Events (see **Creating Intelligent Queues** on page 105

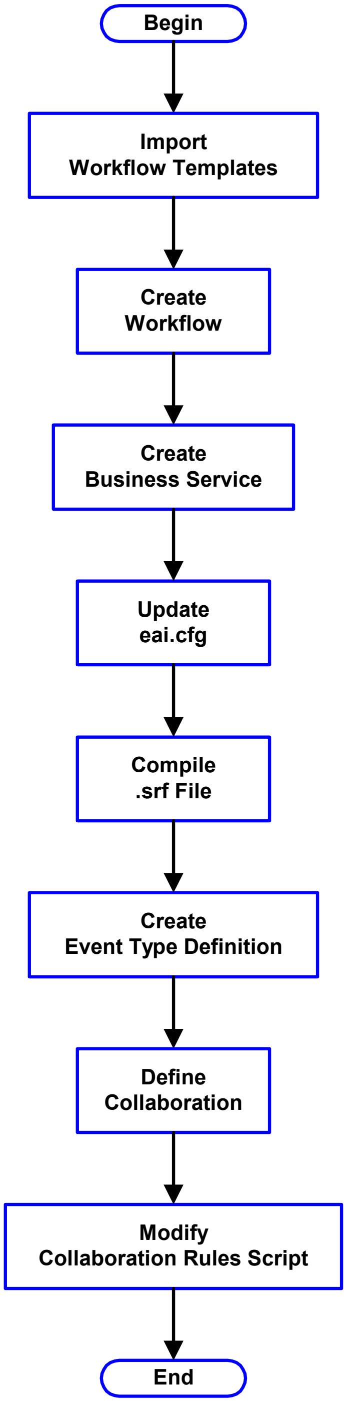
7 Finally, you must test your Schema. Once you have verified that it is working correctly, you may deploy it to your production environment.

Included with the Siebel EAI e*Way are several **SeeBeyond Workflow Templates**, which furnish pre-defined workflows within the Siebel application. More detailed implementation sequences, making use of these templates, appear on the following pages. See **e*Gate to Siebel** on page 81 and **Siebel to e*Gate** on page 82.

Also included with the e*Way are sample schema, which provide pre-defined templates that can be modified to suit your specific requirements.

6.2.2 e*Gate to Siebel

e*Gate-to-Siebel Implementation



Procedure

1 Import SeeBeyond HTTP Workflow templates. See **Importing SeeBeyond Workflow Templates** on page 90.

2 Make a copy of the workflow template that you want to use and modify it to fit your environment. See **Modifying SeeBeyond Workflow Templates** on page 92.

3 Set up the Siebel Business Services to execute the new Workflow. See **Setting Up SeeBeyond Workflow Processes** on page 93.

4 Update the configuration file **eai.cfg** with the above information. See step 8 in **Modifying SeeBeyond Workflow Templates** on page 92.

5 Compile the **.srf** file. See **Verifying the Integration Object DTD** on page 98.

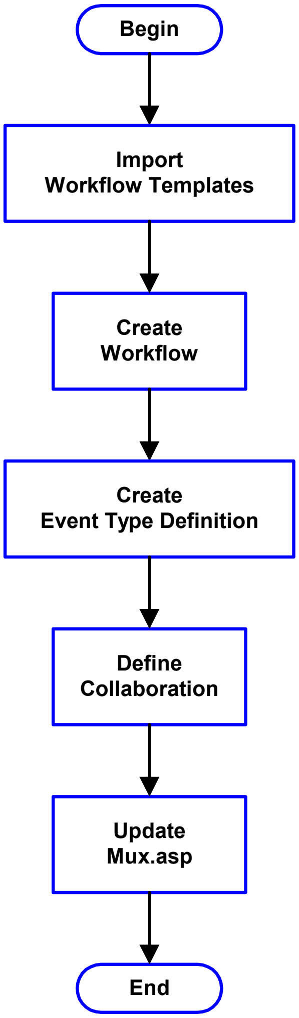
6 Create the Event Type Definitions corresponding to the Business Integration Objects that you want to manipulate. See **Creating Event Type Definitions** on page 101.

7 Define the e*Gate Collaboration to process the ETDs. See **Defining Collaborations** on page 104.

8 Modify the associated Collaboration Rules script. See **Defining Collaborations** on page 104.

6.2.3 Siebel to e*Gate

Siebel-to-e*Gate Implementation



Procedure

1 Import SeeBeyond HTTP Workflow templates. See **Importing SeeBeyond Workflow Templates** on page 90.

2 Make a copy of the Workflow template that you want to use and modify it to fit your environment. See **Modifying SeeBeyond Workflow Templates** on page 92.

3 Create the Event Type Definitions corresponding to the Business Integration Objects that you want to manipulate. See **Creating Event Type Definitions** on page 101.

4 Define the e*Gate Collaboration to process the ETDs. See **Defining Collaborations** on page 104.

5 Modify the **Mux.asp** file as necessary. See **Setting Up the MUX ASP** on page 27.

6.2.4 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.

6.3 SeeBeyond Workflow Templates

6.3.1 Overview

A set of SeeBeyond Workflow templates is included with the Siebel EAI e*Way. These workflow templates invoke the following workflow processes to map the data directly to or from the Siebel database.

- **SeeBeyond HTTP Delete** (see **Figure 29 on page 84**)
- **SeeBeyond HTTP Query** (see **Figure 30 on page 85**)
- **SeeBeyond HTTP Update** (see **Figure 31 on page 85**)

  Inserts or Updates according to the provided input values.
- **SeeBeyond HTTP Execute** (see **Figure 32 on page 86**)

  The preferred Workflow for receiving Siebel XML messages from e*Gate; combines **Delete**, **Query** and **Update** functionality into a single Workflow.
- **SeeBeyond HTTP Send** (see **Figure 33 on page 86**)
- **SeeBeyond HTTP Send Receive** (see **Figure 34 on page 87**)
- **SeeBeyond HTTP Post** (see **Figure 35 on page 87**)

  The preferred Workflow for sending Siebel XML messages to e*Gate; combines **Send** and **Send/Receive** functionality into a single Workflow.

The names entered above are used to set up the Business Service for the sample program supplied with the e*Way. You should use them as templates to create new processes corresponding to the Workflows you create for your own system.

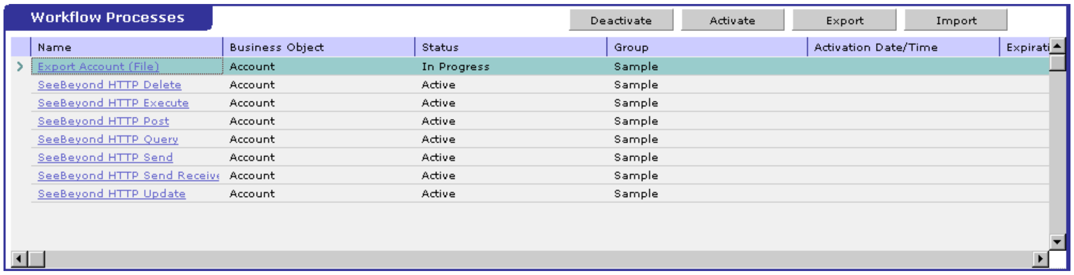
Examples of XML messages used with different Workflow templates are given in **Siebel XML Messages** on page 88.

Follow the Screens menu path

> **Siebel Workflow Administration > Workflow Processes > All Processes**

to display the installed Workflow templates, as shown in Figure 28.

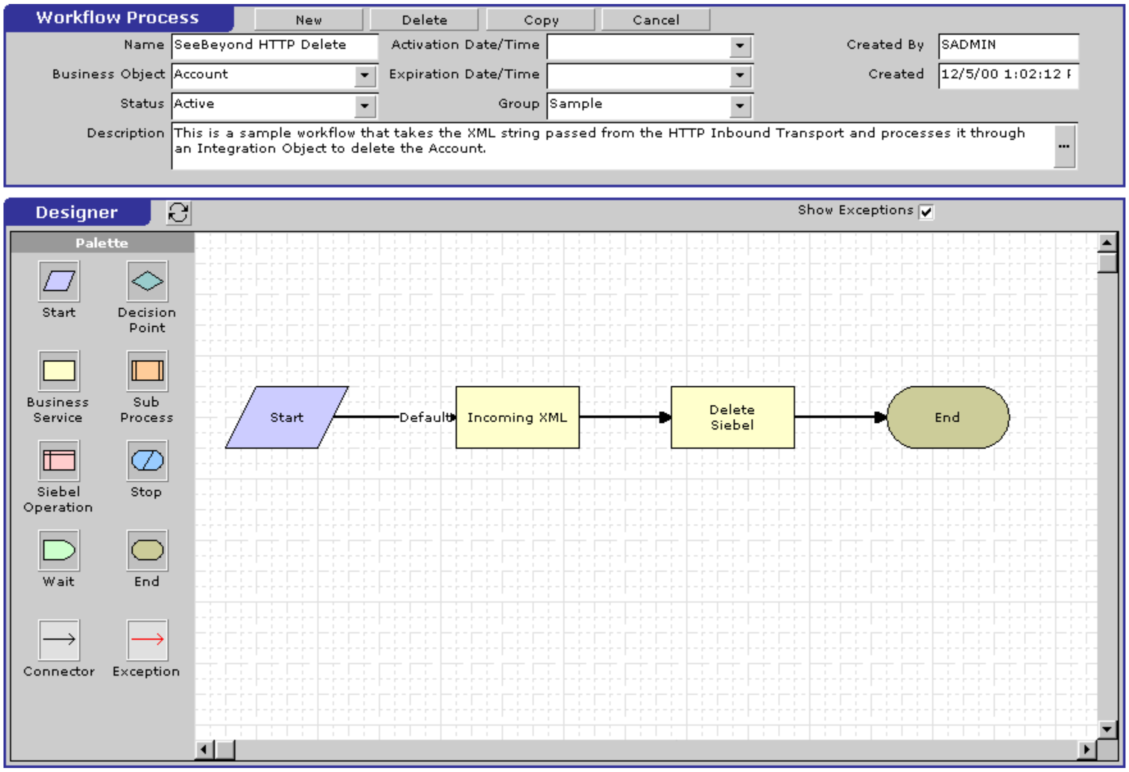
Figure 28 SeeBeyond Workflow Processes



Name	Business Object	Status	Group	Activation Date/Time	Expirati
Export Account (File)	Account	In Progress	Sample		
SeeBeyond HTTP Delete	Account	Active	Sample		
SeeBeyond HTTP Execute	Account	Active	Sample		
SeeBeyond HTTP Post	Account	Active	Sample		
SeeBeyond HTTP Query	Account	Active	Sample		
SeeBeyond HTTP Send	Account	Active	Sample		
SeeBeyond HTTP Send Receiv	Account	Active	Sample		
SeeBeyond HTTP Update	Account	Active	Sample		

Click on the process name to invoke a Workflow Process Designer display for that process, such as shown in Figures 12-18.

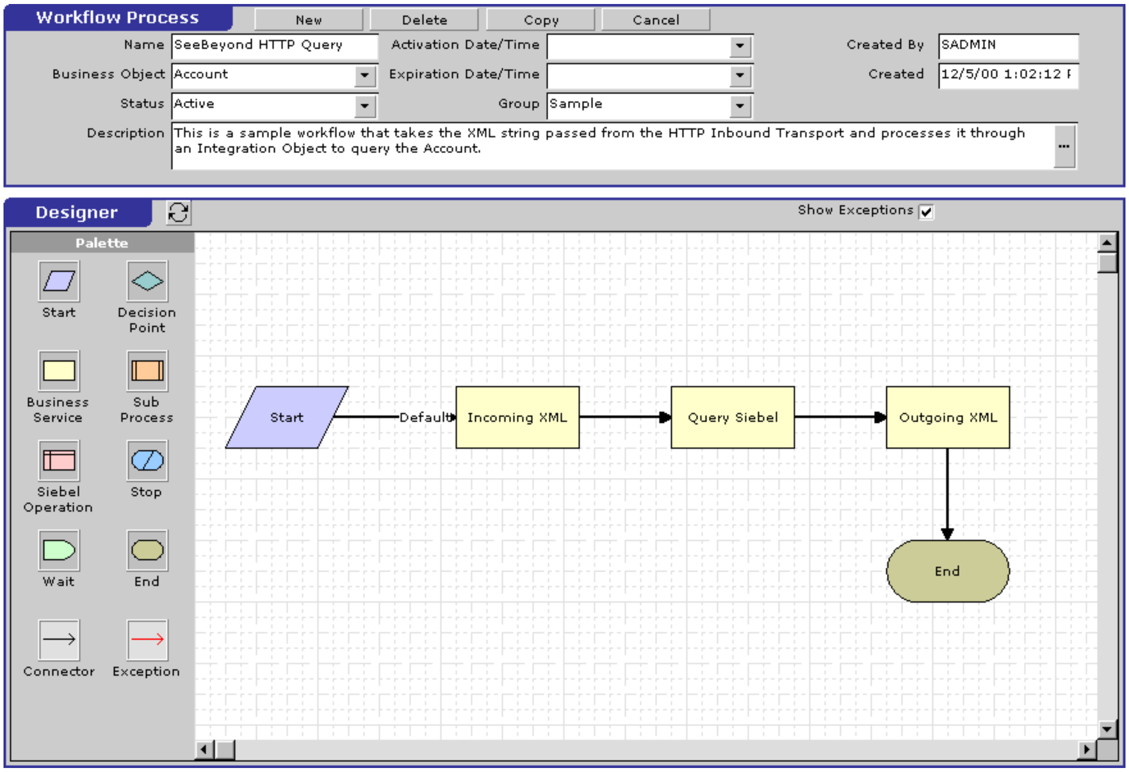
Figure 29 DELETE Workflow Template



Name	SeeBeyond HTTP Delete	Activation Date/Time		Created By	SADMIN
Business Object	Account	Expiration Date/Time		Created	12/5/00 1:02:12 F
Status	Active	Group	Sample		

Description: This is a sample workflow that takes the XML string passed from the HTTP Inbound Transport and processes it through an Integration Object to delete the Account.

Figure 30 QUERY Workflow Template



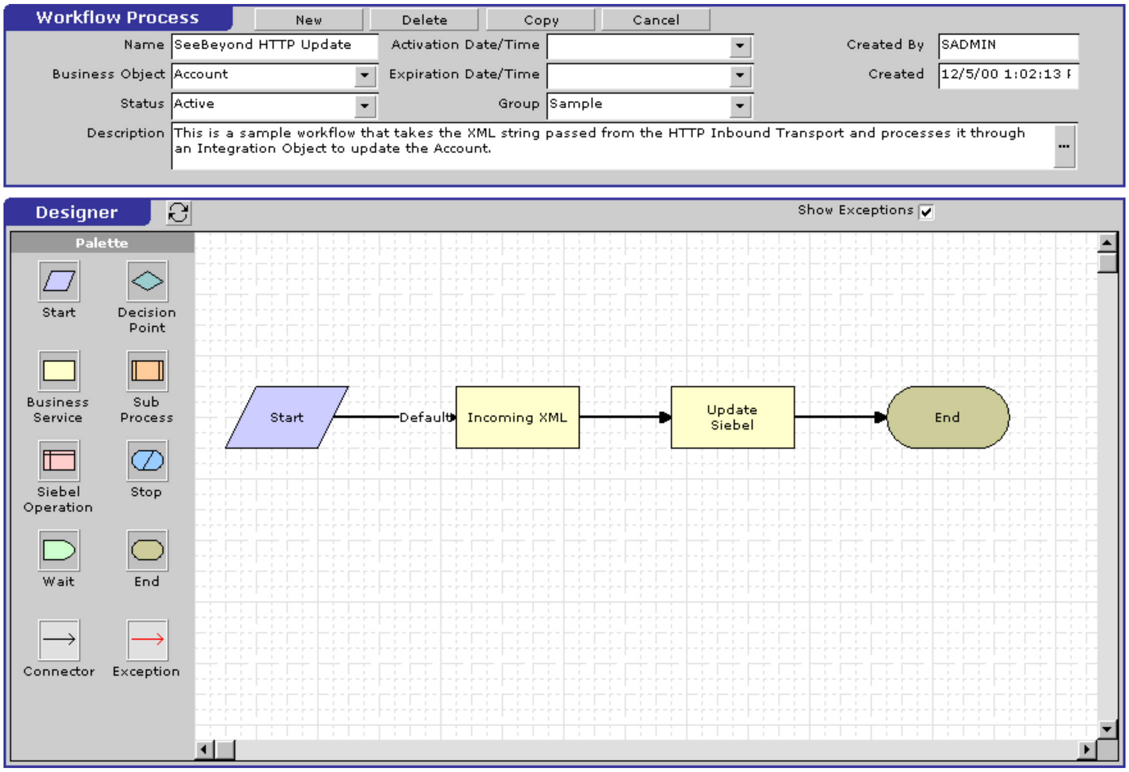
Workflow Process

Name	SeeBeyond HTTP Query	Activation Date/Time		Created By	SADMIN
Business Object	Account	Expiration Date/Time		Created	12/5/00 1:02:12 F
Status	Active	Group	Sample		

Description: This is a sample workflow that takes the XML string passed from the HTTP Inbound Transport and processes it through an Integration Object to query the Account.

Show Exceptions ☑

Figure 31 INSERT/UPDATE Workflow Template



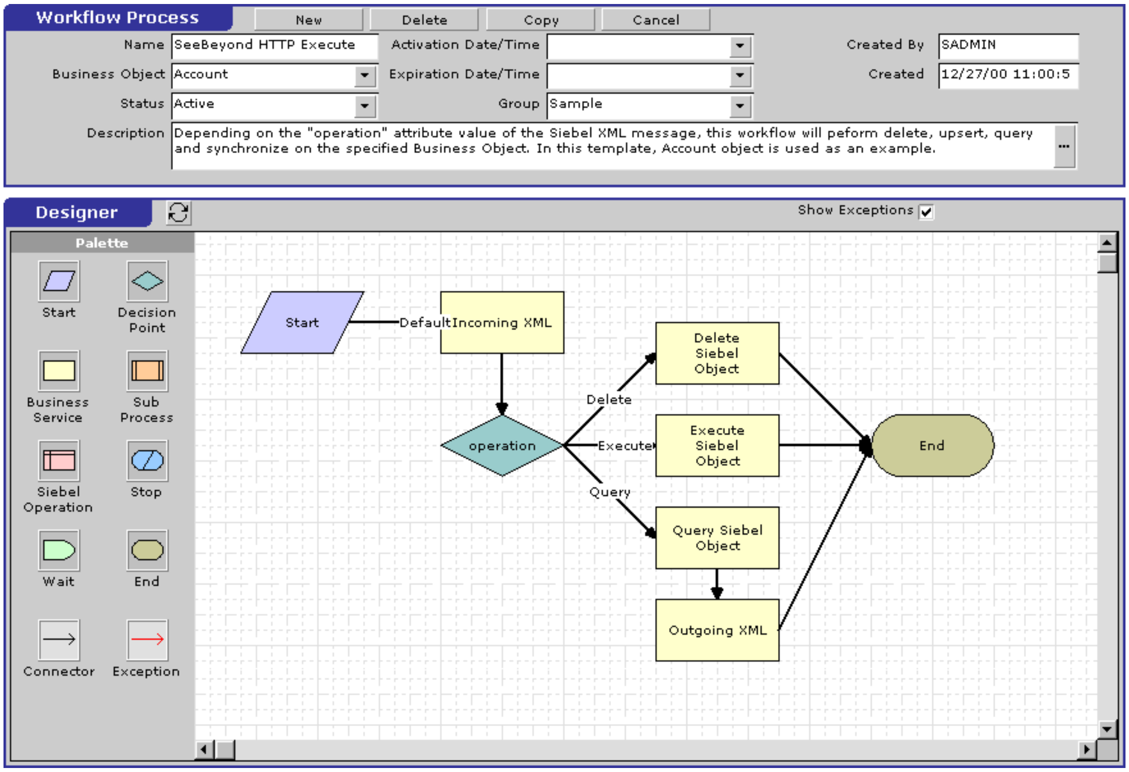
Workflow Process

Name	SeeBeyond HTTP Update	Activation Date/Time		Created By	SADMIN
Business Object	Account	Expiration Date/Time		Created	12/5/00 1:02:13 F
Status	Active	Group	Sample		

Description: This is a sample workflow that takes the XML string passed from the HTTP Inbound Transport and processes it through an Integration Object to update the Account.

Show Exceptions ☑

Figure 32 EXECUTE Workflow Template



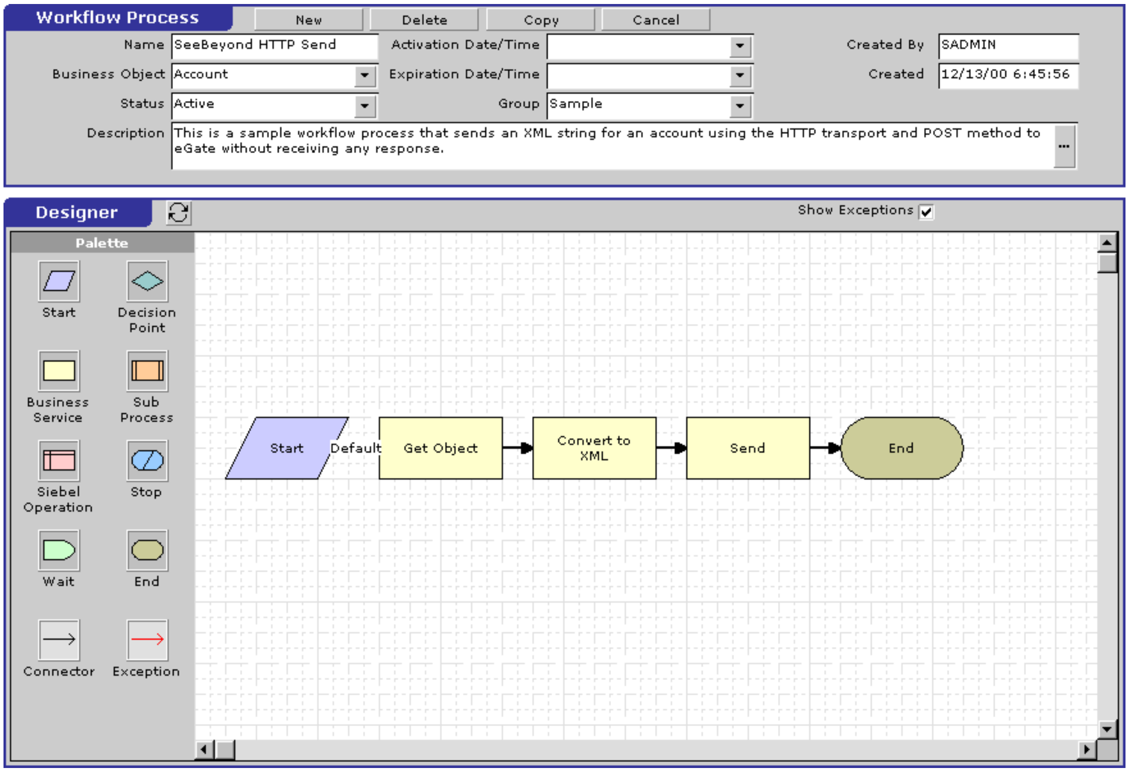
Workflow Process

Name	SeeBeyond HTTP Execute	Activation Date/Time		Created By	SADMIN
Business Object	Account	Expiration Date/Time		Created	12/27/00 11:00:5
Status	Active	Group	Sample		

Description: Depending on the "operation" attribute value of the Siebel XML message, this workflow will peform delete, upsert, query and synchronize on the specified Business Object. In this template, Account object is used as an example.

Show Exceptions ☑

Figure 33 SEND Workflow Template



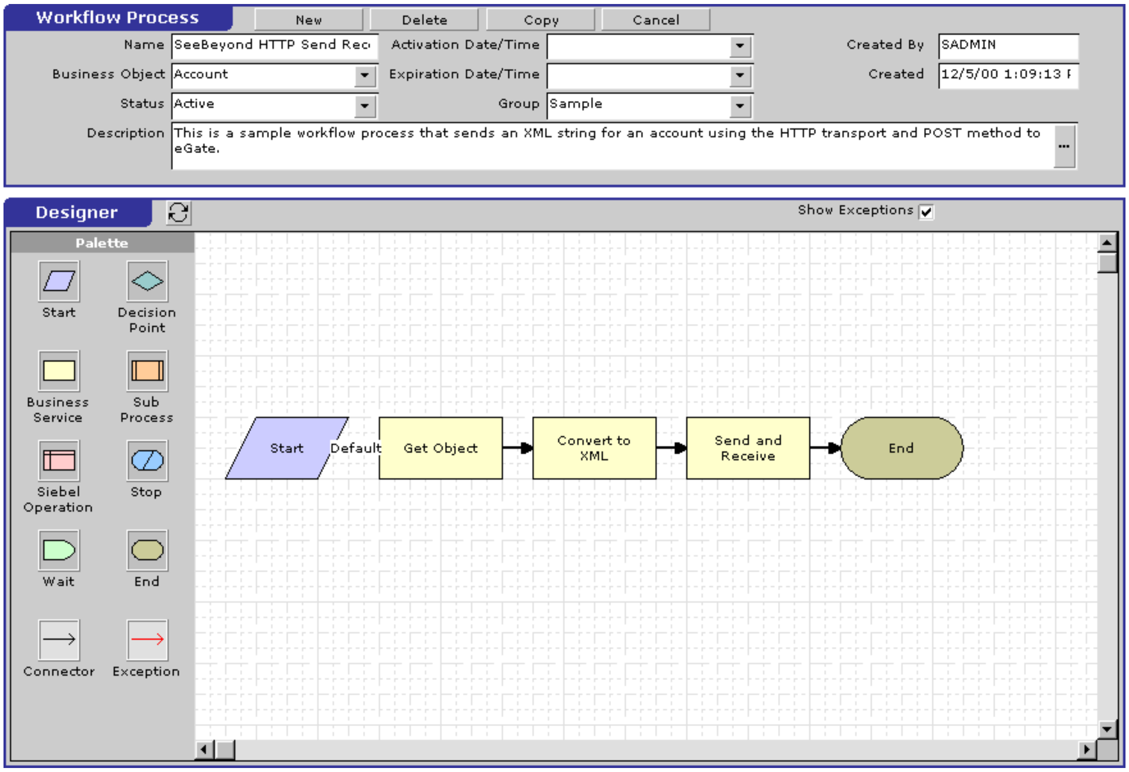
Workflow Process

Name	SeeBeyond HTTP Send	Activation Date/Time		Created By	SADMIN
Business Object	Account	Expiration Date/Time		Created	12/13/00 6:45:56
Status	Active	Group	Sample		

Description: This is a sample workflow process that sends an XML string for an account using the HTTP transport and POST method to eGate without receiving any response.

Show Exceptions ☑

Figure 34 SEND/RECEIVE Workflow Template



Workflow Process

Name: SeeBeyond HTTP Send Rec

Business Object: Account

Status: Active

Activation Date/Time:

Expiration Date/Time:

Group: Sample

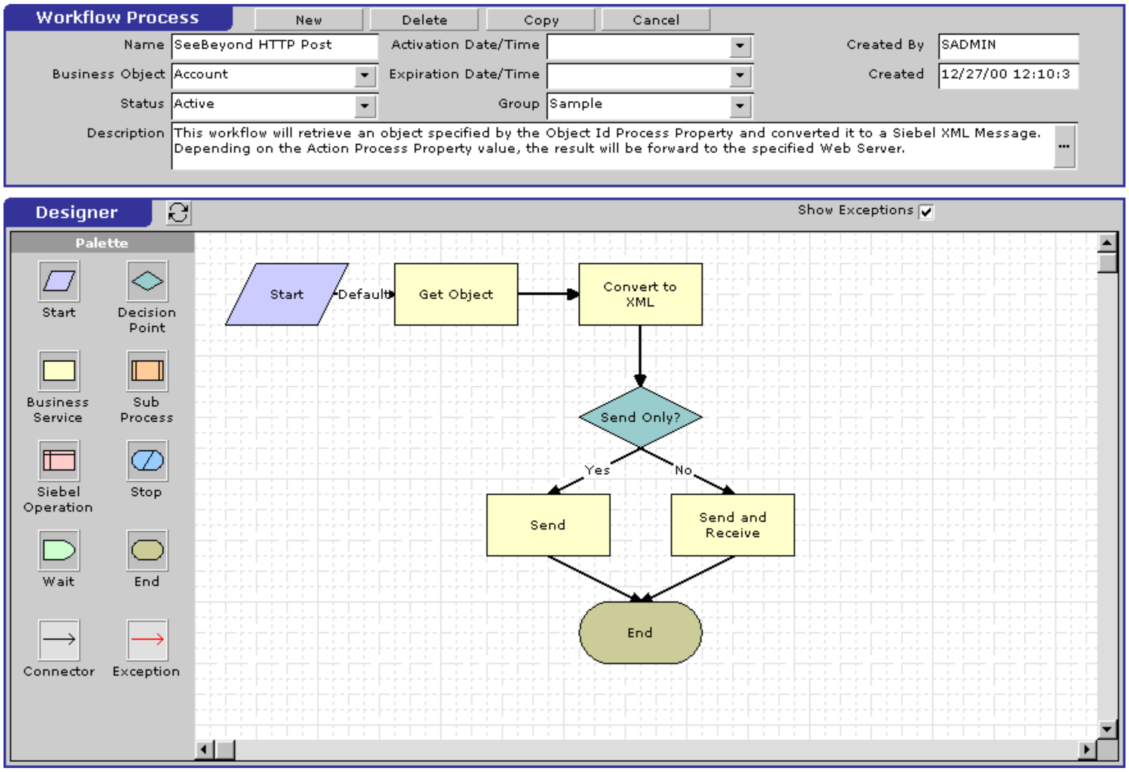
Created By: SADMIN

Created: 12/5/00 1:09:13 P

Description: This is a sample workflow process that sends an XML string for an account using the HTTP transport and POST method to eGate.

Show Exceptions ☑

Figure 35 POST Workflow Template



Workflow Process

Name: SeeBeyond HTTP Post

Business Object: Account

Status: Active

Activation Date/Time:

Expiration Date/Time:

Group: Sample

Created By: SADMIN

Created: 12/27/00 12:10:3

Description: This workflow will retrieve an object specified by the Object Id Process Property and converted it to a Siebel XML Message. Depending on the Action Process Property value, the result will be forward to the specified Web Server.

Show Exceptions ☑

6.3.2 Siebel XML Messages

Format

A Siebel XML Message used by Siebel EAI e*Way has the following format:

Header/Prefix

Integration Object (in XML format)

Footer/Suffix

where:

Header =

```
<SiebelMessage MessageId="" MessageType="Integration Object"
IntObjectName="(Name of Integration Object)" operation=(action)>
```

Footer =

```
</SiebelMessage>
```

and **(action)** can be any of the following values:

- **delete**
- **query**
- **upsert**

Note: **operation=(action)** *is used only with the* **EXECUTE** *workflow template.*

Examples

Example 1

The following Siebel XML message specifies that the Integration Object that we are dealing with is **Sample Account**. If we send this message to Siebel EAI using the **INSERT/UPDATE** workflow template, either a new record is generated or an existing record is updated.

```
<SiebelMessage  MessageId=""  MessageType="Integration Object"
    IntObjectName="Sample Account">
<ListofSampleAccount>
<Account>
<Name>A. K. Parker Distribution</Name>
<Location>HQ-Distribution</Location>
<Organization>North American Organization</Organization>
<Division></Division>
<CurrencyCode>USD</CurrencyCode>
<Description>This is THE key account in the AK Parker Family</
    Description>
<HomePage>www.parker.com</HomePage>
<LineofBusiness>Manufacturing</LineofBusiness>
</Account>
</ListofSampleAccount>
```

```
</SiebelMessage>
```

Example 2

The following Siebel XML message specifies that the Integration Object that we are dealing with is **Sample Account**. If we send this message to Siebel EAI using the **QUERY** workflow template, it returns the object that matches the Name **A. K***

```
<SiebelMessage  MessageId=""  MessageType="Integration Object"
    IntObjectName="Sample Account">
<ListofSampleAccount>
<Account>
<Name>A. K*</Name>
</Account>
</ListofSampleAccount>
</SiebelMessage>
```

Example 3

The following Siebel XML message provides an example of how to use the **operation** attribute with the **Execute** workflow. Here we send the message to Siebel EAI using the **EXECUTE** workflow template to perform a **query** operation. The result is the same as in Example 2.

```
<SiebelMessage  MessageId=""  MessageType="Integration Object"
    IntObjectName="Sample Account" operation=query>
<ListofSampleAccount>
<Account>
<Name>A. K*</Name>
</Account>
</ListofSampleAccount>
</SiebelMessage>
```

6.3.3 Importing SeeBeyond Workflow Templates

Note: *If you are using Siebel 2000 Japanese, the file* **SeeBeyondHTTPWorkflowJPN.xml** *replaces the file* **SeeBeyondHTTPWorkflow.xml** *in the following procedure.*

To Import the SeeBeyond Workflow Templates

1. On the e*Gate installation CD-ROM, go to:

   ```
   \setup\addons\ewsiebelhttp\common.taz
   ```

2. Decompress the **.taz** file and open the **.tar** file contained within.
3. Extract the file **SeeBeyondHTTPWorkflow.xml** to an appropriate directory.
4. Start **Siebel EAI Client** and select **Siebel Sales**.
5. Follow the **Screens** menu path:

   Siebel Workflow Administration > Workflow Processes > All Processes

6. Click **Import** and browse to the directory that contains **SeeBeyondHTTPWorkflow.xml**.
7. Click **Open** to begin importing the Workflow template.
8. Check or set up the following configuration files:

   - In the file **SWEApp\eapps.cfg**, verify that the following section is specified correctly:

     ```
     [/eai]
     ConnectString = siebel.TCPIP.none.none://<Your Gateway
     Server>:3230/ <Your Enterprise Server>/eaiObjMgr/<Your App
     Server>
     EnableExtServiceOnly = TRUE
     ```

   - For the **e*Gate-to-Siebel** sample, add the following sections in the file **siebsrvr\eai.cfg**:

     ```
     [HTTP Services]
     ...
     SEEBEYOND_HTTP_DELETE = SB_HTTP_DELETE
     SEEBEYOND_HTTP_QUERY = SB_HTTP_QUERY
     SEEBEYOND_HTTP_UPDATE = SB_HTTP_UPDATE
     SEEBEYOND_HTTP_EXECUTE = SB_HTTP_EXECUTE

     [SB_HTTP_DELETE]
     Mode = Document
     Service = SeeBeyond HTTP Delete
     Method = RunProcess

     [SB_HTTP_QUERY]
     Mode = Document
     Service = SeeBeyond HTTP Query
     Method = RunProcess

     [SB_HTTP_UPDATE]
     Mode = Document
     Service = SeeBeyond HTTP Update
     Method = RunProcess

     [SB_HTTP_EXECUTE]
     ```

```
Mode = Document
Service = SeeBeyond HTTP Execute
Method = RunProcess
```

6.3.4 Modifying SeeBeyond Workflow Templates

Note: *The SeeBeyond Workflow templates provided with the e*Way use* **Account** *as the Business Object—you must modify them for use with a different Business Object.*

To Modify a SeeBeyond Workflow Template

1. Log in to **Siebel Client 6.0**, designating the appropriate Siebel server.
2. Follow the **Screens** menu path:

   Siebel Workflow Administration > Workflow Processes > All Processes
3. Highlight the SeeBeyond Workflow Process template you want to modify.
4. Right-click and select **Copy Record**.
5. Rename the copied Process.
6. Specify the Business Object to which you want to apply the template, and any other fields that may be necessary (for example, Description).
7. After modifying a Workflow template you must create the Business Service to execute it, using the supplied Workflow processes as templates. This procedure is described in the following section.
8. A new Services section should be added to your **siebsrvr\eai.cfg** file, as shown in the preceding section.

   For example, if you have a Business Service named Employee Execute, you should add the following lines to the **eai.cfg** file:

   ```
   [HTTP Services]
   ...
   EMPLOYEE_EXECUTE = EE

   [EE]
   Mode = Document
   Service = Employee Execute
   Method = RunProcess
   ```

6.3.5 Setting Up SeeBeyond Workflow Processes

The Workflow processes invoked by the SeeBeyond Workflow Templates must be set up in Siebel Business Services.

Note: *The names entered in step 8 are used to set up the Business Service for the sample program supplied with the e*Way. You should use them as templates to create new processes corresponding to the Workflows you create for your own system.*

To set up the Business Service to execute the Workflow:

1 Make sure the following services are running:

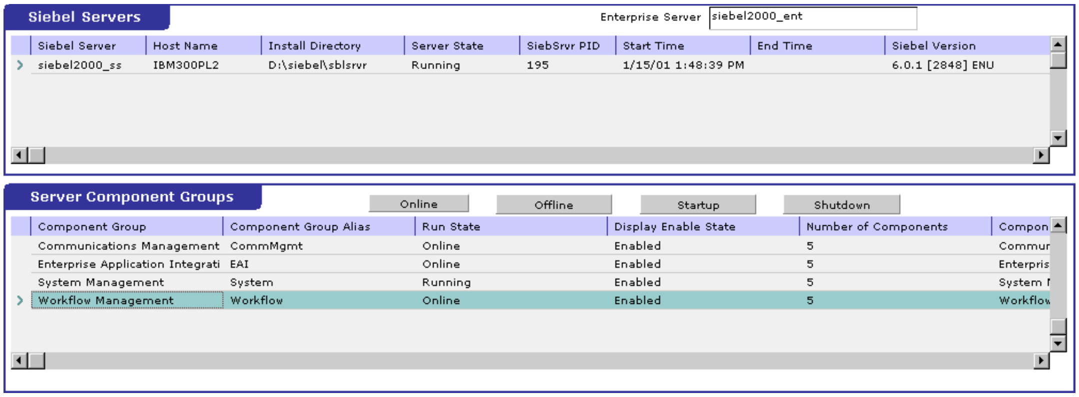
- Siebel Gateway Server
- Siebel Server
- World Wide Web Publishing Service

2 Log in to Siebel Sales 6.0.

3 Follow the **Screens** menu path:

Server Administration > **Servers** > **Server Component Groups**

Figure 36 Server Component Groups



Siebel Servers

Enterprise Server: siebel2000_ent

Siebel Server	Host Name	Install Directory	Server State	SiebSrvr PID	Start Time	End Time	Siebel Version
siebel2000_ss	IBM300PL2	D:\siebel\sblsrvr	Running	195	1/15/01 1:48:39 PM		6.0.1 [2848] ENU

Server Component Groups

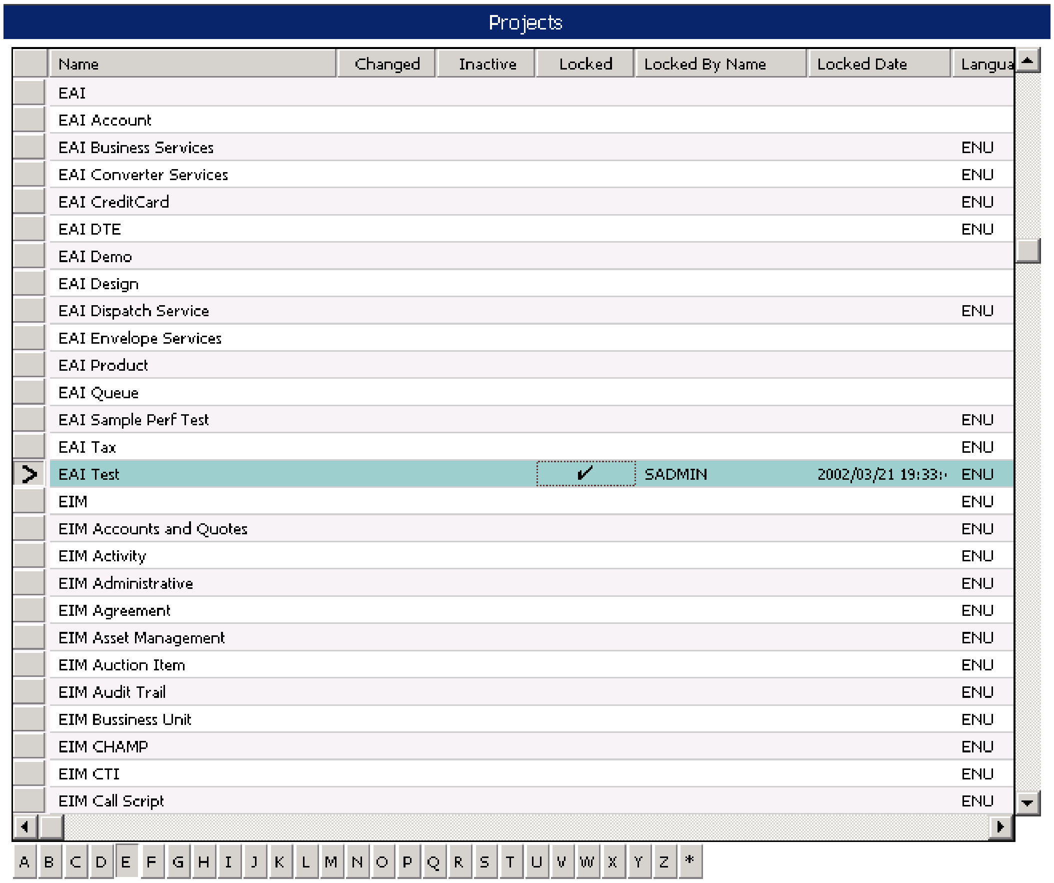
Component Group	Component Group Alias	Run State	Display Enable State	Number of Components	Compon
Communications Management	CommMgmt	Online	Enabled	5	Commur
Enterprise Application Integrati	EAI	Online	Enabled	5	Enterpris
System Management	System	Running	Enabled	5	System I
Workflow Management	Workflow	Online	Enabled	5	Workflow

4 Make sure that **Workflow Management** is **Online** and **Enabled**.

5 Log in to Siebel Tools 6.0 and designate the server as the database by entering **sadmin**, **sadmin**, **server**.

6 In Object Explorer, go to **Siebel Objects** > **Project** and lock the project (see Figure 37).

Figure 37 Lock Project

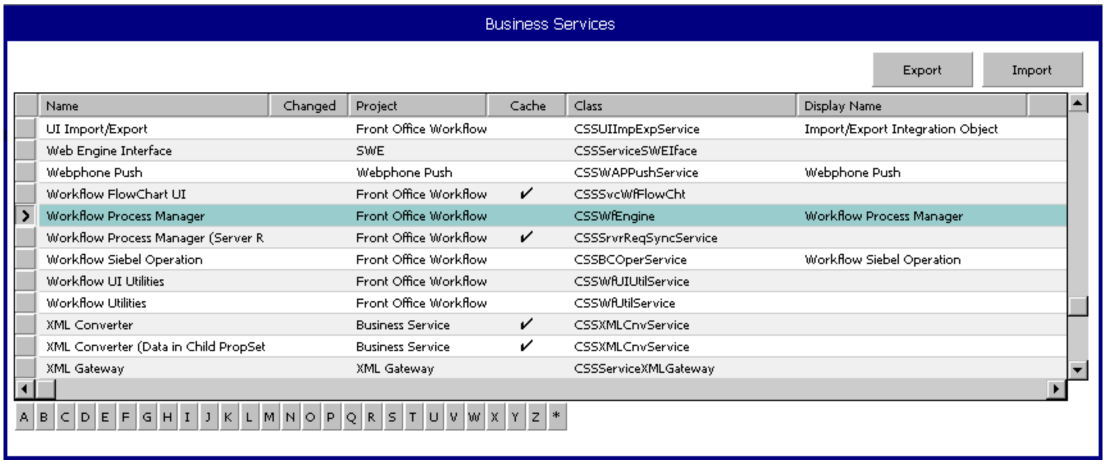


Projects

Name	Changed	Inactive	Locked	Locked By Name	Locked Date	Langua
EAI						
EAI Account						
EAI Business Services						ENU
EAI Converter Services						ENU
EAI CreditCard						ENU
EAI DTE						ENU
EAI Demo						
EAI Design						
EAI Dispatch Service						ENU
EAI Envelope Services						
EAI Product						
EAI Queue						
EAI Sample Perf Test						ENU
EAI Tax						ENU
EAI Test			✓	SADMIN	2002/03/21 19:33:	ENU
EIM						ENU
EIM Accounts and Quotes						ENU
EIM Activity						ENU
EIM Administrative						ENU
EIM Agreement						ENU
EIM Asset Management						ENU
EIM Auction Item						ENU
EIM Audit Trail						ENU
EIM Bussiness Unit						ENU
EIM CHAMP						ENU
EIM CTI						ENU
EIM Call Script						ENU

7 In Object Explorer, go to **Business Service**, make a copy of **Workflow Process Manager** (menu path **Edit** > **Copy Record**).

Figure 38 Business Services View - Workflow Process Manager



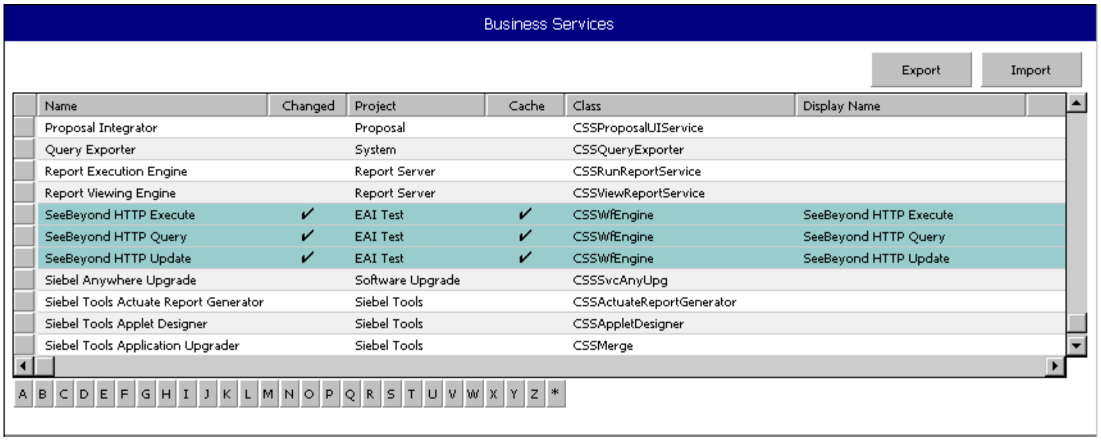
Name	Changed	Project	Cache	Class	Display Name
UI Import/Export		Front Office Workflow		CSSUIImpExpService	Import/Export Integration Object
Web Engine Interface		SWE		CSSServiceSWEIface	
Webphone Push		Webphone Push		CSSWAPPushService	Webphone Push
Workflow FlowChart UI		Front Office Workflow	✓	CSSSvcWfFlowCht	
Workflow Process Manager		Front Office Workflow		CSSWfEngine	Workflow Process Manager
Workflow Process Manager (Server R		Front Office Workflow	✓	CSSSrvrReqSyncService	
Workflow Siebel Operation		Front Office Workflow		CSSBCOperService	Workflow Siebel Operation
Workflow UI Utilities		Front Office Workflow		CSSWfUIUtilService	
Workflow Utilities		Front Office Workflow		CSSWfUtilService	
XML Converter		Business Service	✓	CSSXMLCnvService	
XML Converter (Data in Child PropSet		Business Service	✓	CSSXMLCnvService	
XML Gateway		XML Gateway		CSSServiceXMLGateway	

8 Type the Process Name into the **Name** and **Display Names** fields, as shown in Figure 39 (this name is specified in the **eai.cfg** file).

For e*Gate-to-Siebel operation, perform this step for:

- **SeeBeyond HTTP Delete**
- **SeeBeyond HTTP Execute**
- **SeeBeyond HTTP Query**
- **SeeBeyond HTTP Update**

Figure 39 Business Services View - Renamed Fields



Name	Changed	Project	Cache	Class	Display Name
Proposal Integrator		Proposal		CSSProposalUIService	
Query Exporter		System		CSSQueryExporter	
Report Execution Engine		Report Server		CSSRunReportService	
Report Viewing Engine		Report Server		CSSViewReportService	
SeeBeyond HTTP Execute	✓	EAI Test	✓	CSSWfEngine	SeeBeyond HTTP Execute
SeeBeyond HTTP Query	✓	EAI Test	✓	CSSWfEngine	SeeBeyond HTTP Query
SeeBeyond HTTP Update	✓	EAI Test	✓	CSSWfEngine	SeeBeyond HTTP Update
Siebel Anywhere Upgrade		Software Upgrade		CSSSvcAnyUpg	
Siebel Tools Actuate Report Generator		Siebel Tools		CSSActuateReportGenerator	
Siebel Tools Applet Designer		Siebel Tools		CSSAppletDesigner	
Siebel Tools Application Upgrader		Siebel Tools		CSSMerge	

9 In the **Repository** > **Business Service** > **Business Service User Props** view:

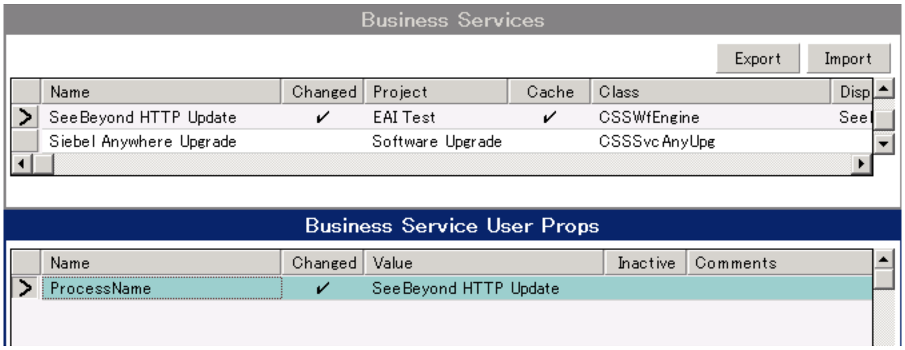
A Type **ProcessName** into the **Name** field.

B Type the actual Process Name into the **Value** field (see Figure 40).

For e*Gate-to-Siebel operation, perform this step for:

- **SeeBeyond HTTP Delete**
- **SeeBeyond HTTP Execute**
- **SeeBeyond HTTP Query**
- **SeeBeyond HTTP Update**

Figure 40 Business Services User Properties



	Name	Changed	Project	Cache	Class	Disp
>	SeeBeyond HTTP Update	✓	EAI Test	✓	CSSWfEngine	See
	Siebel Anywhere Upgrade		Software Upgrade		CSSSvcAnyUpg	

	Name	Changed	Value	Inactive	Comments
>	ProcessName	✓	SeeBeyond HTTP Update		

The Workflow Processes you create in the Business Services are similar to those shown in Figure 39.

6.4 Creating a Schema

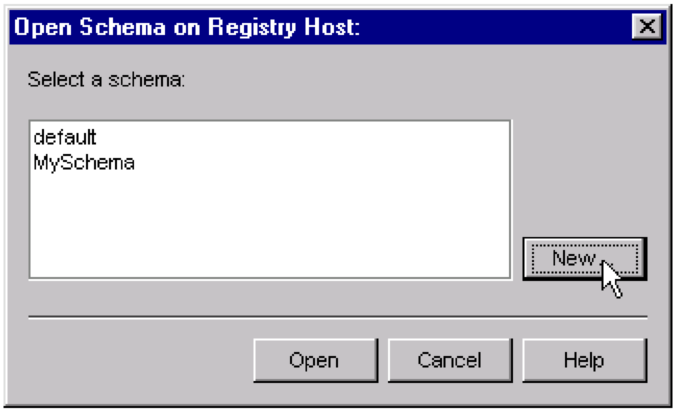
A schema is the structure that defines e*Gate system parameters and the relationships between components within the e*Gate system. Schemas can span multiple hosts.

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

To select or create a schema

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

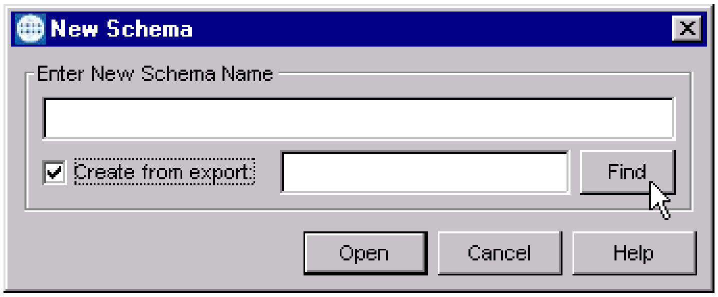
Figure 41 Open Schema Dialog



default
MySchema

2 Clicking **New** invokes the **New Schema** dialog box (Figure 42).

Figure 42 New Schema Dialog



☑ Create from export:

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.

6.5 Generating the Integration Object DTD

To generate the DTD

1. In Siebel Tools, click on an Integration Object to activate it.
2. Click **Generate Schema**, which displays the initial page of the Generate Schema Wizard.
3. Select the **EAI XML DTD Generator** business service.
4. Select a location to store the resulting file.
5. Click **Finish**.

The Wizard generates an XML DTD of the Integration Object you selected. You can use this DTD to create an ETD using the SeeBeyond XML Converter/ETD Builder, as described in **Using the ETD Editor's Build Tool** on page 101.

Note: *There is a defect in Siebel's Integration Object DTD. Element names are* ***not*** *unique. It has been reported as a product defect #12-1TQJN7. Following is the workaround recommended by Siebel:*

- Do **not** specify an XML Parent Element name in the Integration Component
- Add the prefix **ListOf** to the XML Tag

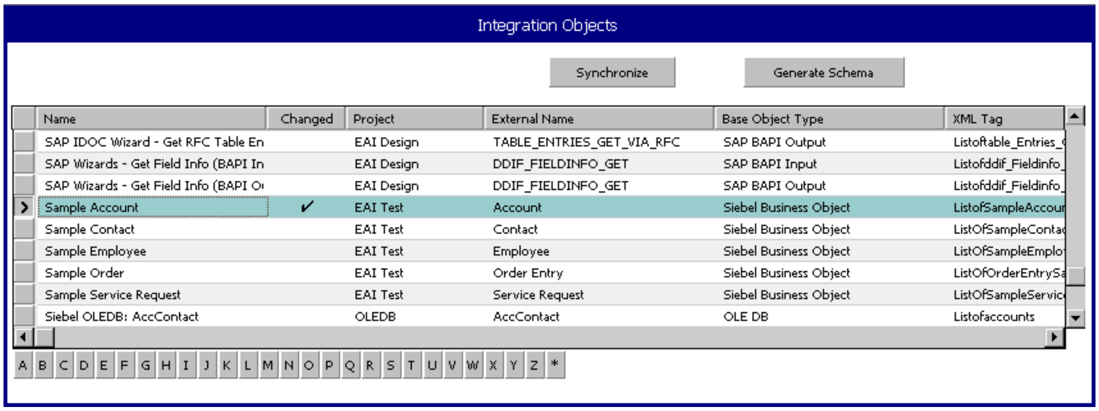
6.6 Verifying the Integration Object DTD

The next step is to confirm that the Integration Object DTD is generated correctly. You should export the DTD and run the SeeBeyond XML Converter/ETD Builder to verify that it can generate the Event Type Definition correctly. An incorrect ETD build usually indicates that the Siebel DTD has a repeated element name. In this case, you need to modify the Integration Object. See the *Note* in **Generating the Integration Object DTD** on page 98.

The supplied sample program uses the **Sample Account** Integration Object, the integrity of which is verified as follows:

1. Navigate to the **Integration Object** view and select **Sample Account**.

Figure 43 Integration Object View - Sample Account



Integration Objects

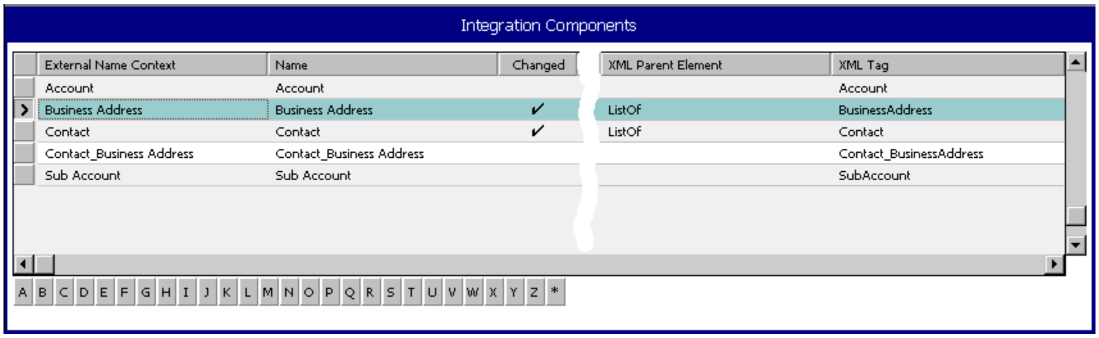
Name	Changed	Project	External Name	Base Object Type	XML Tag
SAP IDOC Wizard - Get RFC Table En		EAI Design	TABLE_ENTRIES_GET_VIA_RFC	SAP BAPI Output	Listoftable_Entries_
SAP Wizards - Get Field Info (BAPI In		EAI Design	DDIF_FIELDINFO_GET	SAP BAPI Input	Listofddif_Fieldinfo_
SAP Wizards - Get Field Info (BAPI O		EAI Design	DDIF_FIELDINFO_GET	SAP BAPI Output	Listofddif_Fieldinfo_
Sample Account	✓	EAI Test	Account	Siebel Business Object	ListofSampleAccour
Sample Contact		EAI Test	Contact	Siebel Business Object	ListOfSampleContac
Sample Employee		EAI Test	Employee	Siebel Business Object	ListOfSampleEmplo
Sample Order		EAI Test	Order Entry	Siebel Business Object	ListOfOrderEntrySa
Sample Service Request		EAI Test	Service Request	Siebel Business Object	ListOfSampleServic
Siebel OLEDB: AccContact		OLEDB	AccContact	OLE DB	Listofaccounts

2 Navigate to the **Integration Object** > **Integration Component** view.

A In the **Business Address/XML Parent Element** field, type **ListOf**.

B In the **Contact/XML Parent Element** field, type **ListOf**.

Figure 44 Integration Object > Integration Component View



Integration Components

External Name Context	Name	Changed	XML Parent Element	XML Tag
Account	Account			Account
Business Address	Business Address	✓	ListOf	BusinessAddress
Contact	Contact	✓	ListOf	Contact
Contact_Business Address	Contact_Business Address			Contact_BusinessAddress
Sub Account	Sub Account			SubAccount

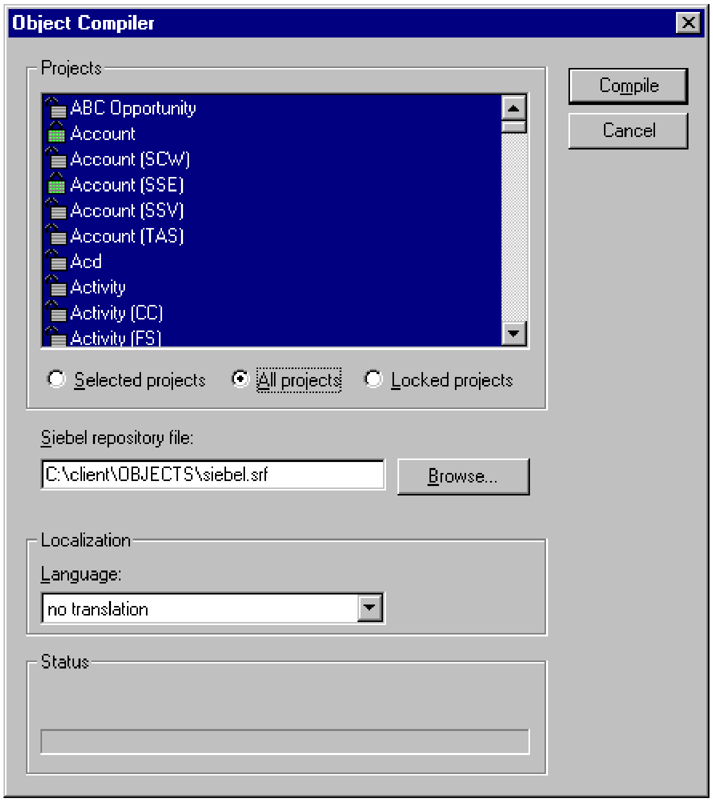
3 Stop the following services, in this order:

A Siebel Server.

B Siebel Gateway Name Server.

4 Follow the menu path **Repository** > **Compile**, which invokes the **Object Compiler** dialog box.

Figure 45 Objects Compiler Dialog Box



Object Compiler

Projects

- ABC Opportunity
- Account
- Account (SCW)
- Account (SSE)
- Account (SSV)
- Account (TAS)
- Acd
- Activity
- Activity (CC)
- Activity (FS)

○ Selected projects ● All projects ○ Locked projects

Siebel repository file: C:\client\OBJECTS\siebel.srf

Localization

Language: no translation

Status

5 If you have completed all projects, select the **All Projects** option; otherwise, select **Locked Projects** to shorten the compilation time.

6 Select the Siebel repository file **\client\OBJECTS\siebel.srf**.

7 Click **Compile** and copy the Siebel **.srf** file to the **siebel\sblsrvr\OBJECTS** directory.

8 Start the following services, in this order:

A Siebel Gateway Name Server.

B Siebel Server.

9 Verify that the EAI Object Manager is running.

6.7 Creating Event Type Definitions

Before e*Gate can process any data to or from a Siebel EAI system, you must create an Event Type Definition to package and route that data within the e*Gate system. The ETD is derived from a Siebel Integration Object Data Type Definition (DTD). See the *e*Gate Integrator User's Guide* for additional information about Event Type Definitions and the e*Gate ETD Editor.

6.7.1 Using the ETD Editor's Build Tool

The Event Type Definition Editor's Build tool automatically creates an Event Type Definition file based upon structural metadata. 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.*

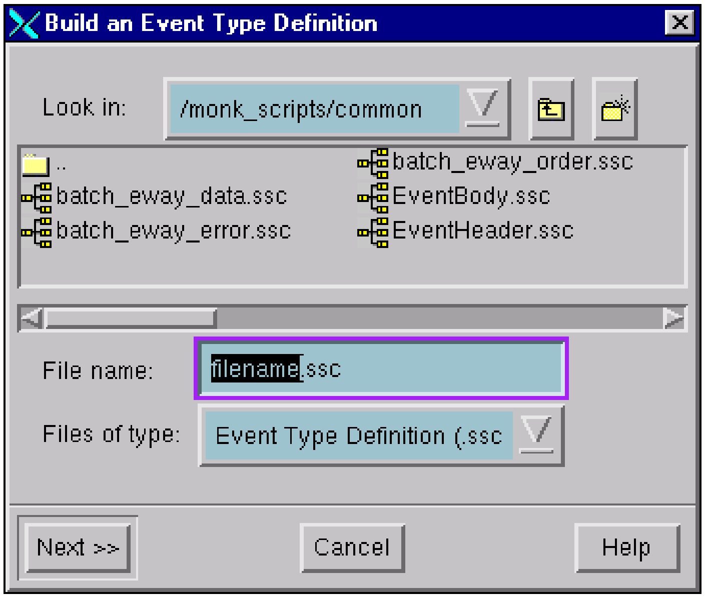
To create an Event Type Definition using the Build tool

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 46 Build Event Type Definition Dialog



Build an Event Type Definition

Look in: /monk_scripts/common

..
batch_eway_data.ssc
batch_eway_error.ssc
batch_eway_order.ssc
EventBody.ssc
EventHeader.ssc

File name: filename.ssc

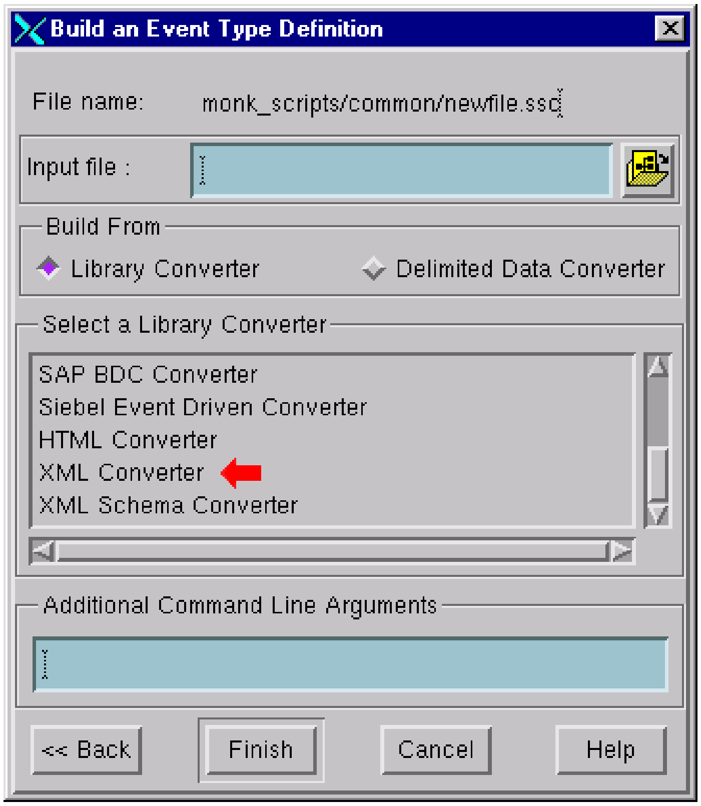
Files of type: Event Type Definition (.ssc

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 47.

Figure 47 Building the ETD



Build an Event Type Definition

File name: monk_scripts/common/newfile.ssc

Input file :

Build From

● Library Converter ○ Delimited Data Converter

Select a Library Converter
SAP BDC Converter
Siebel Event Driven Converter
HTML Converter
XML Converter
XML Schema Converter
Additional Command Line Arguments
<< Back
Finish
Cancel
Help

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

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

7 Leave the *Input file* field blank—the XML Converter does not use this field.

8 Under **Additional Command Line Arguments**, type the following:

`-xml` *input_file*

where *input_file* is either:

- the XML filename (including the path)
- the URL to the XML page

9 Click **Finish**. The Build tool creates the ETD.

If your input XML page contains more than one form, the Build tool creates multiple ETD files, one for each form. The name of each file is the file name you entered in step 4 above with an underscore and number appended to it, starting with zero. For example: **xml_0.ssc**, **xml_1.ssc**, and so on.

If your XML page contained only a single form, the ETD Editor opens the resulting ETD file automatically at the conclusion of the conversion process. If multiple ETD files were created, you must open each file manually.

6.7.2 Assigning ETDs to Event Types

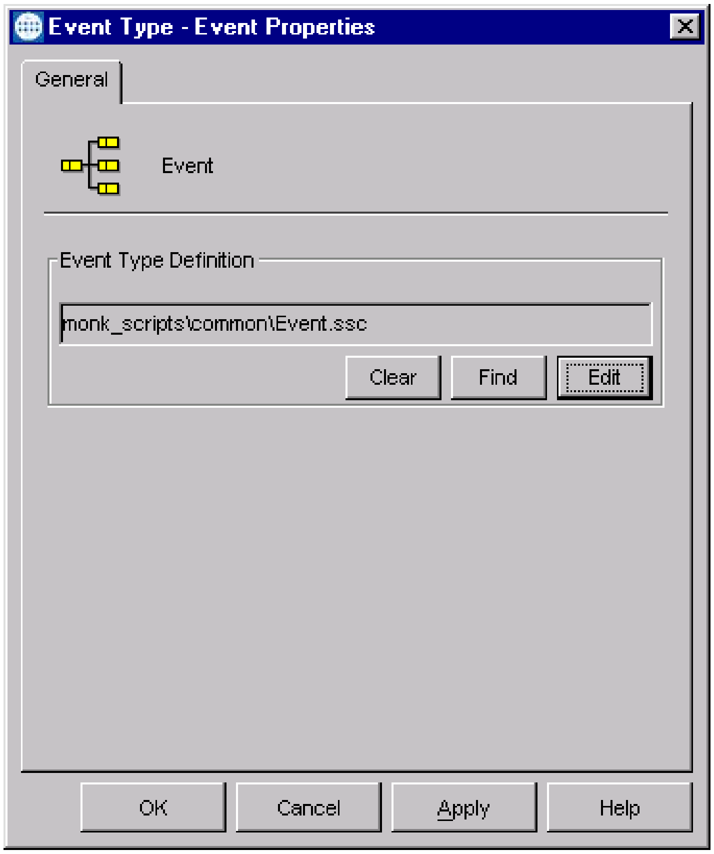
After you have created the e*Gate system's ETD files, you can assign them to Event Types you have already created.

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 48.

Figure 48 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.

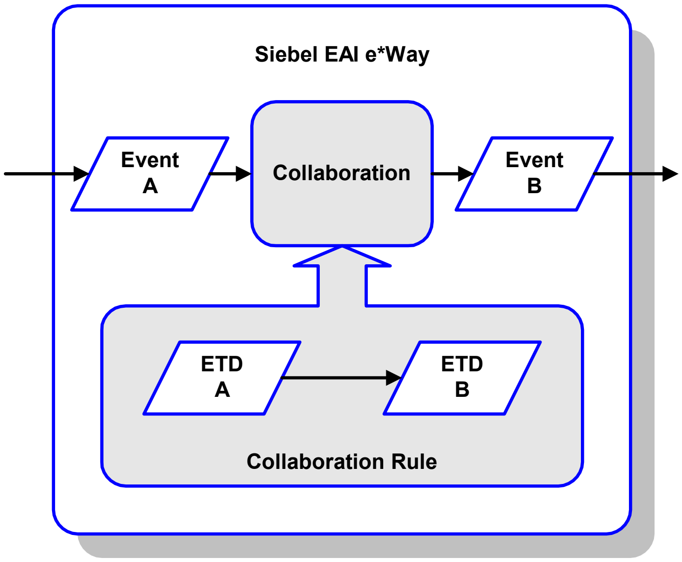
Each Event Type is associated with the specified Event Type Definition.

6.8 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 49 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**.

6.9 Creating Intelligent Queues

IQs are components that provide nonvolatile storage for Events within the e*Gate system as they pass from one component to another. IQs are *intelligent* in that they are more than just a “holding tank” for Events. They actively record information about the current state of Events.

Each schema must have an IQ Manager before you can add any IQs to it. You must create at least one IQ per schema for published Events within the e*Gate system. Note that e*Ways that publish Events externally do not need IQs.

For more information on how to add and configure IQs and IQ Managers, see the *e*Gate Integrator System Administration and Operations Guide.* See the *e*Gate Integrator Intelligent Queue Services Reference Guide* and the *SeeBeyond JMS Intelligent Queue User’s Guide* for complete information on working with IQs.

6.10 Using the e*Way

In the following examples, we assume that you have already imported the SeeBeyond HTTP Workflow templates (see **Importing SeeBeyond Workflow Templates** on page 90).

Three sample Monk Collaboration Rule script files are included in the installation:

- **siebel-http-outgoing-delete.dsc**
- **siebel-http-outgoing-execute.dsc**
- **siebel-http-outgoing-insert.dsc**

If you are using Siebel 2000 (Japanese), you should use the following alternate files instead:

- **siebel-http-outgoing-delete-sjis.dsc**
- **siebel-http-outgoing-execute-sjis.dsc**
- **siebel-http-outgoing-insert-sjis.dsc**

6.10.1 Connecting to Siebel

When an HTML form is submitted to the Microsoft IIS and the specified action is **http://<webserver>/eai/start.swe**, the IIS loads the Siebel Web Engine (SWE) DLL (**sweiis.dll**). The SWE then obtains the connection string from the **/eai** section of the configuration file **eapps.cfg**. This connection string contains the following information:

- Transport
- Siebel Gateway Server
- Siebel Enterprise Server
- Siebel Object Manager (**eaiObjectManager**)
- Siebel Application Server

Below is an example of a connection string:

```
ConnectString = siebel.TCPIP.none.none://MyGatewayServer:3230/
     MyEnterpriseServer/eaiObjMgr/MyAppServer
```

With this information, the IIS can connect to the Siebel Server utilizing the user name and password given in the form.

6.10.2 Specifying the Business Service

Additional information must be provided to specify the specific method of the business service to be executed. Typically, this information is placed in the configuration file associated with the application. Since the e*Way uses the EAI Object Manager, the appropriate file is **eai.cfg**. This file has two sections that are used by the HTTP adapter, **HTTP Services** and a user-defined method information section. **HTTP Services** is the section in which you define the **SWEExtSource** and the name of the method. The method section allows you to define the adapter mode and the name and method of the Business Service.

Below is an example of how an HTTP Service is specified:

```
[HTTP Services]
ACCOUNT_UPSERT_SERVICE = ACCOUNT_UPSERT_METHOD

[ACCOUNT_UPSERT_METHOD]
Mode = Document
Service = ACCOUNT_UPSERT
Method = RunProcess
```

In this example, the method **RunProcess** of the Business Service **ACCOUNT_UPSERT** is executed if the form has an "input" **SWEExtSource** with the value **"ACCOUNT_UPSERT_SERVICE"**.

An adapter in **Document** mode sends data across a specific data transport without converting the data to a property set. A Business Service of class **CSSWfEngine** is provided, which has a **RunProcess** method to execute a workflow process. The name of the process (i.e., **ProcessName**) needs to be specified in the BIM **BS User Property**.

6.10.3 The Siebel Workflow Process

The Workflow process has the following properties:

- **<Value>** with a type **String**

  This property refers to the **Value** attribute of the property set that is currently active. In the workflow, it can be either the **Inputs** or **Outputs** property set that executes it. In the **Inputs** property set, **Value** contains the incoming XML message; in the **Outputs** property set, **Value** consists of a result string that can be sent back to the Web page.

- Incoming XML with a type **String** and a default value **<Value>**

  Anything you pass along to the URL as data is placed in this variable.

- Message with a type **Hierarchy**

  The message is used to hold the intermediate property set that is generated by the EAI XML Converter.

6.10.4 e*Gate-to-Siebel Example

To insert or update an Employee Record

1 Make a copy of the template **SeeBeyond HTTP Update**.

   A Change the name of the Workflow to **Employee Update Workflow.**

   B Specify the **Business Object** to be **Employee**.

2 The Update Siebel Business Service is hard-coded with the return value <**h1> Update completed.** </h>. You may leave it as it is.

3 In **Siebel Tools**, make a copy of the **Workflow Process Manager** Business Service.

   A Change the value of **Name** to **Employee Update Business Service.**

   B Change the value of **Project** to **EAI**.

4 Add a new **Business Service User Property** named **ProcessName** with the value of **Employee Update Workflow**.

5 Next, add the following sections in the **eai.cfg** file. It should be located in **siebsrvr\bin** directory.

```
[HTTP Services]
...
EUHS = EMPLOYEE_UPDATE_HTTP_SERVICE

[EMPLOYEE_UPDATE_HTTP_SERVICE]
Mode = Document
Service = Employee Update Business Service
Method = RunProcess
```

6 Compile the **.srf** file.

7 In **Siebel Tools**, export the **Employee Integration Object**.

8 Run the **SeeBeyond XML Converter** to generate the **Employee Integration Object ETD**.

9 Assuming that you have defined a Collaboration that satisfies your requirements, you are now ready to modify the Collaboration Rules script.

   A Using the e*Gate Editor, open the sample Collaboration Rules Monk script **siebel-http-outgoing-insert.dsc**.

   B Change the **Integration Object** from **Sample Account** to **Employee.**

   C Change the **HTTP Service** name from **SEEBEYOND_HTTP_UPDATE** to **EUHS**.

   D Since you only want to perform an Insert/Update, delete the **siebel-http-process** call that performs the query operation.

   E You also need to modify the script to match the Collaboration that you defined.

   F Save the modified Collaboration Rules script under a different name.

Note: *See* **Siebel XML Messages** *on page 88 regarding the message format.*

6.10.5 Siebel-to-e*Gate Example

To retrieve an Employee Record and forward it to the e*Gate system

1 Make a copy of the template **SeeBeyond HTTP Send**.

   A Change the name of the Workflow to **Employee Send Workflow.**

   B Specify the **Business Object** to be **Employee**.

2 The Send Business Service is hard-coded with the Request URL Template value **http://<web server>/mux.asp**. You need to specify the **MS IIS** as the web server.

3 Since you are testing the implementation in Siebel Workflow Designer, you need to change the value of **Object Id** of the Process Properties to the value used in your system (in this example, assume that **1-D9T** is the correct ID).

4 In the **MS IIS**:

   A Modify the **Mux.asp** to have the IP address and port number of the Siebel EAI (MUX) e*Way.

   B Since you are not gathering data from a form, set **blnUseBinary** = **true**.

5 In **Siebel Tools**, export the **Employee** Integration Object.

6 Run the **SeeBeyond XML Converter** to generate the **Employee** Integration Object ETD.

7 Create the e*Gate Collaboration to process the ETD.

Note: *See* **Siebel XML Messages** *on page 88 regarding the message format.*

6.11 Sample Schema

Sample implementations are available in the **\samples\ewsiebelhttp\Siebel2000** directory of the e*Gate CD-ROM.

- **SiebelHTTPSample** (e*Gate-to-Siebel schema example)
- **SiebelHTTPSample2** (Siebel-to-e*Gate schema example)

These samples can be used to test your system following installation and, if appropriate, as templates that you can modify to produce your own schema.

See **Optional Example Files** on page 28 for installation instructions.

The sample schema make use of the SeeBeyond Workflow Templates included with the e*Way. You must set up your environment by following the instructions on setting up the templates to execute the Workflow in **SeeBeyond Workflow Templates** on page 83.

6.11.1 e*Gate-to-Siebel Example

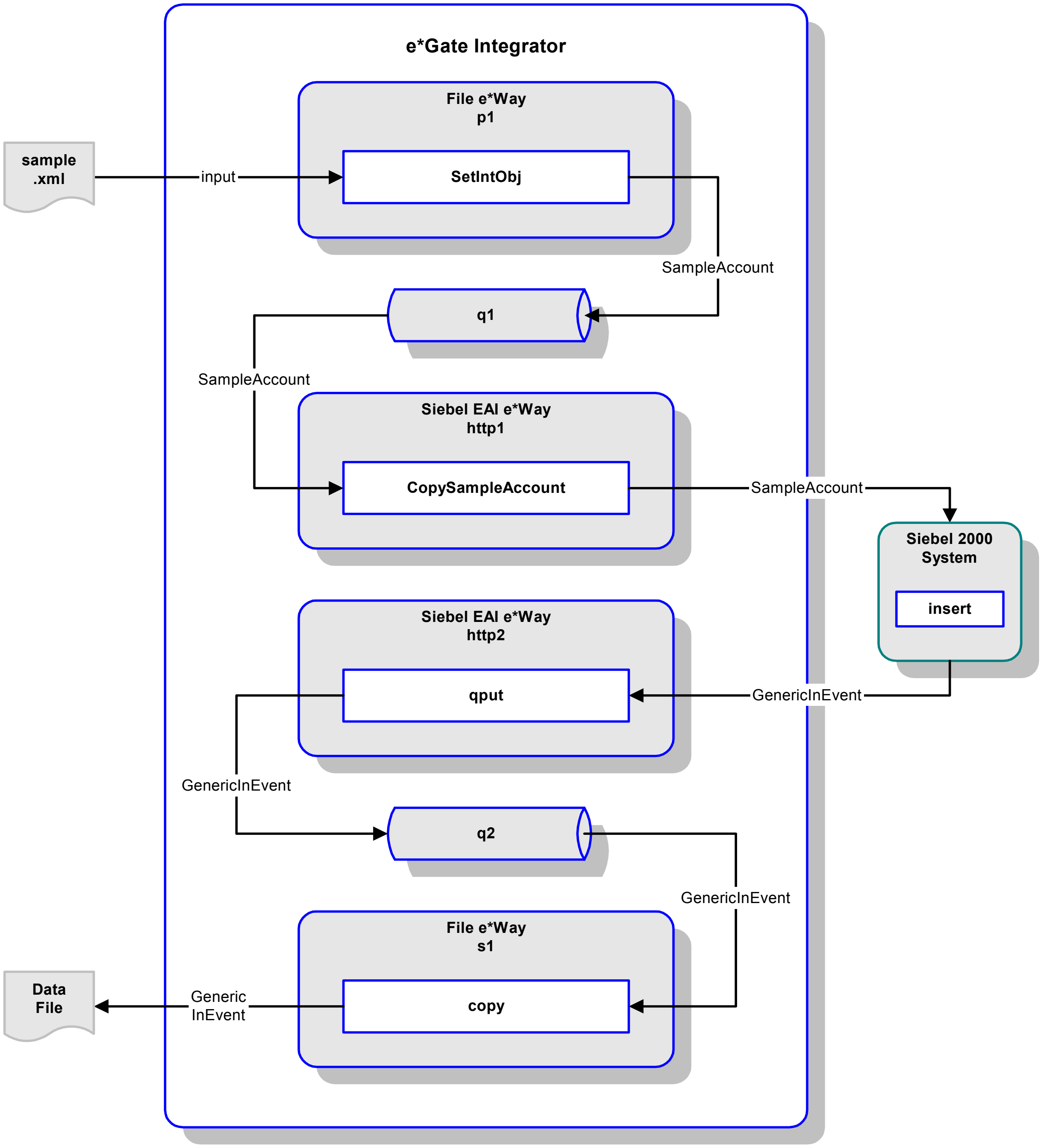
Note: *The default URL is set to* **http://<Web Server>/eai/start.swe**. *You must replace* **<Web Server>** *with a valid web server name or IP address. See* **URL** *on page 46.*

In this example (**SiebelHTTPSample**) the e*Gate system picks up an input file, which has three records. The records are converted to Siebel XML Messages using the Siebel Integration Object ETD. The Siebel EAI e*Way forwards the translated messages via HTTP to the Microsoft IIS which triggers the Siebel Web Engine. The SWE sends it to the Siebel Server, according to the configuration data in the **eapps.cfg** file.

The information in the **eai.cfg** file specifies which Business Service to execute. In turn, the Business Service starts the **SeeBeyond HTTP Update** Workflow process and populates the Siebel database. The Collaboration in the Siebel EAI (HTTP) e*Way also retrieves those Account records and puts them into an IQ. The output File e*Way picks them up from the IQ and writes them to a file.

Note: *Using the SeeBeyond HTTP Execute Workflow template would achieve the same result.*

Figure 50 e*Gate-to-Siebel Sample Schema



Collaborations

The Collaborations used in this sample schema transform data as shown in Figure 51, Figure 52, Figure 54, and Figure 55. Each e*Way instance performs one Collaboration, which may be no more than a pass-through copy process. An additional Collaboration, shown in Figure 53, is invoked by the selected Workflow template.

In the e*Gate-to-Siebel direction, the XML-format **Input** Event is transformed into a **SampleAccount** Event (or message), which is then copied and sent to Siebel without further transformation. See Figure 51 and Figure 52.

Figure 51 Set Internal Object Collaboration

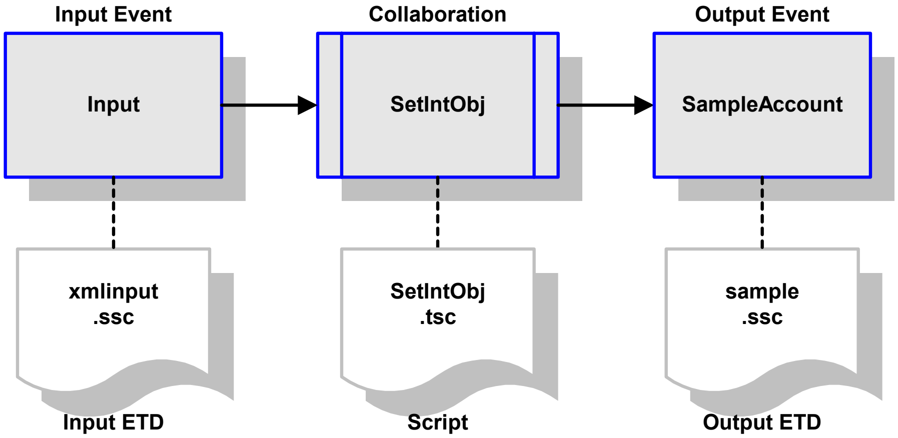
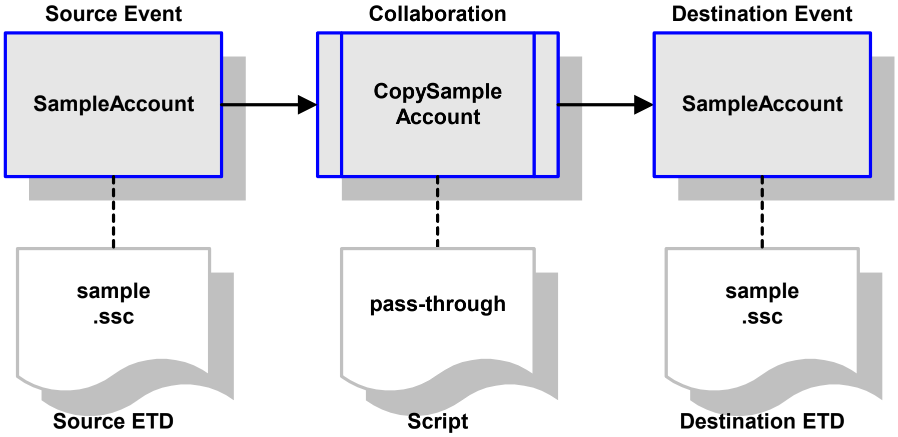


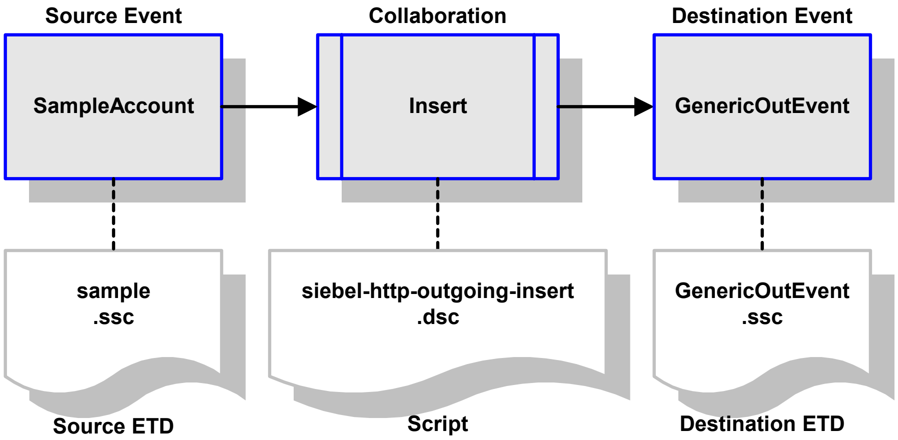
Figure 52 Copy Sample Account Collaboration



At this point, the Workflow takes control and invokes the appropriate database-access Collaboration for loading the data into the Siebel database. In this example, either the **Insert/Update** Workflow or the **Execute** Workflow can be used to achieve the same result. The Collaborations invoked by these Workflow templates are, respectively, **siebel-http-outgoing-insert.dsc** and **siebel-http-outgoing-execute.dsc**. The former is depicted in Figure 53.

Note: *If you are using Siebel 2000 (Japanese), you must rename these database-access Collaboration files to* **siebel-http-outgoing-insert-sjis.dsc** *or* **siebel-http-outgoing-execute-sjis.dsc***, respectively, in the sample schema (***.exp***).*

Figure 53 Insert/Update Collaboration



The sample schema also retrieves the Account records and writes them to a file, by means of the Collaborations depicted in Figure 54 and Figure 55.

Figure 54 qput Collaboration

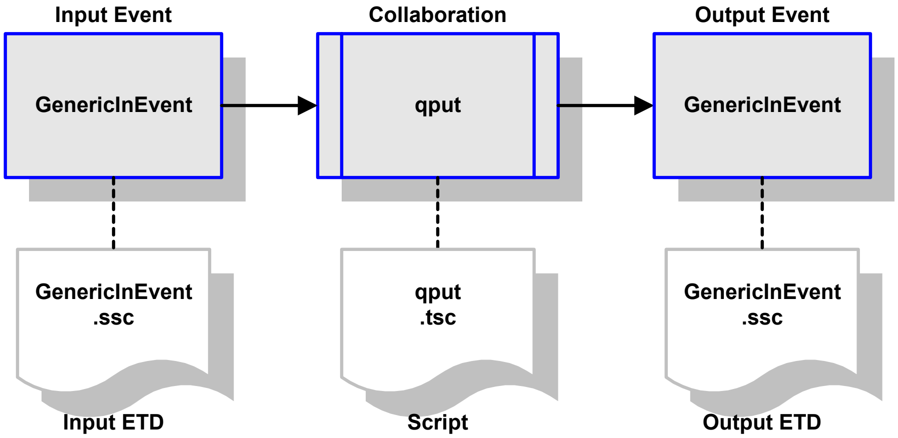
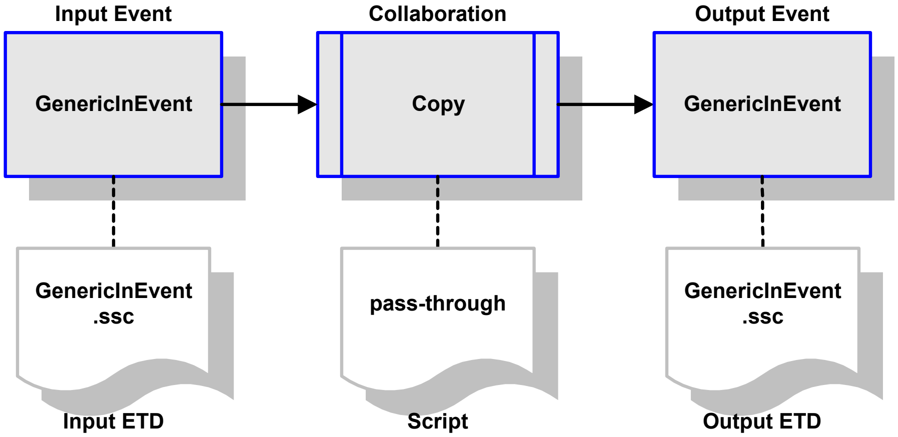


Figure 55 Copy Collaboration

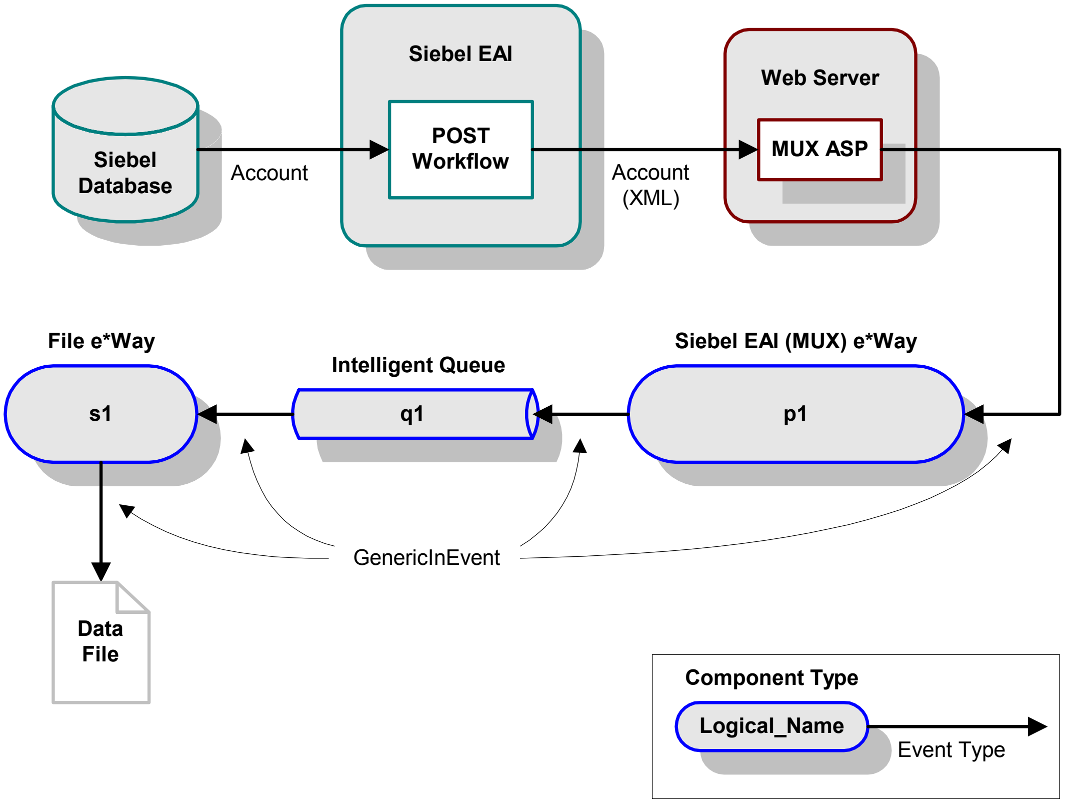


6.11.2 Siebel-to-e*Gate Example

This example (**SiebelHTTPSample2**) uses the **SeeBeyond HTTP Post** Workflow template to forward a Siebel XML message to e*Gate and receive the response from the Siebel EAI (MUX) e*Way. You should modify the workflow as needed to process the return message in your production system.

Selecting the **Start** and **Continue** buttons in Siebel Workflow Designer invokes the **SeeBeyond HTTP Post** Workflow process to retrieve the **Account** object, convert it to a Siebel XML Message, and send it to the Web Server. The MUX ASP then forwards the message to the Siebel EAI (MUX) e*Way. The message is then converted to a file.

Figure 56 Siebel-to-e*Gate Sample Schema



Collaborations

The Collaborations used in this sample schema transform data as shown in the following diagrams. The Siebel EAI (MUX) e*Way receives the Event from the Web server, strips the MUX header using the Collaboration shown in Figure 57. The File e*Way then writes the data to a file using the pass-through Collaboration service shown in Figure 58.

Figure 57 stripMuxHeader Collaboration

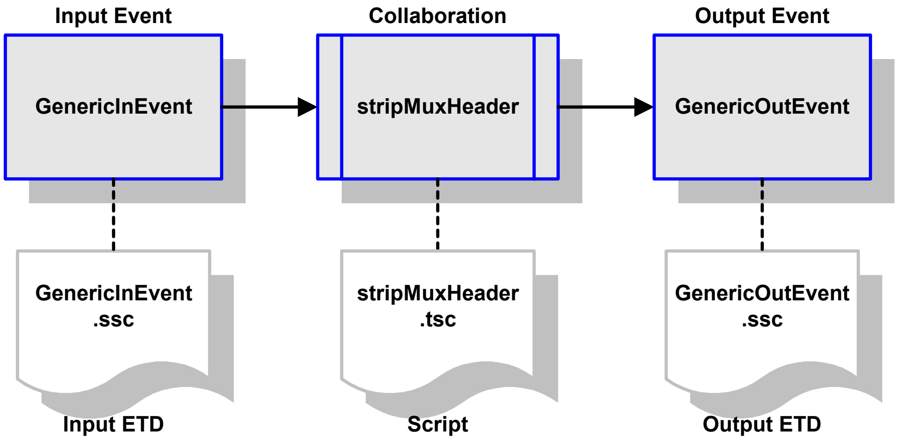
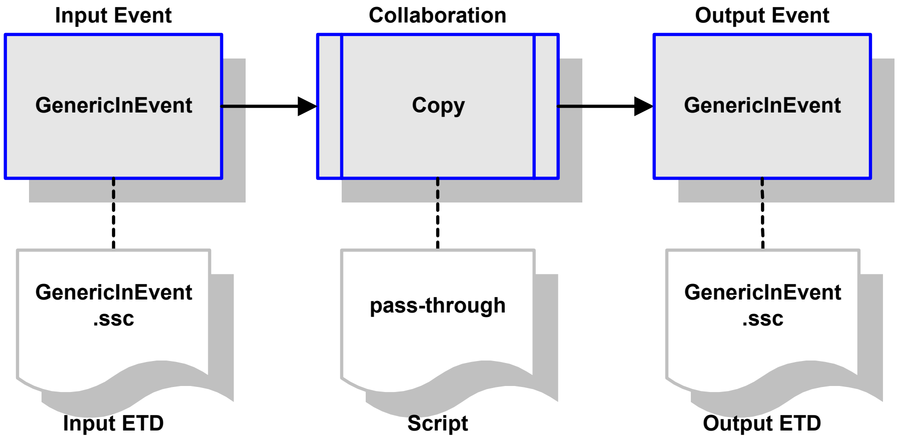


Figure 58 Copy Collaboration



Chapter 7

API Functions

This chapter describes the various Monk functions used by the SeeBeyond e*Way Intelligent Adapter for Siebel EAI.

7.1 Overview

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

- **Siebel HTTP Monk Functions** on page 118
- **Generic e*Way Functions** on page 127
- **e*Gate API Kit Functions** on page 134

7.2 Siebel HTTP Monk Functions

The current suite of Siebel HTTP Monk functions are:

siebel-http-ack on page 118

siebel-http-connect on page 119

siebel-http-exchange on page 119

siebel-http-init on page 120

siebel-http-nack on page 121

siebel-http-notify on page 122

siebel-http-outgoing on page 122

siebel-http-process on page 123

siebel-http-shutdown on page 124

siebel-http-startup on page 125

siebel-http-verify on page 125

siebel-http-ack

Description

Sends a positive acknowledgment to the external system after all Collaborations to which the e*Way sent data have processed and enqueued that data successfully.

Signature

```
(siebel-http-ack message-string)
```

Parameters

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

Returns

A string:

String	Explanation
““ (NULL)	Indicates a successful operation. The e*Way is then able to proceed with the next request.
“CONNERR”	Indicates a problem with the connection to the external system. This function is re-executed when the connection is re-established.

Throws

None.

Location

siebel-http-ack.monk

See also

Positive Acknowledgment Function on page 43.

siebel-http-connect

Description

Establishes a connection to the external system. The connection handle is stored in the Monk variable siebel **hCon**.

Signature

```
(siebel-http-connect)
```

Parameters

None.

Returns

The string **"UP"** indicates the connection is established. Anything else indicates no connection.

Throws

None.

Location

siebel-http-connect.monk

See also

External Connection Establishment Function on page 41.

siebel-http-exchange

Description

Sends a received event from the external system to e*Gate. The function expects no input.

Signature

```
(siebel-http-exchange)
```

Parameters

None.

Returns

A string:

String	Explanation
"" (NULL)	Indicates successful operation. Nothing is sent to e*Gate.
message-string	Indicates successful operation, and the Event is sent to e*Gate.
"CONNERR"	Indicates a problem with the connection to the external system. When the connection is re-established this function is re-executed with the same input Event.

Throws

None.

Location

siebel-http-exchange.monk

Additional Information

This function is provided to show how a normal HTTP e*Way exchanges messages with an external system. If you wish to use it to access Siebel EAI you need to write your own business logic using **siebel-http-process**.

See also

Process Outgoing Message Function on page 39.

siebel-http-init

Description

Begins the initialization process for the e*Way. This function loads the **stc_monkhttp.dll** file and the initialization file, thereby making the function scripts available for future use.

Signature

```
(siebel-http-init)
```

Parameters

None.

Returns

If a **"FAILURE"** string is returned, the e*Way shuts down. Any other return indicates success.

Throws

None.

Location

siebel-http-init.monk

Additional Information

Within this function, any necessary global variables to be used by the function scripts could be defined. The internal function that loads this file is called once when the e*Way first starts up.

See also

Monk Environment Initialization File on page 38.

siebel-http-nack

Description

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

Signature

```
(siebel-http-nack message-string)
```

Parameters

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

Returns

A string:

String	Explanation
"" (NULL)	Indicates a successful operation.
"CONNERR"	Indicates a problem with the connection to the external system. This function is re-executed when the connection is re-established.

Throws

None.

Location

siebel-http-nack.monk

See also

Negative Acknowledgment Function on page 43.

siebel-http-notify

Description

Notifies the external system that the e*Way is shutting down.

Signature

```
(siebel-http-notify command)
```

Parameters

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

Returns

A **null string**.

Throws

None.

Location

siebel-http-notify.monk

See also

Shutdown Command Notification Function on page 44.

siebel-http-outgoing

Description

Used for sending a received message from e*Gate to the external system.

Signature

```
(siebel-http-outgoing event-string)
```

Parameters

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

Returns

A string:

String	Explanation
"" (NULL)	Indicates a successful operation.
"RESEND"	Causes the Event to be resent immediately.
"CONNERR"	Indicates a problem with the connection to the external system. When the connection is re-established this function is re-executed with the same input Event.
"DATAERR"	Indicates the function had a problem processing data. If the e*Gate journal is enabled, the Event is Journaled and the failed Event count is increased. (The input Event is essentially skipped in this process.) Use the **event-send-to-egate** function to place bad events in a "bad event" queue.

Throws

None.

Location

siebel-http-outgoing.monk

Additional Information

This function is provided to demonstrate how a normal HTTP e*Way exchanges messages with an external system. If you wish to use it to access Siebel EAI, you need to write your own business logic incorporating **siebel-http-process**.

Two templates are provided with the e*Way:

- **siebel-http-outgoing-insert**
- **siebel-http-outgoing-delete**

See also

Process Outgoing Message Function on page 39.

siebel-http-process

Description

Forwards the Event to Siebel and specifies the HTTP service that should process it.

Signature

```
(siebel-http-process hCon pszSWEExtSource pszSWEExtCmd pszUserName
    pszPassWord pszContent)
```

Parameters

Name	Type	Description
hCon	Monk object	HTTP connection handle. It holds the connection to the Web server.
pszSWEExtSource	string	The name of the Siebel HTTP service. It should match one of the services specified in **eai.cfg**.
pszSWEExtCmd	string	The command used to execute the HTTP service.
pszUserName	string	The user name used to log onto the Siebel Server.
pszPassWord	string	The password used to log onto Siebel EAI.
pszContent	string	The Siebel XML message that requires processing.

Returns

If the operation has returned data, a string containing the results; otherwise, an empty string.

Throws

None.

Location

siebel-http-process.monk

siebel-http-shutdown

Description

Requests that the external connection shut down. Any return value other than "**SUCCESS**" indicates that the shutdown Event must be delayed. You are then required to execute a **shutdown-request** call from within a Monk function to allow the requested shutdown to continue.

Signature

```
(siebel-http-shutdown shutdown-string)
```

Parameters

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

Returns

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

Throws

None.

Location

siebel-http-shutdown.monk

See also

External Connection Shutdown Function on page 42.

siebel-http-startup

Description

Used for e*Way-specific function loads; also invokes startup.

Signature

```
(siebel-http-startup)
```

Parameters

None.

Returns

The string **"FAILURE"** causes the e*Way to shut down. Anything else indicates success.

Throws

None.

Location

siebel-http-startup.monk

Additional Information

This function should be used to initialize the external system before data exchange starts. Any additional variables may be defined here.

See also

Startup Function on page 39.

siebel-http-verify

Description

Used to verify that the connection to the external system is established.

Signature

```
(siebel-http-verify)
```

Parameters

None.

Returns

The string **"UP"** or **"SUCCESS"** indicates connection established. Any other value indicates the connection was not established.

Throws

None.

Location

siebel-http-verify.monk

See also

External Connection Verification Function on page 42.

7.3 Generic e*Way Functions

The functions described in this section are implemented in the e*Way Kernel layer and control the e*Way's most basic operations. They can be used only 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. These functions are located in **stcewgenericmonk.exe**.

The current set of basic Monk functions is:

event-commit-to-egate on page 127

event-rollback-to-egate on page 128

event-send-to-egate on page 128

event-send-to-egate-ignore-shutdown on page 129

event-send-to-egate-no-commit on page 129

get-logical-name on page 130

insert-exchange-data-event on page 130

send-external-up on page 131

send-external-down on page 131

shutdown-request on page 132

start-schedule on page 132

stop-schedule on page 133

waiting-to-shutdown on page 133

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 129

event-send-to-egate-no-commit on page 129

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 128

event-send-to-egate-no-commit on page 129

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 127

event-rollback-to-egate on page 128

event-send-to-egate on page 128

event-send-to-egate-ignore-shutdown on page 129

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 34

Zero Wait Between Successful Exchanges on page 34

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.

7.4 e*Gate API Kit Functions

The Siebel EAI (MUX) e*Way only makes use of selected elements of the e*Gate API Kit: however, the API Kit executable (**stcewipmp.exe**) contains several sets of functions that are accessible for your use. For information regarding these functions, please see the *e*Gate API Kit User's Guide*.

Index

A

Accept-type parameter **48**
Additional Path parameter **37**
Agent parameter **47**
APIs - see Monk functions
Application Object Manager **12**
Assigning ETDs to Event Types **103**
Autorun **20**
Auxiliary Library Directories parameter **38**

B

Business Integration Manager (BIM) **12**
Business Service **12**

C

Client Certificate Map File parameter **51**
Collaboration **67**, **71**, **104**
  Rules **71**, **104**
Collaboration Rules script **68**
Collaborations **109**
components, e*Way **16**
configuration
  Communication Setup **34–36**
  General Settings (HTTP) **32–33**
  General Settings (MUX) **54–55**
  HTTP Configuration **46–48**
  HTTP Proxy Configuration **49–50**
  HTTP SSL Configuration **51–52**
  Monk Configuration **37–45**
  Siebel Configuration **53**
configuration files, e*Way
  eai.cfg **24**
  eapps.cfg **24**
  stcewipmp.def **16**
  stcewsiebelhttp.def **16**
configuration parameters
  Accept-type **48**
  Additional Path **37**
  Agent **47**
  Auxiliary Library Directories **38**
  Client Certificate Map File **51**
  Content-type **47**
  Down Timeout **36**
  Encrypted Password (HTTP Configuration) **47**
  Encrypted Password (HTTP Proxy Configuration) **49**
  Encrypted Password (Siebel Configuration) **53**
  Exchange Data Interval **34**
  Exchange Data with External Function **40**
  External Connection Establishment Function **41**
  External Connection Shutdown Function **42**
  External Connection Verification Function **42**
  Forward External Errors **33**
  Journal File Name **32**
  Max Failed Events **32**
  Max Resends per Message **32**
  Monk Environment Initialization File **38**
  Negative Acknowledgment Function **43**
  Port Number **50**
  Positive Acknowledgment Function **43**
  Process Outgoing Message Function **39**
  Push IP Port **54**
  Request **46**
  Request Reply IP Port **54**
  Request-content **47**
  Resend Timeout **36**
  Rollback if No Clients on Push Port **54**
  Server Address **49**
  Shutdown Command Notification Function **44**
  Start Exchange Data Schedule **35**
  Startup Function **39**
  Stop Exchange Data Schedule **35**
  SWEExtCmd **53**
  SWEExtSource **53**
  Timeout **46**
  Trusted CA Certificated Directory **51**
  Up Timeout **36**
  URL **46**
  Use Client Certificate Map **51**
  Use Proxy Server **49**
  User Name (HTTP Proxy) **49**
  User Name (HTTP) **47**
  User Name (Siebel) **53**
  Wait for IQ Ack **54**
  Zero Wait Between Successful Exchanges **34**
configuration procedures **59**
Content-type parameter **47**
conventions, writing **9**

D

DELETE Workflow Template **84**
DLL files
  stc_ewipmpclnt.dll **16**
  stc_ewipmpclntperl.dll **16**
  stc_monkhttp.dll **16**

stc_xipmpclnt.dll **16**
sweiis.dll **26**
Down Timeout parameter **36**

E

e*Gate API Kit Functions **134**
e*Way
Components **16**
configuration procedures **59**
creating **57**
Installation **20**
Properties **58**
Schedules **63**
Startup Options **63**
troubleshooting **67**
User name **63**
EAI Siebel Adapter **12, 14**
EAI XML Converter **12, 14**
eai.cfg file **24**
eapps.cfg file **24**
Encrypted Password (HTTP Configuration) parameter **47**
Encrypted Password (HTTP Proxy Configuration) parameter **49**
Encrypted Password (Siebel Configuration) parameter **53**
Event Type **103**
Event Type Definition (ETD) **101, 103**
event-commit-to-egate function **127**
event-rollback-to-egate function **128**
Events **70**
event-send-to-egate function **128**
event-send-to-egate-ignore-shutdown function **129**
event-send-to-egate-no-commit function **129**
Exchange Data Interval parameter **34**
Exchange Data with External Function parameter **40**
executable files
stc_form2ssc.exe **16**
stcewgenericmonk.exe **16**
stcewipmp.exe **16**
stcewmuxipserver.exe **16**
stcxipmptest.exe **16**
EXECUTE Workflow Template **86**
External Connection Establishment Function parameter **41**
External Connection Shutdown Function parameter **42**
External Connection Verification Function parameter **42**

F

Forward External Errors parameter **33**
functions (see also Monk functions)
e*Gate API Kit **134**
Generic **127–133**
Siebel HTTP **118–126**

G

Generic e*Way Functions **127–133**
get-logical-name function **130**

I

INSERT/UPDATE Workflow Template **85**
insert-exchange-data-event function **130**
Installation procedure
sample schema **29**
InstallShield **20**
Intelligent Queue (IQ) **105**
Internet Information Service (IIS) **12, 14**

J

Journal File Name parameter **32**

L

library files
stdole2.tlb **19**
stdole32.tlb **19**
Load Path, Monk **37**
logging options **65**

M

Max Failed Events parameter **32**
Max Resends Per Message parameter **32**
Microsoft Internet Information Service (IIS) **12, 14**
monitoring thresholds **66**
Monk Configuration
Load Path **37**
Specifying File Names **37**
Specifying Function Names **37**
Specifying Multiple Directories **37**
Monk Environment Initialization File **38**
Monk functions
event-commit-to-egate **127**
event-rollback-to-egate **128**
event-send-to-egate **128**
event-send-to-egate-ignore-shutdown **129**
event-send-to-egate-no-commit **129**
get-logical-name **130**
insert-exchange-data-event **130**
send-external down **131**

send-external-up **131**
shutdown-request **132**
siebel-http-ack **118**
siebel-http-connect **119**
siebel-http-exchange **119**
siebel-http-init **120**
siebel-http-nack **121**
siebel-http-notify **122**
siebel-http-outgoing **122**
siebel-http-process **123**
siebel-http-shutdown **124**
siebel-http-startup **125**
siebel-http-verify **125**
start-schedule **132**
stop-schedule **133**
waiting-to-shutdown **133**
MUX ASP **14, 27**

N

Negative Acknowledgment Function parameter **43**

P

parameters - see configuration parameters
Port Number parameter **50**
Positive Acknowledgment Function parameter **43**
POST Workflow Template **87**
procedures
configuration **59**
installation **20**
setup **57**
Process Outgoing Message Function parameter **39**
Properties, e*Way **58**
publish **67**
Push IP Port parameter **54**

Q

QUERY Workflow Template **85**
Queues **105**

R

Request parameter **46**
Request Reply IP Port parameter **54**
Request-content parameter **47**
Resend Timeout parameter **36**
Rollback if No Clients on Push Port parameter **54**

S

sample schema
descriptions **110**
installation **29**
Schedules **63**
SEND Workflow Template **86**
SEND/RECEIVE Workflow Template **87**
send-external down function **131**
send-external-up function **131**
Server Address parameter **49**
Setting Startup Options or Schedules **63**
Shutdown Command Notification Function parameter **44**
shutdown-request function **132**
Siebel EAI Adapter **13**
Siebel EAI Toolkit **12**
Siebel HTTP Monk Functions **118–126**
Siebel Integration Objects **12**
Siebel Object Manager **13**
Siebel Web Engine (SWE) **12**
Siebel Web Server Extension (SWSE) **12, 13, 24, 26**
siebel-http-ack function (Monk) **118**
siebel-http-connect function (Monk) **119**
siebel-http-exchange function (Monk) **119**
siebel-http-init function (Monk) **120**
siebel-http-nack function (Monk) **121**
siebel-http-notify function (Monk) **122**
siebel-http-outgoing function (Monk) **122**
siebel-http-process function (Monk) **123**
siebel-http-shutdown function (Monk) **124**
siebel-http-startup function (Monk) **125**
siebel-http-verify function (Monk) **125**
Start Exchange Data Schedule parameter **35**
start-schedule function **132**
Startup Function parameter **39**
Startup Options **63**
stc_ewipmpclnt.dll file **16**
stc_ewipmpclntperl.dll file **16**
stc_form2ssc.exe file **16**
stc_monkhttp.dll file **16**
stc_xipmpclnt.dll file **16**
stcewgenericmonk.exe file **16**
stcewipmp.def file **16**
stcewipmp.exe file **16**
stcewmuxipserver.exe file **16**
stcewsiebelhttp.def file **16**
stcxipmptest.exe file **16**
stdole2.tlb file **19**
stdole32.tlb file **19**
Stop Exchange Data Schedule parameter **35**
stop-schedule function **133**
subscribe **67**
SWEExtCmd parameter **53**
SWEExtSource parameter **53**
sweiis.dll file **26**

T

Timeout parameter **46**
Transport Adapters **12**
troubleshooting the e*Way **67**
Trusted CA Certificated Directory parameter **51**

U

Up Timeout parameter **36**
URL parameter **46**
Use Client Certificate Map parameter **51**
Use Proxy Server parameter **49**
User Name (HTTP Proxy)
  parameter **49**
User Name (HTTP)
  parameter **47**
User Name (Siebel)
  parameter **53**
User name, e*Way **63**

W

Wait for IQ Ack parameter **54**
waiting-to-shutdown function **133**
Web server
  Microsoft Internet Information Service (IIS) **12**, **14**
Workflow Process Designer **12**
Workflow Templates
  DELETE **84**
  EXECUTE **86**
  INSERT/UPDATE **85**
  POST **87**
  QUERY **85**
  SEND **86**
  SEND/RECEIVE **87**
writing conventions **9**

X

XML Converter **13**

Z

Zero Wait Between Successful Exchanges parameter **34**