SeeBeyond eBusiness Integration Suite Deployment Guide

Release 4.5.3

SEEBEYOND™

Contents

Chapter 6

Testing, Transition to Production, and Maintenance 161

Chapter 7

Frequently Asked Questions 201

Chapter 8

Deploying for High Availability 210

List of Figures

List of Tables

Introduction

Welcome to the *eBusiness Integration Suite Deployment Guide*. This chapter provides an overview of the purpose and contents of this guide. You will also find references for related documentation and the writing conventions used in this guide.

In This Chapter

- **About This Deployment Guide** on page 17
- **Contents of This Guide** on page 18
- **Writing Conventions** on page 18
- **Supporting Documents** on page 20
- **SeeBeyond Web Site** on page 21

1.1 About This Deployment Guide

The Deployment Guide provides deployment planning guidelines and deployment strategies for the SeeBeyond™ eBusiness Integration (eBI™) Suite. This guide is designed for management, system administrators, and others who are tasked with deployment of the eBI Suite. The purpose of this guide is to help you successfully complete the following stages of deployment:

- Analyzing the requirements
- Planning the deployment
- Determining system requirements
- Designing and developing the eBI Suite environment
- Testing the eBI Suite
- Transition to production
- Maintaining the eBI Suite environment

Note: *This guide does not discuss deployment of e*Index™ Global Identifier. For more information about this product, refer to the e*Index documentation.*

1.2 Contents of This Guide

This document includes the following information:

- **Chapter 1**, **"Introduction"** provides an overview of this document's purpose, contents, writing conventions, and supported documents.

- **Chapter 2**, **"Overview of the eBI Suite"** discusses the general features and architecture of the eBI Suite.

- **Chapter 3**, **"Analysis and Planning"** explains how to analyze your current business processes and IS setup in order to plan your eBI Suite deployment.

- **Chapter 4**, **"Determining System Requirements"** helps you gather relevant information and make decisions to determine what the type of hardware required to support your eBI Suite environment.

- **Chapter 5**, **"Designing and Developing the eBI Suite Environment"** explains how to design and develop and create an eBI Suite environment to best meet your overall business and IS needs. It also contains valuable system optimization information.

- **Chapter 6**, **"Testing, Transition to Production, and Maintenance"** tells you what to do during the final phases of your eBI Suite deployment, including pre-transition testing, the transition to production, and post-transition maintenance.

- **Chapter 7**, **"Frequently Asked Questions"** provides a list of FAQs with answers to common questions to provide helpful hints, best practices, and information about obtaining the best eBI Suite performance. It is recommended that you read this chapter before starting deployment.

- **Chapter 8**, **"Deploying for High Availability"** explains ways you can design your e*Gate system for high availability in case of failure.

- **Appendix A**, **"Deployment Surveys"** contains a *Deployment Survey Questionnaire* you can print and photocopy for your own use.

- **Appendix B**, **"Sample QA Report"** has a sample quality-assurance report on optimizing a sample system.

- **Appendix C** ,**"Upgrading from DataGate to e*Gate"** provides some basic guidelines on upgrading from e*Gate 3.6 to e*Gate 4.5.

- **Appendix D**, **"Installing e*Gate on Windows 2000 Clusters"** explains how to configure e*Gate for Windows 2000 with Microsoft clustering software.

This guide also includes a **Glossary** on page 249. The glossary provides definitions of the eBI Suite terminology.

1.3 Writing Conventions

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

Hypertext Links

When you are using this guide online, cross-references are also hypertext links and appear in **blue text** as shown below. Click the **blue text** to jump to the section.

> For information on these and related topics, see **"Parameter, Function, and Command Names" on page 20**.

Command Line

Text to be typed at the command line is displayed in a special font as shown below.

```
java -jar ValidationBuilder.jar
```

Variables within a command line are set in the same font and bold italic as shown below.

```
stcregutil -rh host-name -rs schema-name -un user-name
-up password -ef output-directory
```

Code and Samples

Computer code and samples (including printouts) on a separate line or lines are set in Courier as shown below.

```
Configuration for BOB_Promotion
```

However, when these elements (or portions of them) or variables representing several possible elements appear within ordinary text, they are set in *italics* as shown below.

> *path* and *file-name* are the path and file name specified as arguments to **-fr** in the **stcregutil** command line.

Notes and Cautions

Points of particular interest or significance to the reader are introduced with *Note*, *Caution*, or *Important*, and the text is displayed in *italics*, for example:

Note: The Actions menu is only available when a Properties window is displayed.

User Input

The names of items in the user interface such as icons or buttons that you click or select appear in **bold** as shown below.

> Click **Apply** to save, or **OK** to save and close.

File Names and Paths

When names of files are given in the text, they appear in **bold** as shown below.

> Use a text editor to open the **ValidationBuilder.properties** file.

When file paths and drive designations are used, with or without the file name, they appear in **bold** as shown below.

> In the **Open** field, type **D:\setup\setup.exe** where **D:** is your CD-ROM drive.

Parameter, Function, and Command Names

When names of parameters, functions, and commands are given in the body of the text, they appear in **bold** as follows:

The default parameter **localhost** is normally only used for testing.

The Monk function **iq-put** places an Event into an IQ.

You can use the **stccb** utility to start the Control Broker.

Additional Conventions

This guide uses the term "Windows" to refer to Windows NT, Windows 2000, and Windows XP.

1.4 Supporting Documents

For more information about the eBI Suite, refer to the following documents:

- *Creating an End-to-end Scenario with e*Gate Integrator*
- *e*Gate Integrator Alert Agent User's Guide*
- *e*Gate Integrator Alert and Log File Reference Guide*
- *e*Gate Integrator Collaboration Services Reference Guide*
- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator Intelligent Queue Services Reference Guide*
- *e*Gate Integrator SNMP Agent User's Guide*
- *e*Gate Integrator System Administration and Operations Guide*
- *e*Gate Integrator Upgrade Guide*
- *e*Gate Integrator User's Guide*
- *e*Insight Business Process Manager Implementation Guide*
- *e*Insight Business Process Manager User's Guide*
- *e*Xchange Partner Manager Implementation Guide*
- *e*Xchange Partner Manager User's Guide*
- *e*Xpressway Integrator OnRamp Setup Guide for Trading Partners*
- *e*Xpressway Integrator Server Setup and Maintenance Guide for Trading Exchanges*
- *e*Xpressway Integrator OnRamp Customization Guide for Solution Providers*
- *Monk Developer's Reference*
- *SeeBeyond eBusiness Integration Suite Primer*
- *SeeBeyond JMS Intelligent Queue User's Guide*
- *Standard e*WayTM Intelligent Adapters User's Guide*

- *XML Toolkit*

See the *SeeBeyond eBusiness Integration Suite Primer* for a complete list of eBI Suite-related documentation. You can also refer to the appropriate Microsoft Windows, UNIX, or Linux documents, if necessary.

Note: *For information about using a specific add-on, such as an e*Way Intelligent Adapter or Intelligent Queue (IQ™), refer to the user's guide for that add-on.*

1.5   SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The Web site's URL is **http://www.seebeyond.com**.

Overview of the eBI Suite

This chapter gives a general overview of the eBI Suite, including system descriptions, general operation, and basic features.

In This Chapter

- **"About e*Gate" on page 23**
- **"About e*Insight" on page 32**
- **"About e*Xchange" on page 34**
- **"About e*Xpressway" on page 35**
- **"Deployment: Getting Started" on page 36**

2.1 Introduction: The eBI Suite

The eBI Suite is a group of products that merges traditional Enterprise Application Integration (EAI) and business-to-business (B2B) interactions into multi-enterprise eBusiness Integration (eBI). eBI makes eBusiness attainable for organizations with complex and dynamic partner relationships, and provides the ability to create and manage virtual organizations across the entire supply chain.

e*Gate Integrator (e*Gate)

e*Gate is a robust eBusiness Integration platform that solves complex connectivity issues and enables the dynamic, guaranteed delivery of information across applications and systems, to partners and customers with unmatched performance, flexibility, and speed of implementation.

e*Insight Business Process Manager (e*Insight)

e*Insight facilitates the automation and administration of business process flow across eBusiness activities. Through graphical modeling and monitoring, business analysts can quickly assess the detailed state of a business process instance and identify bottlenecks in the process.

e*Xchange Partner Manager (e*Xchange)

Conducting eBusiness over the public domain requires secure transmissions and utilization of standard business process protocols. The e*Xchange application provides features to ensure full security and non-repudiation of data. This system supports numerous messaging protocols, all of which are accepted industry standards.

e*Xpressway Integrator (e*Xpressway)

e*Xpressway enables rapid trading-partner connectivity and integration through a comprehensive B2B implementation methodology, graphical configuration wizards, and downloadable partner connectivity software.

This chapter contains basic background information about the eBI Suite you need to know and review before you begin your deployment project.

2.2   About e*Gate

The basic purpose of e*Gate is to automate the exchange of information between systems, within its own data Events. e*Gate is based on a distributed and open architecture, allowing components to reside on different workstations within a global network. Based on which communication protocols and adapters you choose, e*Gate can communicate with and link multiple applications and databases across a variety of operating systems.

e*Gate performs effectively with a wide variety of hardware, Event standards, operating systems, databases, and communication protocols in both real-time and batch and scheduled integration modes. e*Gate bridges legacy and new systems, resulting in a centrally managed, intelligent, unified enterprise. This architecture gives network administrators the flexibility to incorporate best of breed technology into their business strategy, without any need to uproot legacy IS investments.

e*Gate system components are organized into schemas. A schema is a configuration scheme that contains all the modules and configuration parameters that control, route, and transform data as it travels through the e*Gate system. Schemas also maintain the relationships between the components, including the publish and subscribe (pub/sub) information that is at the heart of the data transportation process.

e*Gate delivers a high level of performance and flexibility in the control of cross-application business processes.

2.2.1 Architectural Overview

The e*Gate system implements a "transparent" architecture, which is well-suited for distributed network architecture. This means the different components of an e*Gate environment do not have to all reside on the same system; instead they can be distributed across several different systems in the network.

Principal features of this type of architecture include:

- High scalability
- Parallelism
- High availability
- Protection through isolation
- Extensibility

- Avoidance of data processing bottlenecks and single points of failure

e*Gate Layers

The e*Gate architecture consists of the following primary layers:

- View

- Control

- Business rules and data processing

- Intelligent queuing

- Application connectivity

Figure 1 shows each of these layers.

Figure 1 e*Gate Architecture
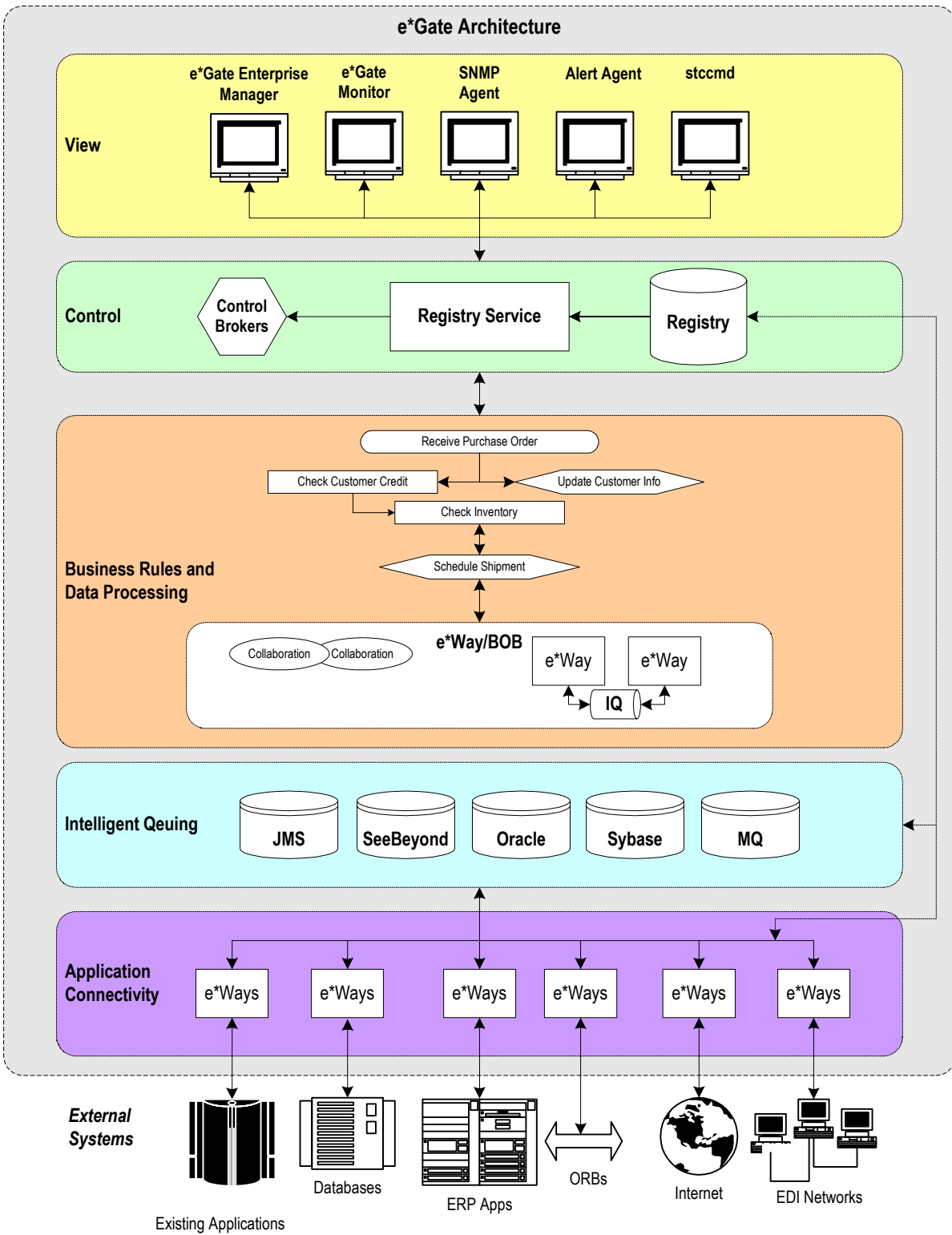
2.2.2 View Layer

The View layer contains those components users interact with, such as user interfaces. Most View layer components implement graphical user interfaces (GUIs), including Java programming language-based features, to simplify their use. Specifically, these components are:

- e*Gate Enterprise Manager
- SeeBeyond Java Collaboration Rules Editor
- SeeBeyond Java Event Type Definition (ETD) Editor
- SeeBeyond Monk Collaboration Rules Editor
- SeeBeyond Monk ETD Editor
- e*Way Editor
- e*Gate Monitor
- e*Gate Alert Agent configuration tool
- e*Gate SNMP Agent
- Command-line interface monitoring tool **stccmd**

Enterprise Manager

The Enterprise Manager allows users to create and configure the components of the e*Gate system. In the Enterprise Manager, users define and maintain configuration schemes (schemas) that contain the parameters of all the components that control, route, and transform data as it travels through the system.

There are GUI features in the Enterprise Manager, which allow easy access to all its operations. A Navigator feature organizes all system components into a tree structure, similar to the one in Windows Explorer. An Editor feature displays additional details about the selected component or the subsidiary components that the selected element controls.

The Navigator feature allows the following views of your e*Gate system:

- **Component** provides views of all elements in the schema (that is, system configuration) based on logical components.
- **Network** provides a visualization of the e*Gate system based on physical resources.

*Note: See the **e*Gate Integrator User's Guide** for illustrations of the GUI features discussed in this section, as well as information on how to use them.*

e*Gate Editors

e*Gate systems implement several distinct, specialized editors to allow users to view and revise each element in the system. e*Gate provides the following editors:

- Collaboration Rules Editors (Java and Monk)
- ETD Editors (Java and Monk)

- e*Way Editor

The e*Gate editors allow users to define the topology of the data processing environment and the relationships among various data processing components. They specify static routes between stages of data processing, when the user knows at configuration time which components feed into others. They also define pub/sub relationships. The remainder of this section provides details on each of the editors.

Collaboration Rules Editors

Available in Java and Monk versions, the Collaboration Rules Editors define the low-level rules that make up Collaboration Rules. These rules specify how application programming interfaces (APIs) are invoked, how data is transformed or operated on, what information is placed in headers, and so on.

The rules can be as general (*n* to *m* apps) or as specific (1 to 1) as required. Users can use ETDs provided by SeeBeyond, ETDs generated through e*Gate system tools, or their own creations built through an ETD Editor when defining Collaboration Rules input-output logic.

ETD Editors

Also available in Java and Monk versions, the ETD Editors set up data-processing logic. ETDs visually represent the hierarchy of data fields as nodes in a tree structure. e*Gate systems use them at run time to parse buffers into data units that the rules can be applied to. Libraries of prebuilt structures already exist for common message/Event format standards including X12, HL7, SWIFT, cXML, and UN/EDIFACT.

Additionally, SeeBeyond provides tools for automatically generating these structures from database tables, stored procedure definitions, proprietary application programming interfaces (APIs), and electronic specifications distributed by standard bodies. For example, an XML Builder tool is available for generating e*Gate ETDs from XML DTDs and schemas, in Java and in Monk.

e*Way Editor

The e*Way Editor configures the communication parameters necessary for establishing and maintaining e*Ways. These e*Gate components enable connectivity with systems and applications external to the e*Gate system. Users see different parameters in the editor window, depending on the external system type. Users can edit the values associated with these parameters or accept provided defaults.

e*Gate Monitor

The e*Gate Monitor provides all basic monitoring and control functions for e*Gate systems. This feature displays status and alert Events for all components across the enterprise and provides high-level control of data processes (for example, start, suspend, and shutdown).

e*Gate Alert Agent

The e*Gate Integrator Alert Agent provides advanced routing services, delivering alert Events via page, e-mail, print, fax, or telephone (synthesized speech). This feature alleviates the need for a dedicated resource to be continuously located near a GUI

display. The Alert Agent has one primary function, to collect error and informational Events from monitored systems and forward them to a repository.

e*Gate SNMP Agent

The e*Gate Integrator SNMP Agent is available for users who require the ability to forward status and alert Events to a central, third party, SNMP-compliant monitor. The SNMP Agent forwards Monitoring Events to external monitoring systems. The SNMP Agent is compatible with systems which support SNMP protocol version 1.

Command Line's stccmd

e*Gate systems include **stccmd**, a command-line API that assists the user in administering and monitoring e*Gate systems in situations where low-bandwidth remote administration is necessary.

2.2.3 Control Layer

The Control layer is responsible for storing and dynamically distributing all system configuration information, starting up and shutting down e*Gate processes, enforcing various access control mechanisms, and the selective forwarding of metadata. Control layer components include:

- Registry, including the Registry Service
- Control Brokers

APIs that are exposed at the Control layer are necessary for broad extensibility in e*Gate systems. APIs permit access to run-time processes, including user, IQs, and message/Event queues. APIs also allow the Registry to be implemented as either a relational database or a flat-file.

Registry

The e*Gate Registry is the store for all configuration details, either through references to supplemental configuration files or direct containment. The Registry Service handles all requests for configuration updates, changing the content of the Registry when new information is provided and forwarding these updates to the appropriate clients as necessary. The Registry Service is multi-threaded and handles all configuration updating and distribution.

The Registry employs a "Team Registry" concept. The Registry itself is divided into Sandbox areas for user-specific file development and an area for the run-time environment. This subdivision separates work-in-progress files from those used in the production schema.

Control Brokers

The Control Broker in a schema is responsible for starting and stopping processes, and selectively forwarding metadata (alert, status, and configuration Events) to authorized

user interfaces. Control Brokers route operational Events to scripts, invoking control APIs to perform basic maintenance or administrative actions.

e*Gate systems support over 80 system-standard Events, such as the detection of disk-space usage beyond a configurable limit, detecting data content of interest, Event creation volume, Event receipt volume, numerous IQ operations, and so on. e*Ways add many application-specific Events to this set and implement user-defined Events.

2.2.4 Business Rules and Data Processing Layer

The Business Rules and Data Processing layer uses Collaborations, Collaboration Rules, Business Object Brokers (BOBs), and e*Ways to implement user-defined business logic in response to input Events.

The business rules and data processing layer contains instructions specifying the details of how applications work together. These instructions are referred to as Collaboration Rules. Examples of Collaboration Rules include data identification and transformation rules, invoking application Events through APIs.

Collaboration Rules are defined with the SeeBeyond Collaboration Rules Editors, which allow the point-and-click/drag-and-drop definition of business rules. The Collaboration Rules are processed by e*Gate and are designed to handle any data format with the highest level of functional abstraction, reducing the amount of user configuration required to a minimum.

Collaborations

Collaborations use Collaboration Rules, which allow users to define how data is to be mapped from n input Events to m output Events. They also define how databases are queried in response to request Events and how APIs to one or more applications are invoked for coordinated action, and so on.

The same Collaborations can be simultaneously used in multiple BOBs or e*Ways, providing business process services that can be tapped into by any number of distributed components. Conversely, multiple Collaborations can be loaded into a single e*Way or BOB, as both components are multi-threaded.

For more information, see the *e*Gate Integrator Collaboration Services References Guide*.

Unicode Support

The e*Gate product suite supports Unicode, Extended UNIX Code (EUC) and Shift-Japanese Industrial Standard (SJIS) multi-byte encoding methods, which allows the handling of Kanji data.

2.2.5 Intelligent Queuing Layer

e*Gate IQ Services act as powerful facilitators of reliable interprocess communication. The intelligence arises from the persistent recording of Event-state information. This state information is necessary to ensure that subscribers acquire the Events they expect,

in the proper sequence and without risk of duplication, even when recovering from hardware failure.

APIs exposed at the Intelligent Queuing layer allow users to implement IQs in a medium-independent manner. e*Gate IQs interface with libraries written for various data-storage implementations, including SeeBeyond's default (a B+ tree design), Sybase-CTLib, Oracle-OCI, OracleAQ (available in a future add-on), and IBM-MQSeries. These APIs also support any custom IQ inspection and interaction logic the user develops.

IQs

IQs provide a smart store-and-forward mechanism for e*Gate Events by supporting pub/sub processing and providing a form of interprocess communication for more reliable delivery of Events. Publisher components populate IQs with Events. IQs have control of when Events are made available to subscribers.

IQ Services

The standard e*Gate installation includes SeeBeyond IQs for all-purpose data handling, as well as specialized Services, for example Sybase and Oracle, that handle data related to specific applications. For more information, see the *e*Gate Integrator Intelligent Queue Services Reference Guide.*

IQ Administrator

e*Gate provides a specialized GUI utility, available through the e*Gate Monitor, which allows you to check the status of an IQ or observe the journaled Event properties of an IQ. For more information, see the *e*Gate Integrator User's Guide.*

SeeBeyond JMS IQ

In e*Gate, you can implement the Java Message Service (JMS) using IQ Managers, IQs, and a special e*Way connection to SeeBeyond's standard Multi-Mode e*Way. For more information, see the *SeeBeyond JMS Intelligent Queue User's Guide.*

2.2.6 Application Connectivity Layer

The Application Connectivity layer consists of one or more e*Ways, components that connect e*Gate and business applications, and optional BOBs, components that implement high-performance, distributed, and complex processing of data flow.

e*Way Component Operation

e*Ways connect business applications with the e*Gate system, communicating with both external applications and IQs. When integrating different systems, the appropriate e*Way on each end of the route provides the adaptation necessary for seamless Event flow, allowing the integration of applications and databases without changing them.

e*Ways establish connectivity with business applications, using whatever communication protocol is appropriate. Some examples of communication details managed by e*Ways include rules for responding to or generating positive and negative acknowledgments, resend and reconnect criteria, time-out logic, data

envelope parsing and reformatting rules, permitted buffer size, retrieval/transmission schedules, and error logging and alerting.

SeeBeyond provides hundreds of application-specific e*Ways. The diversity of e*Ways already available ensures that an organization can use an extensive, available coding library as a basis when integrating a new system. Furthermore, SeeBeyond provides prebuilt Collaboration Rules that define business rules and logic commonly associated with given e*Ways.

APIs provide users with the ability to extend e*Way functionality, particularly when applications expose APIs themselves. Additionally, standard libraries and templates enable users to build custom e*Ways that take full advantage of the e*Way framework.

e*Ways for Database Access

These e*Ways deliver a powerful dimension of business process automation by enabling administrators to incorporate relational database access into enterprise-wide application integration strategies.

ETDs are generated by the database converter in the ETD Build tool. The ETDs map table columns, stored procedures and SQL statements, allowing users to gain access to databases or invoke stored procedures by manipulating them graphically rather than with complex database programming. These e*Ways can query a database and automatically generate a GUI tree representation of database access objects and populate the structure with the actual data values during run time. No SQL coding is required, but these e*Ways do support the full set of SQL operations, so advanced users can engage them directly.

The e*Ways for database access use the same GUI paradigm as the rest of the e*Gate system to describe Event flows through the entire enterprise. This enables business analysts to define the relationships between a database and relevant applications simply by dragging and dropping elements between tree structures.

SeeBeyond provides a number of specific e*Way Intelligent Adapters for database access, for example, Oracle, Sybase, and ODBC. These intelligent e*Ways extend the benefits of database connectivity to all databases (Oracle, Sybase, or ODBC, respectively) that communicate with the e*Gate system, without requiring users to purchase additional third party software

e*Ways for SAP

The e*Ways for SAP have been specifically designed to connect e*Gate to SAP R/3 enterprise management software within a network of diverse hardware and software systems. Using one or more e*Ways for SAP, e*Gate can act as a hub between SAP R/3 and other software systems, or between differently configured SAP R/3 systems.

These e*Ways control the communication protocol layer between the SAP host and e*Gate, and can be configured to process data in either direction. As with other e*Ways, they contain their own Monk engine to process mapping Collaborations without drawing on e*Gate resources.

Generic e*Way Extension Developer's Kits

These Kits provide templates for users to design and build custom e*Ways for their specific business requirements. The resulting e*Ways can incorporate core e*Gate

technology, using the e*Gate editors for configuration and the e*Gate monitors to provide extensive viewing and monitoring capabilities.

The Kits also provides a standard component within which to implement standard communication interfaces. It is tightly integrated with other e*Gate components, acting as a producer and consumer.

Users can configure e*Ways created with these Kits using the standard e*Way Editor. They can then extend those e*Ways by modifying their related Java, Monk, or other programming-language environments.

Additional e*Ways

SeeBeyond is continually developing new e*Ways to address special needs. See the SeeBeyond Web site at:

http://www.seebeyond.com/services/support.html

Also, you can contact SeeBeyond directly for the most current information on product availability.

BOBs

A BOB is similar to an e*Way in that it establishes connectivity. BOBs only communicate with IQs within e*Gate, they do not communicate with external systems as e*Ways do.

You can add BOBs to an environment if you prefer to abstract some portion of the data processing from the e*Ways, either to set up easily maintainable modularized processing or to optimize system performance by using multiple processes.

2.3 About e*Insight

A business process is a collection of activities and messages that flow in a defined sequence to produce an end result. Within an eBusiness Integration framework, e*Insight helps you organize these processes into *message-based integration solutions*. The message-based process modeling you do in e*Insight determines how data messages flow from activity to activity.

Note: *The rest of this guide provides diagrams, procedures, and discussions of specific system features, deployment methodologies, and architectures discussed in this section.*

2.3.1 Managing Business Processes

Business process management (BPM) is strategically automating the movement of information and the flow of complex processes between participants (systems, users, and organizations) to accomplish larger business objectives. e*Insight is also a BPM solution that delivers management and optimization by providing a clarity of view into the internal and external processes of an organization.

For example, you can use e*Insight to streamline operations, reduce costs, and increase customer satisfaction. In this context, e*Insight creates a layer of enterprise information flow and business logic that exists above an organization's current system-level processes. In this way, e*Insight can help you reach outward to include both customers and trading partners.

Using e*Insight to implement a BPM layer removes inefficiencies by orchestrating business processes into a unified work flow. This flow can include multiple systems and users, therefore extending to customers.

2.3.2 Ensuring Process Integrity

e*Insight can provide you with real-time monitoring and management that ensures process integrity. This application also includes sophisticated reporting tools and a complete business management environment. Using e*Insight, you can flexibly change, enhance, and integrate business processes as your enterprise grows and evolves. e*Insight enables clear visualization, monitoring, management, and optimization of business processes, spanning within and beyond the enterprise.

2.3.3 Benefits to You

e*Insight offers you the following important benefits:

- **Modeling:** UML-compliant graphical modeling and documentation of business processes that seamlessly integrate system and user activities

- **Synergy with e*Gate:** Automatic generation of e*Gate integration components allowing the rapid implementation of business process models

- **Monitoring:** Real-time graphical visibility and monitoring of running business processes spanning the enterprise, which allows for human interaction with the monitoring operation

- **Security:** The ability to assign levels of security for specific items, such as business process models, to users and groups

- **Exception Management:** Robust management of long-lived business processes to ensure process integrity, including the ability to roll back failed operations

- **Analysis and Reporting:** Graphical Wizards and reports for analyzing business processes, to identify trends and optimize results

These eBusiness Integration features in turn enable organizations to:

- Allow for meaningful human interaction with the ongoing business process

- Transform applications into components for both reuse and increased flexibility to adapt to change

- Increase visibility and improve the management of critical business processes with auditability and traceability for each participant's activity

- Optimize processes through reporting and analysis over time

- Enhance customer service and provide for customer self-service

- Increase utilization of critical assets and improve operational efficiency
- Expand capabilities to deliver new products and services at Internet speeds

2.3.4 The e*Insight Schema

The e*Insight Schema is the e*Gate schema that implements a particular e*Insight installation. The starting point for a working e*Gate schema for e*Insight are the e*Gate schemas provided with the product. These schemas are:

- eIJSchema (Java)
- eISchema (Classic)

These schemas contain a number of pre-configured and partially pre-configured e*Gate components used by e*Insight. In addition to the components that are provided, a complete e*Insight implementation requires several other e*Gate components that are added to the e*Insight schema during the implementation process. The pre-configured components that are used, as well as the additional e*Gate components that are added to make up the final working e*Insight schema, depends entirely on the specifics of the implementation.

The eIJSchema was introduced for 4.5.2. It is designed specifically to be used in a Java environment and all the components provided are Java based. The eISchema can be used in a combined Monk and Java environment.

The e*Insight database holds information about your e*Insight implementation, including configuration information and run-time messages. The common database model is automatically created when a business process is first created. This can be used with no modifications. However, a model specific database can be used which uses an optimized structure for the attributes. This can improve performance.

For complete information on e*Insight implementation, see the *e*Insight Business Process Manager Implementation Guide*.

2.4 About e*Xchange

e*Xchange provides eBusiness protocol support, allows effective partner management, and ensures secure eBusiness communications. It manages trading partner profiles and supports standard eBusiness process protocols, for example, ASC X12, RosettaNet, and UN/EDIFACT. e*Xchange includes a Validation Rules Builder tool to assist in creating X12 validation Collaborations.

Note: *The rest of this guide provides diagrams, procedures, and discussions of specific system features, deployment methodologies, and architectures discussed in this section.*

2.4.1 Exchanging Partner Information

The e*Xchange system allows you to set up and store information about each of your trading partners so you can exchange (send and receive) messages. e*Xchange also includes features to assist you with managing and troubleshooting the information exchange process.

2.4.2 e*Xchange Features

Specifically, e*Xchange allows you to:

- Receive, process, and route inbound and outbound transactions in batch, fast batch, and interactive transmission modes

- Use a Web-based management environment to allow configuration and support from any location that has an Internet connection

- Validate and translate messages/Events based on libraries of ETDs and Collaboration scripts that conform to eBusiness process protocols such as ANSI X12, CIDX, RosettaNet, and UN/EDIFACT

- Use a database (Oracle, SQL Server, DB2 UDB, or Sybase) to store trading partner information, transactions, acknowledgments, and errors

- Automatically generate and reconcile acknowledgments

- Handle and report errors

- Define and maintain trading partner profiles

- Generate custom reports that have been predefined in Seagate Crystal Reports

- Automatically support message enveloping as specified by the supported standards

- Monitor and view messages in Message Tracking

- Create and respond to action items

- Provide secure communications via the eSecurity Manager

Note: *The e*Gate system moves its own data in packages called Events. Similar types of data packages outside e*Gate can be called messages.*

For complete information on e*Xchange implementation, see the *e*Xchange Partner Manager Implementation Guide*.

2.5 About e*Xpressway

One of the most significant eBI challenges faced by eMarketplaces and Global 2000 businesses is connecting trading partners quickly. To meet this challenge, the eBI Suite includes e*Xpressway, the most powerful Web-based, trading partner on-ramp feature

available for Global 2000 business and B2B marketplaces. This application provides secure marketplace connectivity through file-based software packages.

e*Xpressway enables rapid trading partner connectivity and integration through a comprehensive B2B implementation methodology, graphical configuration wizards, and downloadable partner connectivity software. Trading partners follow an intuitive, step-by-step process for registering their company profile, configuring connectivity and integration software, then quickly installing their personalized software.

Note: *The rest of this guide provides diagrams, procedures, and discussions of specific system features, deployment methodologies, and architectures discussed in this section.*

2.5.1 e*Xpressway Integrator Server

e*Xpressway Integrator Server hosts the Trading Exchange Web site, which consists of an administration area where the administrator host controls membership and the contents of the download packages. This Web site provides both public and members-only services.

2.5.2 e*Xpressway Integrator OnRamp

e*Xpressway Integrator OnRamp is an extremely light-footprint solution that provides rapid eMarketplace connectivity through a standard or proprietary protocol-based exchange. e*Xpressway consists of an e*Gate schema—using a variety of e*Ways, such as HTTPs, Batch, CGI, and Apache Web Server—and a Java-based configuration tool for configuring connectivity parameters.

2.5.3 Hosting e*Xpressway Integrator

The Trading Exchange customer purchases the e*Xpressway Integrator Server, allowing them to host e*Xpressway and the Web site.

For more information on e*Xpressway, see:

- *e*Xpressway Integrator Server Setup and Maintenance Guide for Trading Exchanges*
- *e*Xpressway Integrator OnRamp Customization Guide for Solution Providers*
- *e*Xpressway Integrator OnRamp Setup Guide for Trading Partners*

2.6 Deployment: Getting Started

This chapter provided a comprehensive overview of the eBI Suite, its features, and its functionality. The rest of this guide explains generally how to deploy and implement your e*Gate environment.

Moving Forward

The following chapters cover all phases of planning and implementation for the eBI Suite deployment project:

- **Analysis and Planning:** For specific information on deployment planning and determining your system's general requirements, see **Chapter 3**. This chapter describes how you start deployment.

- **Hardware Requirements:** For information on planning your hardware requirements, see **Chapter 4**.

- **System Design and Transition to Production:** For information on these topics, see **Chapter 5** (design) and **Chapter 6** (transition).

Analysis and Planning

This chapter explains how to analyze your current business systems and processes in order to plan the optimum eBI Suite design and deployment to meet your stated requirements.

In This Chapter

- **"Introduction: Analysis and Planning" on page 38**
- **"Gathering Information" on page 39**
- **"Analyzing Your Requirements" on page 40**
- **"Planning Your Deployment" on page 51**

3.1 Introduction: Analysis and Planning

Deploying the eBI Suite requires completion of the following phases:

1 Analysis of requirements

2 Deployment planning

3 System design and development

4 Pre-transition testing

5 Transition to production

6 Post-transition maintenance

Figure 2 on page 39 shows a diagram of these six deployment phases. This chapter explains the first two phases, which are:

- **Analysis of requirements phase:** This deployment guide seeks to give you a road map of how to deploy an eBI Suite. First, to use a road map, you have to know where you are (analysis) and where you are going (planning). In other words, find out everything you can about your information system (IS) setup and business processes. Then, you can decide what IS and business process needs you want the eBI Suite to meet.

- **Deployment planning phase:** Deployment begins when you plan out and schedule how, in view of your analysis information and allocated resources, you want to implement your eBI Suite environment. During this phase, you set up the operation procedure and schedule for the entire deployment project.

The first two phases are the most important in the deployment project. This chapter discusses these phases in detail (later chapters treat the rest of the phases). Analysis and planning are information-gathering operations. Poor planning can cause serious problems during the later phases, but a good planning process can make system design and deployment easier, more efficient, and less costly. Thorough, comprehensive analysis and planning techniques lay a solid foundation for the entire deployment project.

Figure 2   eBI Suite Deployment Phases

Phase 1:

Analysis of Requirements

Phase 2:

Development Planning

Phase 3:

System Design and Development

Phase 4:

Pre-Transition Testing

Phase 5:

Transition to Production

Phase 6:

Post-Transition Maintenance
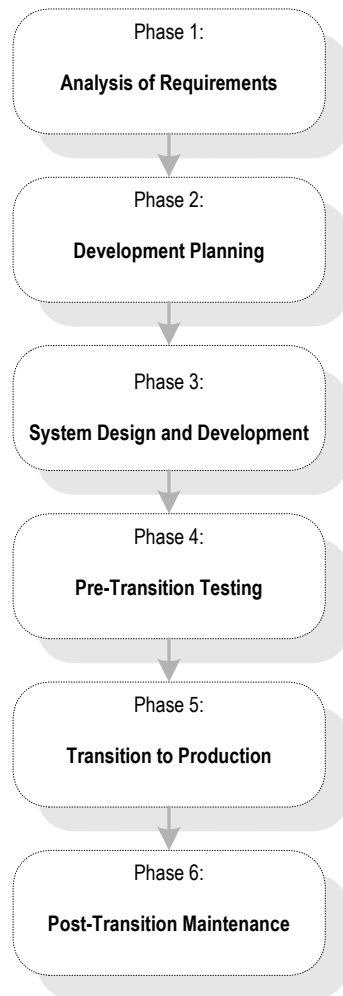
3.2   Gathering Information

You must prepare for your deployment project by gathering as much relevant information as possible. The more comprehensive your analysis and data are, the better. For best results, use the most modern survey and polling tools available to your organization to obtain the information you need, discarding unnecessary data.

Information-Gathering Tools

Use the following tools to assemble your deployment research:

- Research and interviews
- Surveys

3.2.1 Research and Interviews

These methods are the time-honored, traditional ways of gathering information. Use them as diligently as a college student writing a term paper. Your company has reams of paper, cabinets full of files, and databases overflowing with useful information, from management directives to marketing papers to MIS memoranda. Much important deployment information exists here, provided that you make good use of it.

Interview and talk to the employees of your organization. Find out what they do and what their IS needs are. Of course, input from relevant management and MIS people is necessary, but do not forget marketing employees, secretaries, and anyone else in touch with data flow needs. You want to put together a complete picture of your organization's current and future IS/business process requirements.

SeeBeyond's Professional Services department can help you in answering specific questions on how to gather data and what kinds of data are relevant for your own deployment project. You can utilize this resource, as necessary.

3.2.2 Surveys

Formal surveys are excellent tools for getting information. Surveys allow you to organize your own thoughts and processes, as well as helping to gather the desired information from others. There is a lot of helpful literature available on creating, giving, and analyzing polls and surveys. Reading some of this literature can provide a helpful background for doing these tasks.

Appendix A contains a sample survey. You can employ it as a self-survey or use modified versions of it to research and gather information from others. Feel free to print, copy, and use it as desired.

3.3 Analyzing Your Requirements

In gathering and analyzing information on your eBI Suite needs, you must first know what kind of information you need. Remember that the eBI Suite links to your current networks, business systems, and applications together into a single, seamless IS. The purpose of this system is to facilitate your current and future business process needs. In other words, in as much detail as possible, find out where you are and what you need.

In addition, e*Xchange links you to other businesses, trading partners, and systems outside of your organization, such as the Internet.

Examining Your Needs

During the analysis of requirements phase, you examine your needs and define the properties that the system must possess to meet those needs. Also, you identify system constraints and performance requirements. Define what functions you want the deployed system to perform but not how the functions work (this task happens during the design and development phase; see **Chapter 5**).

This section tells you what kinds of information you need to gather to facilitate your eBI Suite deployment, by posing a series of relevant questions. Make sure you answer all these questions as thoroughly and correctly as possible and discard any information that does not help you in answering them.

These questions fall into the following general categories:

- System-specific
- Operation and performance
- Personnel and training
- Business planning

Note: **Appendix A** *poses important questions under these categories, in survey form.*

Keep in mind that examples given in this section are general and are only meant to start you thinking in the right direction. You must begin by assembling general information on your needs, categorize that information, and expand on it by filling in necessary details to fully explain each category. See **Chapter 5** for more detailed examples of specific information you must put together.

3.3.1 System-Specific Needs

These needs are the basic IS, network, and database-related requirements you want your eBI Suite to meet. Determine your system-specific needs by asking the following questions:

What existing systems do we need to connect?

Create a complete picture of your current IS setup. Include applications, networks, systems, platforms, and outside information pathways.

Example: An Intel PC LAN with Windows NT network connecting workgroups with office applications, a UNIX system with an Oracle database containing customer information, and an IBM OS/390 system with IMS tracking financial transactions.

How do we want to do the connecting?

Find out how you want your various systems to talk to each other (communication protocols), which systems must be linked, and the direction of communication.

Example: We have systems A, B, C, and D. Systems A and B use TCP/IP, C uses SNA, and D uses SAP. All systems must talk to each other except system D which only needs to communicate with A. All communication in all systems is two-way, except that system C only needs to receive information from the others and not send it.

What are our data requirements?

What types of data do you use, how much, and when?

Example: Our system uses HL7 and X12 data types. On average, our system needs to move about 100,000 Events per day at about 5 MB per message, with 90 percent of that data moving between 8 a.m. and 5 p.m. every Monday through Friday. Peak data loads are generally between 2 and 4 p.m. on weekdays (60 percent of volume).

What are our system/hardware limitations and constraints?

Installing eBI Suite requires that you have the necessary hardware and operating system (OS) software and purchase (and install if necessary) additional hardware and software to contain the eBI Suite. Do you want UNIX or Windows? What is your budget for additional hardware and software? Do you have any space limitations in the area where this hardware will reside?

Example: We have compiled a detailed checklist showing all the hardware and software necessary to install and operate the eBI Suite.

Note: *Planning for hardware needs requires special considerations, for example, how many systems you need, memory (RAM) required, the number of CPUs you need, and total disk space.* **Chapter 4** *discusses in detail how to analyze and plan for these additional hardware and system-sizing needs.*

3.3.2 Operation and Performance Needs

Do you have any specific system operation and performance issues? Now is the time to discover, organize, and itemize them by asking the following questions:

What are our system performance requirements?

Ultimate system performance comes down to a trade-off between speed and maintainability. This fact is true overall, as well as being true for the operation of individual system component operations. You must prioritize these needs specifically.

Example: Our customer databases must be totally accurate and detailed because the information is often used and vital to the company. Detailed maintenance of this data is more important than speed of processing. However, our moment-by-moment stock quotations have to be fast and up-to-the minute. Maintainability here is negligible because this data changes so fast that long-term retrieval is not an issue.

What are our internal security requirements?

The eBI Suite has access security, that is, special features allowing only certain persons to log on to the system and different persons to have specific privileges after the log on.

Example: We will only allow five people to log on to the system: one with system administrator privileges, two with operator privileges, and two with monitor privileges.

Do we need to trade large amounts of data with other businesses?

The e*Xchange application allows you to manage trading partner profiles and support standard eBusiness process protocols.

Example: We need to manage a large number and many different types of data exchanges with our trading partners. We have a detailed list of how many and what kinds of transactions these are, including the types of data involved.

What are our data security needs in transactions with other businesses?

The eSecurity Manager (part of e*Xchange) allows you to send and receive secure transmissions of B2B exchanges over public domains, for example, the Internet.

Example: We must be able to exchange financial data with our trading partners, when necessary, via the Internet. We know how much data needs to be exchanged, what kind of data, and when the transactions take place.

What are our error-handling and data validation requirements?

How, when, and where in the system do you want data checked for errors and validated? Keep in mind that processing speed decreases as checking instances and the detail of error checking increases.

Example: All data passing through our eBI Suite must be validated to the most thorough extent possible. To facilitate this process, we have compiled a complete list of all the different types of data that need to be validated.

What are our auditing requirements to maintain historical data?

How long does your organization need to maintain auditing data that has passed through the e*Xchange Engine for non-repudiation purposes and accountability? Keeping large amounts of data within the database eventually causes performance problems and could demand additional hardware resources (for example, disk space and processor capacity). Think about how often you need to archive data to keep the database optimal and how it can be stored so it can be easily de-archived if auditing needs demand data availability.

Example: We expect to do around 1,000,000 transactions per week and have calculated that our Oracle database and hardware can accommodate approximately 25,000,000 transactions online without causing any problems. Based on these calculations, we will schedule to run an archiving routine every two weeks to remove transactions older than six months. Each of these archive files will be saved to tape and stored for seven years in the off-site safe. This process will not replace any of our standard backup and recovery procedures.

Do any of our trading partners want to connect their back-end systems through to us?

e*Xchange can greatly facilitate the management of trading partner definitions and associated validation, security, and auditing. However, some of your trading partners may also want to provide the capability for them to connect their systems through to your trading environment. In such cases, you must assess what systems trading partners have and whether they need a tighter integration environment to trade with you. Using e*Xpressway you can facilitate these needs. Also, try to categorize the different types of integration your trading partners need.

Example: We have 1000 trading partners. Approximately 90 percent of them will just want simple Web-based access that we can provide via e*Xchange and custom solutions within e*Gate. However, 10 percent want to connect their own systems. Many of them have file-based X12, but some trading partners have custom file formats. We will need to document these integration requirements and ensure that OnRamp

Solution Packages are built to fulfill each custom need. We also need to identify who will help us build/provide these solutions to the trading partners.

What kinds of transactions do we need to handle?

Within the supported eBusiness protocols in e*Xchange, there are many different transaction sets within each protocol. It is extremely helpful to determine in advance which standards are needed and which versions and transaction sets you have to support, so you can set them up for your trading partners.

Example: Most of our partners in Europe use UN/EDIFACT 99a or 00a and many in the U.S. use X12 4010, 4020 and 4041. We will be exchanging orders with all of our suppliers and confirming those orders, with acknowledgement handling. Therefore, our common set of transactions for UN/EDIFACT must be ORDERS, ORDRSP, and CONTRL. For X12, we have to support 850, 855, 857 and 997. Additional transactions will be supplied on an as-needed basis by each trading partner.

What communications protocols do our trading partners require?

With the core e*Xchange application, you can support HTTP, HTTPS, SMTP, and FTP Batch for many of the eBusiness protocols. Determine which communications protocol each trading partner wishes to use and how best to set-up the e*Xchange Engine to support them.

Example: Many of our trading partners want FTP connectivity. For each one, we will set-up a special directory where they can put their files to be picked up in batch mode by e*Xchange (and define this in their profile setup). For our HTTPS users, we will need their certificate to store in the Repository, as well as supplying them with a URL that they can point their products to, to send us data. We will also need to obtain their URLs so we can POST data to them.

3.3.3 Personnel and Training Needs

Deploying the eBI Suite may require some expanded personnel needs, so you must ask yourself the following questions:

Do we have personnel trained and able to deploy the system?

Deploying the eBI Suite does require some training of current personnel and may require the hiring of additional persons, depending on the size of the system you are planning and implementing.

Example: We will need two persons to deploy and later operate the new system. One new person will have to be hired. All three will need to take the basic eBI Suite class (offered by SeeBeyond) and one will additionally have to take the advanced class (also offered by SeeBeyond). The new hire must be thoroughly trained in and familiar with UNIX (training in UNIX is not offered by SeeBeyond).

Do we have personnel trained and able to maintain the system after deployment?

Post-transition maintenance of the eBI Suite could also require additional personnel and training.

Example: In addition to the people we hired to deploy the system, we need to train one additional person to learn how to operate it, in order to enable long-term maintenance of the system.

3.3.4 Business Planning Needs

The eBI Suite can help you facilitate and improve your overall business processes. Assess your needs in these areas by asking the following questions:

What business processes do we want the eBI Suite to help us enable?

e*Insight allows you to design your business process models. Before starting this design process, you must know the overall flow of business processes you want the eBI Suite to help you maintain and enhance, including future needs you want the eBI Suite to help you meet. e*Insight allows you to map out these business work flows in great detail.

Example: We primarily need to facilitate the flow of data from the Sales Department to Customer Service and Orders Processing, so the different departments can access one another's data, and we can easily keep track of each of our customers and their interactions with all these areas of our company. This is a complex, detailed work flow that requires precision planning and analysis. Using e*Insight helps us diagram and evaluate this work flow and its attendant business processes.

What are our record-keeping and documentation needs?

Make sure you set up a system for documenting your eBI Suite operation.

Example: We must put a new system in place to document and diagram the total operation of the eBI Suite. In addition we must keep complete records on that operation.

How do we create a deployment road map?

Plan your deployment well. Choose a Deployment Project Team (for a small deployment, one person could do this task) to carry out the project, and make sure you document your plan in writing. Flowcharts and system diagrams are definitely helpful (see **"Planning Your Deployment" on page 51**).

Example: Figure 3 shows a diagram of the information-gathering cycle in the deployment project's analysis of requirements phase.

Figure 3   Analysis of Requirements Phase/Information-Gathering Cycle
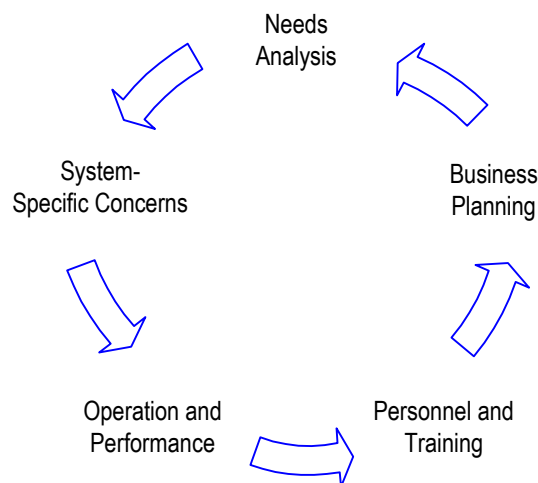
As you continue the analysis process, allow the results to feed back into your overall analysis. If necessary, start the process over again to fine-tune the information you have gathered. This method allows you to ensure the accuracy and usability of the requirements you collect.

Once we have the information, what do we do with it?

Complete the process of documenting and organizing your information as correctly and comprehensively as possible. When you are finished with the analysis of requirements phase, you will use this information to help you with the next phase, planning your eBI Suite deployment project.

3.3.5 e*Insight Deployment

This section explains additional considerations to help you in planning to deploy e*Insight along with your total eBI Suite.

Sample Business Process

Successfully deploying e*Insight requires that you think through and analyze your entire business process. You can easily make this analysis operation a part of your overall eBI Suite deployment planning. Because e*Insight allows you to graphically model your business processes, this application, once it is installed and operating, can be a useful tool in your overall planning process.

For example, as demonstrated in the business process example shown in Figure 4, information critical to customer satisfaction must flow from the customer, across several internal systems, and out to business partners, such as credit agencies, alternate suppliers, and product shippers.

Figure 4   Business Process Example
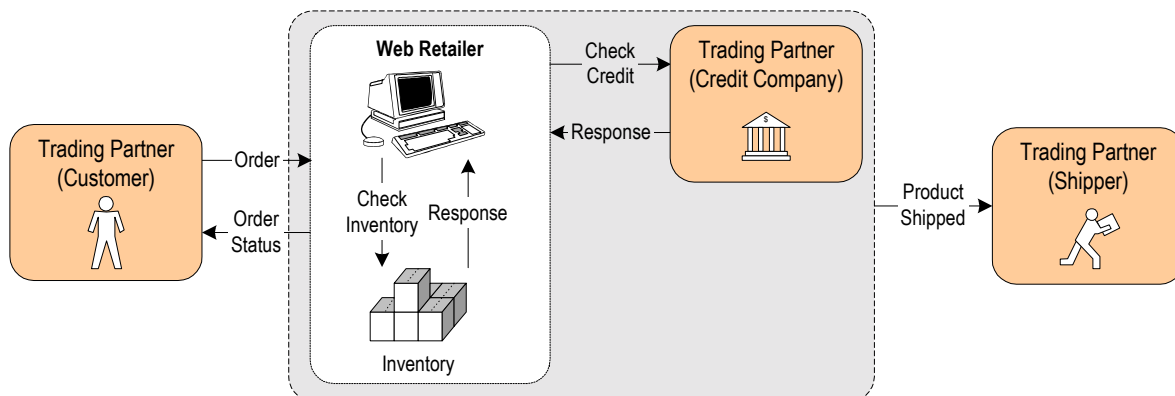
To efficiently and reliably manage the information as it flows from system to system, this type of complex business process clearly benefits from a single, automated process-management application. Without this infrastructure in place to deliver process management services, significant additional custom development is required to remove or minimize the otherwise manual steps.

e*Insight allows you to create, automate, manage, and optimize cross-application and cross-enterprise business processes involving systems, users, and external organizations.

For more information on e*Insight, see the *e*Insight Business Process Manager User's Guide* and *e*Insight Business Process Manager Implementation Guide*. In addition, refer to:

- **"Modeling Business Processes with e*Insight" on page 123**
- **"Role of e*Insight" on page 172**

3.3.6 e*Xchange Deployment

This section explains additional considerations to help you in planning to deploy e*Xchange along with your total eBI Suite.

Business-to-Business Integration

Electronic Business-to-Business (B2B) Integration, or eBI, does more than allow one business to send electronic documents to another. eBI automates and integrates the entire business supply chain so that a business process that uses external trading partners can be managed as a single process. The purpose of e*Xchange is to facilitate and integrate the many trading partner relationships in a business.

In moving from intra-business to inter-business, the integrator must overcome several challenges, most of which stem from the need to use infrastructure that is outside one's control. Once these challenges are overcome, the enterprise can manage the entire end-to-end business process. Using e*Xchange, the enterprise can extend the proven planning and cost savings abilities of Enterprise Application Integration (EAI) to the larger world of eBI.

An eBI Example

The need to integrate a number of trading partners is an essential requirement in the realm of internet retailing. For example, consider a Web retailer that sells sports equipment online. This retailer sets up an electronic storefront that allows a customer to browse an online catalog of items and place orders for them.

After securing payment via credit card, the items are shipped to the customer, along with the status of the order. **Figure 5 on page 48** shows a flow chart of the web retailer's business process outlining the steps involved in a typical transaction.

Figure 5 Web Retailer Business Process
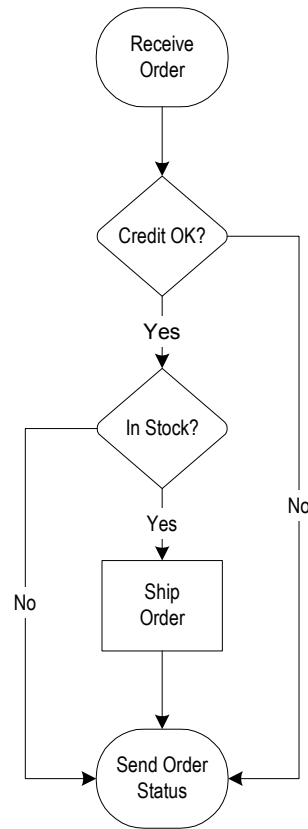
Three out of the five steps in this business process (checking credit, stock availability, and shipping to the customer) are outside the Web retailer's enterprise.

However, from the customer's point of view, the entire transaction is handled by the online retailer. The Web retailer's business model depends on the efficient use of trading partners to fulfill parts of the business transaction that he does not handle directly.

Figure 6 shows the interrelationships between the retailer and the trading partners:

Figure 6 Trading Partner Relationships
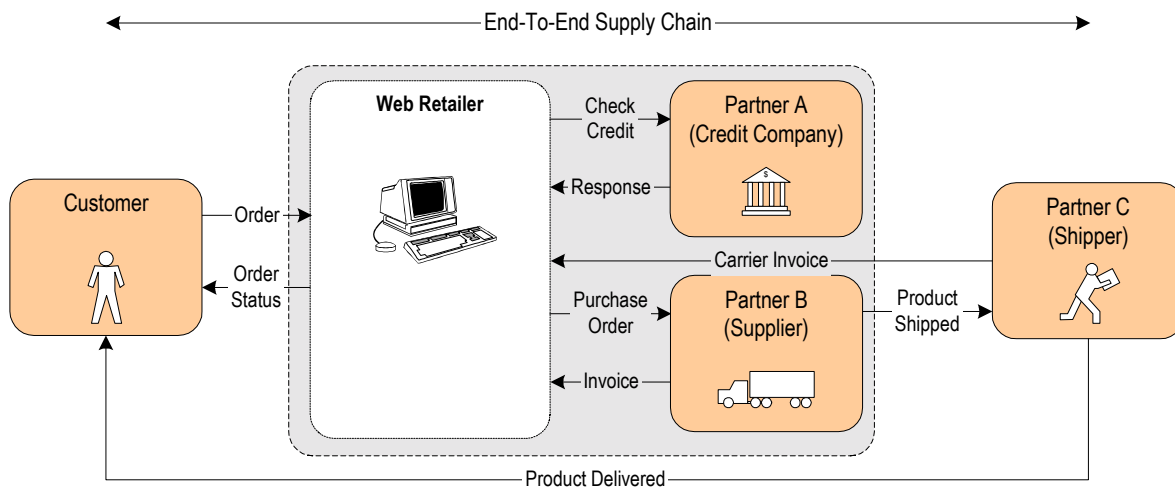
e*Xchange Deployment Methodology

The goal of e*Xchange is to successfully integrate a business' many composite trading-partner relationships into a single overall business process. The successful deployment of e*Xchange creates a composite eApplication that orders and automates these many "chains," making them easily manageable.

Therefore, planning your e*Xchange deployment entails knowing, documenting, and preferably diagramming (see Figure 6) these relationships in as much detail as possible before starting.

For more information on e*Xchange, see the *e*Xchange Parner Manager User's Guide* and *e*Xchange Partner Manager Implementation Guide*. In addition, refer to:

- **"Overview of e*Xchange Implementation" on page 127**
- **"e*Xchange Message Tracking" on page 184**

3.3.7 e*Xpressway Deployment

This section explains additional considerations to help you in planning to deploy e*Xpressway along with your total eBI Suite.

Deployment at a Glance

Deploying e*Xpressway requires that you take the following planning steps:

1 Identify your trading partners who need e*Xpressway or could benefit both your business processes by using it.

2 Analyze and, if necessary, document how they can configure e*Xpressway to improve your joint business processes.

3 Determine what you will require to host e*Xpressway.

Basic Deployment Considerations

Deploying e*Xpressway requires the following prior considerations:

1 You need a Solution Provider (a consultant with e*Gate experience) to customize the OnRamp base schema within e*Gate to provide custom e*Xpressway OnRamp Solution Packages. Your Solution Provider must work with you to create the OnRamp template schemas and Solution Packages.

2 You need to know what systems your trading partners are using. You want your system to be able to communicate with the applications and platforms likely to be used by your trading partners.

3 Your Solution Provider needs to create OnRamp Solution Packages specific to each of your trading partners for uploading to the Web site (and for them to download).

Figure 7 shows a diagram of a sample e*Xpressway deployment plan.

Figure 7 Sample e*Xpressway Deployment Plan
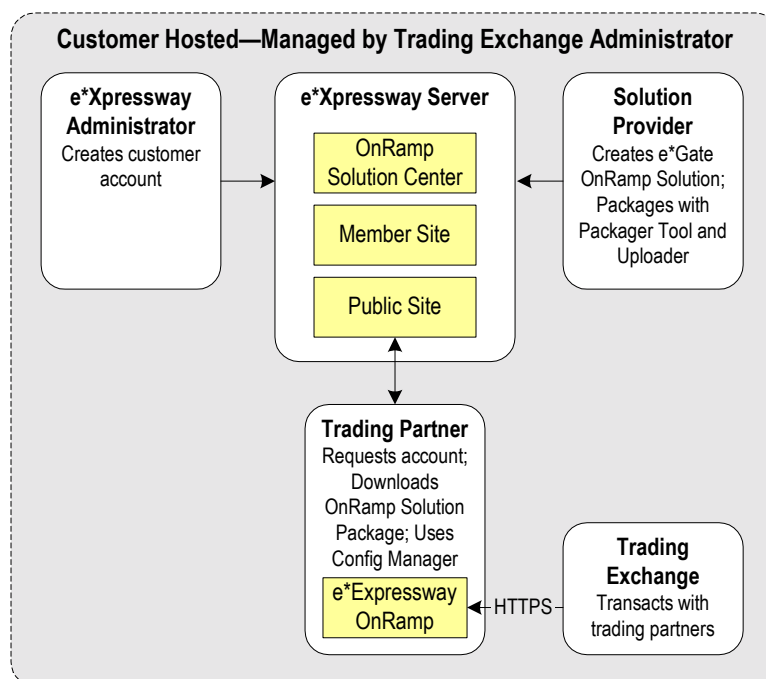
For more information about using and implementing e*Xpressway, see the following references:

- *e*Xpressway Integrator Server Setup and Maintenance Guide for Trading Exchanges*

- *e*Xpressway Integrator OnRamp Customization Guide for Solution Providers*

- *e*Xpressway Integrator OnRamp Setup Guide for Trading Partners*

In addition, refer to:

- **"Overview of e*Xpressway Implementation" on page 131**

3.4 Planning Your Deployment

The deployment planning phase is the next major step in your eBI Suite deployment project. In planning your deployment, you create a road map of what that deployment will look like. You must include criteria like resources, schedules, goals, and objectives. The primary purpose of this phase is to initiate the project, define the integrated system to be developed, create top-level design documents, and create a formal project plan or road map.

In creating your deployment road map, you provide a detailed description of the integrated eBI Suite to be developed. This plan serves the following primary purposes:

- Designing what your future system looks like

- Showing you the resource allocation needed to implement the design

If analysis is finding out where you are, planning tells you where you want to go and how to get there. You can obtain help, when necessary, from SeeBeyond's Professional Services and other SeeBeyond representatives. Thorough and comprehensive planning helps to ensure a smooth-running and satisfactory deployment project.

The major steps in deployment planning are:

- Setting up overall objectives

- Identifying and scheduling tasks

- Determining when objectives are met

3.4.1 Setting Up Overall Objectives

This step of the deployment planning phase entails the following operations:

1 Achieve a consensus on the implemented eBI Suite's overall functionality and scope by taking the following steps:

   ◆ Set up organized technical and functional teams or roles to handle individual phases and aspects of the deployment.

Note: For a small deployment, one person could handle the tasks of a team.

   ◆ Ensure that the system's functionality is clearly stated and agreed upon.

   ◆ Document the functionality and scope of the project based on analysis information, as well as match this information against the scope of the project as stated in the "Approved Proposal."

   ◆ Resolve any differences between the "Approved Proposal" scope and your prepared analysis and requirements information (see **"Analyzing Your Requirements" on page 40**), if necessary.

2 Create a general model of what the system will do. This model serves the following purposes:

   ◆ Serves as the foundation architectural plan for all eBI Suite design (see **Chapter 5**).

⬧ Consists of diagrams and supporting documentation that represents the design strategy for any required eBI Suite interfaces.

3 Set up a Design and Development Team or role and provide this team/role with an understanding of the application domain. Also provide them with approved, clearly stated, top-level design documentation of requirements for the eBI Suite domain.

4 At this point, the groups and persons meeting together must formulate a basis of validation of the final product during acceptance testing (see **Chapter 6**). This validation process includes the testing required to validate the functionality of the system and that it works as stated in the "Approved Proposal."

3.4.2 Identifying and Scheduling Tasks

This step of the deployment planning phase includes:

▪ Deployment initiation steps

▪ Creation of deployment documents

Beginning Deployment

Begin the deployment project via the following actions:

⬧ **Hold a Project Kick-off Meeting:** This meeting will identify all members of the Deployment Project Team or role. The analysis tasks and responsibilities assigned to each resource will also be identified. The purpose of this task is to outline the reporting structure for the project and identify whom the Project Manager will communicate with to ensure that other tasks in the project are being completed as planned. In addition, documentation standards and the project reporting structure are established at this time.

⬧ **Ensure Software and Hardware Installation:** The purpose of this task is to ensure that your hardware and software is in place and ready for the eBI Suite installation. This process includes ensuring that the eBI Suite software is fully supported on your hardware platform and operating system and that the software has been shipped.

⬧ **Complete Installation Test, Installation, and Checklist:** The eBI Suite installation task is completed during the deployment planning phase to ensure there are no issues with your technical environment. You can use a deployment checklist to detail the exact hardware and operating systems where the installation will be performed. This task includes the following steps:

◆ The total eBI Suite environment must be installed and tested. The deployment checklist is updated to identify what items were completed and document outstanding issues that may have kept any items from being completed.

◆ The production, training and test (pre-production) hardware, software and network requirements (current and planned) are identified and verified.

⬧ The end-to-end communications with your other systems are also tested to ensure that communications are set up correctly and systems are exchanging messages correctly according to the communication protocol being invoked.

⬧ Any communications with other businesses or trading partners are tested in the same way.

◆ **Establish the Change Management:** A critical factor through all phases of the project is change management. Change management identifies and track all changes for a project that depart from the original deployment plan. All changes must be identified and tracked because many small changes can and will impact a deployment project in the same way as a more easily identifiable large-scale change. Tracking all changes allows the project manager to plan and control a project and keep track of all changes in the project's scope.

Deployment Documents

You must document the deployment project. This step requires that you create the following documents:

▪ **Preparing the Deployment Project Plan –** This document lists a set of tasks for establishing a baseline reference plan. It is your road map for the deployment project. The roles and responsibilities of each organization, schedule of tasks, and any estimates must be defined in this plan.

It is best that this plan be as detailed as possible. Any project risks must be assessed and documented. Your necessary resources are budgeted using this plan (or validated if you have already created a budget).

The deployment project plan must be reviewed and agreed on by all the organizations involved in the project. It must be communicated to all affected organizations. The following table shows a list of the subject matter the plan must contain.

Table 1   Deployment Project Plan

Contents	Description/Methods
Scope of work	This item must be based on the purchase contract, "Approved Proposal," or any equivalent document.
Project organization	Include the deployment project team (or the person responsible for a small deployment), development organization, review organization and any external organizations involved in the project. The roles and responsibilities must be clearly defined.
Delivery schedule	Indicate the schedules for all specified eBI Suite deliverables, including the final delivery date after all validation and verification tasks are complete.
Estimates	This section includes a work breakdown structure (WBS).

Table 1 Deployment Project Plan (Continued)

Contents	Description/Methods
Overall schedule	This section contains a schedule for all deployment project tasks including resource assignments; key phase completion milestones must be indicated. This schedule will be further elaborated by developing various phase work plans (the use of Microsoft Project is recommended).
Resource requirements	Include the manpower, hardware, and software resources required for the testing phase before transition to production.
Issues and risks	Potential project issues and risks must be identified, and contingency plans must be drawn up for any risks.
Organizational interfaces	The dependencies on other projects and information needed from other organizations must be clearly identified, documented, and conveyed to any affected parties.

▪ **Functional Requirements Specification:** In creating this document, you identify and analyze your specific system requirements. The behavior of the various application components (Event, process, and associated data) are carefully analyzed and documented. You must check and verify each of these components.

The functional requirements specification, along with technical requirements specification, helps form the basis for system design and final project acceptance. The typical subtasks in creating this document are:

- Studying and identifying system requirements to derive the business process functionality required and identify the system architecture needed to meet functional requirements.

- Creating an eBI Suite architecture model to show the proposed integrated system.

- Creating eBI Suite interface models to define interface requirements.

- Being sure that, at every step in its creation, you are allowed to review, give input to, and sign off on this document. Table 2 shows a list of this document's contents.

Table 2 Functional Requirements Specification

Contents	Description/Methods
Statement of requirements	Define the objectives you want the eBI Suite to meet.
Proposed eBI Suite architecture	Show a summary design model of the sending and receiving systems, e*Way Intelligent Adapters to be used, and interfaces that take place.
Proposed directory structure and Events that trigger eBI Suite processing	Provide a map of the external sending/receiving systems, directory structures, and the business/processing Events, including what eBI Suite processing will be initiated by the Events.
Exception processing	Define requirements for processing errors or Events.

Table 2   Functional Requirements Specification (Continued)

Contents	Description/Methods
Constraints	Define data volumes, performance, and any backup/archive requirements.
Interface diagrams	Produce a diagram for each proposed interface, showing the sending/receiving system, Event processing, and any interdependencies.
Hardware/software diagrams	Show the hardware/software environment and high-level related schematics for development, testing, and production systems.

The general design model provided by this document forms the starting basis of the next deployment step, the system design and development phase. See **Chapter 5** for details on how to use this model as the foundation for your complete eBI Suite architecture. For examples, see the following figures:

- Figure 8 shows a sample diagram of an e*Gate environment.
- **Figure 9 on page 56** shows a sample diagram of e*Xchange and e*Insight installations.

Figure 8   e*Gate Environment—General Diagram
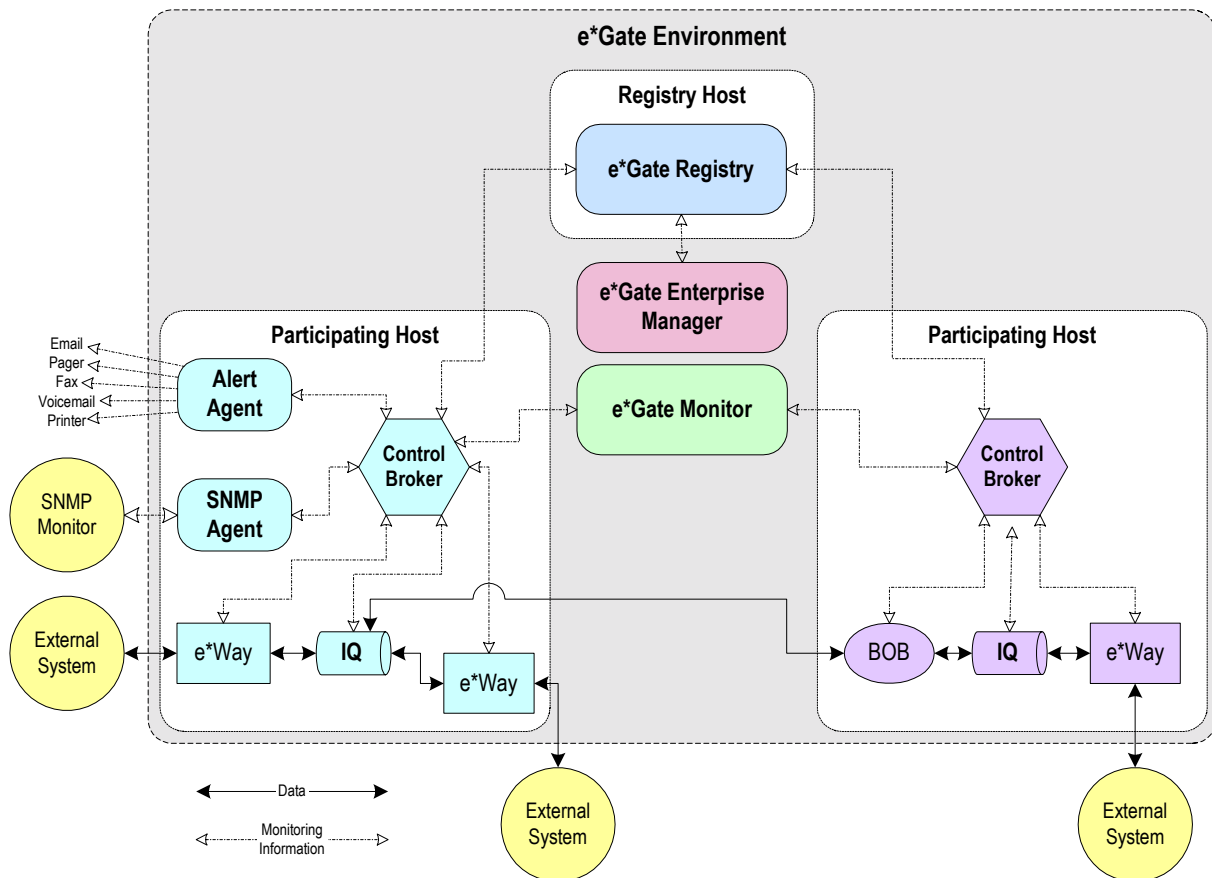
Figure 9 Sample e*Xchange and e*Insight Installations Diagram
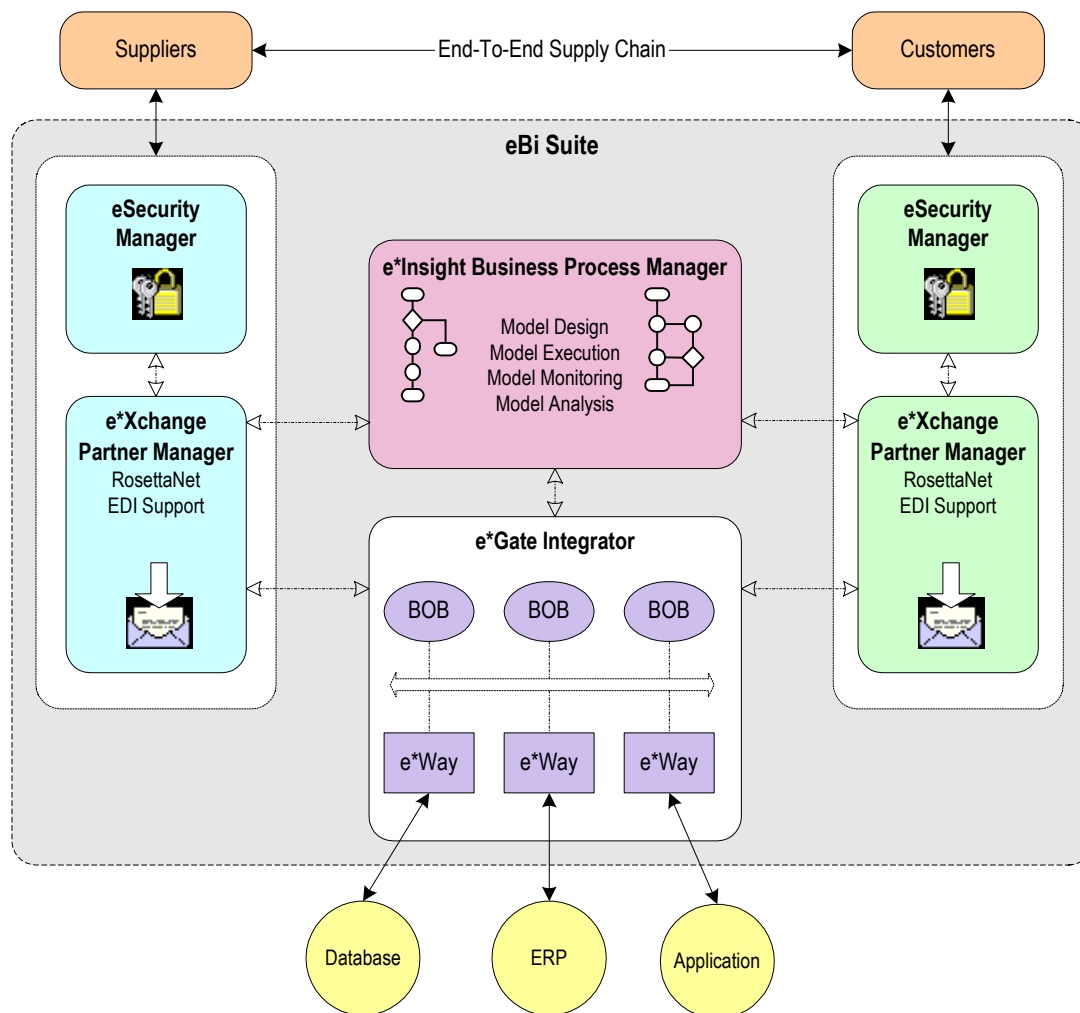
Note: *The eSecurity Manager is a part of the e*Xchange Partner Manager system. See the* **e*Xchange Partner Manager User's Guide** *for details.*

▪ **Technical Requirements Specification:** In creating this document, you identify and analyze your specific technical requirements. The behavior of the various application components (Event, process, and associated data) are carefully analyzed and documented.

Of course, you have input on and verify the need for each of these components. This technical requirements specification, along with the functional requirements specification, helps form the basis for system design and final project acceptance. The typical subtasks in creating this document are:

   ◆ Creating a hardware/software model to define the environment that eBI Suite will process in.

   ◆ Being sure that, at every step in its creation, you are allowed to review, give input to, and sign off on this document. The following table shows a list of this document's contents.

Table 3   Technical Requirements Specification

Contents	Description/Methods
Technical requirements specification	Requirements for security, system availability, and the technology being used to meet these requirements.
	Any additional related requirements.

▪ **Test Plan Requirements Specification Document:** A high-level test plan must be produced, highlighting the testing tasks to be performed during each phase. This document specifies the test approach, the type of tests to be carried out, and the organization responsible to carry out the tests for each test phase.

Important: *A detailed test plan is developed during the design phase (see* **Chapter 5***). The actual testing is carried out during the testing phase before transition to production (see* **Chapter 6***).*

The test plan requirements specification can be a single document, or it can consist of a separate document per project for all the test phases, or one document per phase, depending on the size and complexity of the deployment project. Table 4 shows a list of this document's contents.

Table 4   Test Plan Requirements Specification

Contents	Description/Methods
Test plan	Contains a general description of the testing phase of the deployment project; this plan is preferably produced by an independent test team or role based on your requirements for the testing of applications and their process for promoting applications to production.
Test phases	Includes the programmer test, unit test, integration test, system test, roll-out test, operation readiness test, and so on.
Test approach	Details whether there will be manual or automated testing and the validation process for each test performed.
Organization	Includes the testing team or role (functional and technical).
Schedule	Defines the system availability for test data and system resources needed for the different test phases.
Resource requirements	Defines system, individual, and team resources needed for the test phases.

Chapter 6 contains a complete description of the testing, transition to production, and post-transition maintenance phases of the deployment project.

Note: *The tasks stated in this guide walk through the steps required for developing and delivering the eBI Suite interfaces. This document addresses the interfaces as a total system, and so presents the deployment planning phase as being completed before the system design and development phase. In an actual deployment, there is probably overlap between the two phases. This fact, of course, applies to all the phases.*

3.4.3 Determining When Objectives Are Met

All deployment planning phase objectives have been met when the following steps are completed:

1 The deployment project is successfully initiated, including the following completed tasks:

  ◆ Your deployment project team and eesign and development team are set up, and their assignments and responsibilities are identified.

Note: *For a small deployment, one person could substitute for each of these teams.*

  ◆ The deployment checklist is completed.

  ◆ The prerequisite hardware and software has been identified, installed, and tested.

  ◆ A working change management process is established.

2 The deployment project plan has been completed, updated (if necessary), and approved by deployment project leadership and your management.

3 The functional, technical, and test plan requirements specifications are completed and approved by deployment project leadership and your management.

4 The deployment project leadership must review and approve the following requirements:

  ◆ Architecture design documents must be completed and approved.

  ◆ Test requirements must be identified, documented, and approved.

  ◆ All analysis information must be verified as detailed and accurate enough to predict the deployment's cost and duration.

5 Any subsequent issues resulting in a change of the project scope and resources must be communicated and signed off, including approval by deployment project leadership and your management.

Figure 10 on page 59 shows a complete diagram of the steps of the deployment planning phase, as discussed in this section.

Figure 10 Deployment Planning Phase Steps

Set Up Deployment Objectives

1. Get consensus on overall functionality
2. Create system model
3. Set up design and development team
4. Create basis for validation

Identify and Schedule Tasks

1. Hold project kick-off meeting
2. Ensure software and hardware installation
3. Complete installation test, installation, and checklist
4. Establish change management
5. Prepare deployment documents

Determine When Objectives Are Met

1. All tasks are identified and scheduled
2. Deployment project plan is completed
3. Requirements specifications are completed
4. Deployment project leader has approved:
   · System model
   · Test requirements
   · Cost/duration requirements
5. All change issues are approved

Moving Forward

Once all the members of your management and deployment project team (or responsible person) agree that these objectives have been met, you have successfully finished the deployment planning phase of your eBI Suite deployment. You have already created a complete deployment road map, the "deployment project plan," along with some general designs for your completed system.

Hardware Needs: An important part of planning for your system and deployment is how to determine your hardware requirements. If you need additional information on planning for and determining your system's hardware requirements, see **Chapter 4**.

The rest of the chapters in this guide treat the eBI Suite deployment project under the following topics:

- **System Design:** For a discussion of the next phase of your deployment project, including system design architecture and development, see **Chapter 5**. In this phase, you broaden and fill in the details of the general designs you created during planning.

- **Testing and Transition to Production:** For a discussion of the testing, transition (go-live), and maintenance (fine-tuning) phases of your system deployment, see **Chapter 6**. These phases follow the system design and development phase.

- **Helpful Tips: Chapter 7** "Frequently Asked Questions" gives you some helpful hints and tips for best practices.

Determining System Requirements

This chapter offers guidelines to help you determine the system requirements for the deployment of the eBI Suite.

In This Chapter

4.1 Introduction: System Requirements

This chapter explains how to assess your needs for the following types of hardware:

- CPUs
- Hard disk space
- Random access memory (RAM)

There are many variables and factors to consider in order to adequately determine the hardware requirements for your particular system. As such, this discussion will be limited to issues as they relate directly to the eBI Suite.

This chapter does not consider networking topology, and does not address such issues as shared applications, how resources are distributed throughout a network, and how many workstations are included in the network. Furthermore, in the case of databases, it is assumed that each database management system is installed on a separate host. See **Chapter 5** for details on these considerations.

4.2   Initial Considerations

The eBI Suite merges traditional Enterprise Application Integration (EAI) and Business-to-Business (B2B) interactions into a multi-enterprise eBusiness system. The eBI Suite includes, among other components, e*Insight for managing business processes, e*Xchange for managing connectivity to external systems, and eSecurity Manager (part of e*Xchange).

Depending on the number of external connections, the type of data being processed, and how the data is processed, the required resources can vary. Take the following points into account as you begin estimating your hardware requirements:

- Each eBI Suite deployment is different. Obviously, this is true when custom systems and enhancements to existing systems are present. The configuration of each deployment is unique because there are varying numbers of components as well as variances in interconnectivity. Some components are bidirectional and complex, while others merely pass data through.

- In addition to differences in configuration, the computational requirements will differ both in scope and complexity. The high-performance architecture of the eBI Suite is net-centric, not server-centric, not client-centric, and not hub-based, which makes the eBI Suite highly flexible. It is this flexibility that makes predicting the general requirements of hardware a complex task.

- The eBI Suite solution is distributed via run-time components and is platform-independent. System stability and redundancy are important considerations. Server requirements vary greatly, depending on the components resident on the server, the archiving requirements configured on the customer's system and other factors.

- Instead of only referring to absolute minimum requirements, it is more meaningful to discuss the hardware needs of an installation in terms of recommendations. By using the methods set forth in this chapter, a system analyst can estimate the required resources for a given configuration, from a simple deployment to a complex deployment, and thereby define an initial recommendation. Once installed, the eBI Suite can be fine-tuned, both in terms of hardware and software, to optimize performance.

4.3   Estimating Processor Requirements

The eBI Suite has been deployed in an extremely wide variety of environments, from simple deployments of a single system with a single CPU to highly sophisticated configurations consisting of 64 500-MHz CPUs that process one billion transactions per day.

4.3.1   Consideration Factors

Because the eBI Suite is a general-purpose toolkit that supports multiple languages with a scripting language, and is completely flexible in its deployment and

configuration, estimating the processor requirements is a challenge. There are infinite possibilities and numerous factors with complex interactions that affect the estimations. Some of the factors are more critical than others, depending on the circumstances.

For example, the effect of limited RAM resources that create a paging/swapping situation could completely hide the effects of a complex ETD or a poorly written function. Some of the factors that affect performance in the eBI Suite are:

- CPU type and architecture

- CPU speed

- Presence of a CPU cache and its size

- Number of CPUs

- Physical memory size

- Swap size

- Disk subsystem, that is, bandwidth, latency, block size, RPM, seek time, and the presence and size of the cache

- Network bandwidth and load

- Number of external systems and their latencies in servicing messages and acknowledgements

- Complexity and amount of processing to be performed by each component

- Event volume, size, and distribution through the day

- Throughput and response-time requirements

- Complexity of Events, including the number of nodes and complexity of regular expressions

- Bundling of Events, that is, more than one logical record in one physical record

- Number of transitions between components for a given Event (for example, moving data from an e*Way to an IQ to an e*Way or BOB)

- Type of IQs used

- Number of subscribers to each publication

- Amount of the implementation that can utilize parallel processing

- Other loads on the Participating Hosts (for example, IQ reorganization schedules, back-ups, and other processes)

- Dispersion of solution across multiple CPUs and systems

- Number and architecture of eBI Suite subcomponents participating in the schema

Not only are there more factors, but these factors need to be assessed for each Participating Host in a distributed eBI Suite deployment.

4.3.2 General Guidelines

There is no standard benchmark in the EAI industry like there is in the Database Management System (DBMS) industry, that is, the Transaction Processing Performance

Council (TPC) benchmark. It is debatable whether a benchmark can ever be developed, which could accurately and reliably predict the processing requirement for a given integration implementation. This difficulty results from the number and complexity of factors that could affect performance. The resulting equation would be impractical to use because of the large number of parameters and their weights.

Because there are many areas in which the architecture can be tuned to achieve further performance gains, each time a new change is made, the performance characteristics may be different. Another problem is that any benchmarking equation would rely on other measures that are just as problematic, for example, measuring program complexity lines of code (LOCs) or function points (FPs).

Pragmatic Approach

A more pragmatic approach is to start with a good base configuration as a development system and use that configuration to predict the processor requirements for a production system for your unique implementation. The minimum hardware requirements for a typical e*Gate configuration of 20 interfaces would be one of the following systems:

- Windows NT/2000 system running on dual Pentium III-class 866-MHz CPUs

- e*Gate-supported UNIX system running on two 400-to-500-MHz CPUs

- Both of the above requirements with 1 GB (1000 MB) of RAM and 20 GB of hard disk space, preferably in a hardware chassis that supports more than two CPUs

The recommended methodology is to implement a representative number of interfaces (that exercise a good sample of the various data transformations and communication requirements of the implementation) on this system, run representative data files through the system, and record the CPU load. From these measures, you can project what the final production load will be and therefore the CPU requirement. Of course, the architecture can be tuned to achieve more efficiency using this same technique.

One of the advantages of the eBI Suite's distributed-and-scalable architecture is that hardware does not need to be replaced but can be included in a multi-system implementation. Therefore, as processing requirements grow, you can easily add new hardware.

4.4 Estimating RAM

UNIX and Windows-based operating systems differ in how they use memory during run time. In UNIX, memory allocated to a process is not released to the operating system until that process terminates. Under Windows, there are mechanisms that return some memory to the OS before the process terminates.

These different memory allocation models, in addition to the unique memory allocation schemes of each OS, make it difficult to interpret absolute memory requirements. Because of the virtually infinite variety of types of data and processing requirements, it is difficult to exactly estimate the maximum amount of memory required by specific eBI Suite components for the data that will be flowing through them.

Best Estimate: To arrive at the best RAM estimation, calculate the minimum of RAM consumed by a given eBI Suite deployment. Because of the way each OS allocates memory, an absolute or maximum RAM requirement is extremely hard to determine.

4.4.1 Preliminary Estimates

To arrive at a recommendation for total RAM installation for a given eBI Suite deployment, assemble the following information:

- **Number and type of executable components:** In the eBI Suite, there are, for example, BOBs, e*Ways, Registry daemons, a Control Broker, e*Insight Engines, and IQs. Further, some of these components may be Java-based, C-based, or based on Monk, SeeBeyond's script language.

  For lists of current SeeBeyond products, see the SeeBeyond Web site. **Chapter 2** of this guide lists and explains the eBI Suite product components.

- **Size of the files corresponding to components:** For the files installed by individual e*Gate components (for example, e*Ways) and their memory requirements, see the appropriate user's guide for that component or the user's guide that explains the component. The installation guides for e*Xchange, e*Insight, and e*Gate give total memory requirements for the programs and their individual parts.

- **Size of the Monk environment:** In the case of Monk-enabled components, a separate calculation must be performed. The method of calculation is explained in **"Monk Environment Calculation" on page 64**.

4.4.2 Monk Environment Calculation

Following are the distinct processing paradigms in Monk that require special consideration to determine the actual RAM requirements of any given eBI Suite configuration:

- Parsing and population of Monk Events
- Interpretation of Monk code

Parsing and Population of Monk Events

The parsing and population of a Monk Event is usually performed by a Monk Collaboration processing an inbound Event. After Monk business rules are applied to the inbound Event, an outbound Event is created and returned to the process that called the Collaboration Service. Up to five instances of the data could be allocated in this process.

Inbound Event Map Structure

During the processing of a Collaboration, a map structure is created to represent the internalized ETD representing the inbound Event. This map is the result of processing an **.ssc** file. This is the map into the actual Event data. This map is created once and is reused for the life of the Collaboration in its particular Monk engine.

The RAM requirement is dynamically determined as the Collaboration reads in the **.ssc** file and interprets the definition contained within it. Each time the processor

determines it has read a node, it allocates 144 bytes (132 bytes for node information, 12 bytes for system allocation, and 4 bytes for children pointers). A mechanism is required to track all the child nodes per parent, and this mechanism requires memory.

Use the following formula to calculate the size of the map structure:

map_structure_size = [#_of_nodes x (132 + 12 + 4)] + (#_of_nodes_w/children x 12)

Inbound Event Data Tree

Each time an Event is parsed, a data tree is created. This data tree points to data in the inbound data buffer. To determine the size of a data tree, the following guidelines must be applied:

- Each data node is 28 bytes plus 4 bytes for each child node. To simplify matters, allocate the space required by the parent node to the child node. The result is that each node is 32 bytes (28 plus 4).

- An internal heap management system is used by the parsing process to reduce the overhead required to allocate the space for the Event. Memory is allocated for nodes in blocks of 1000. Each of these blocks of 1000 nodes is allocated an additional 12 bytes for management. The memory allocated by the system is never freed, as it is assumed that additional inbound Events will contain similar numbers of nodes.

Use the following formula to determine the size of the data tree:

data_tree_size = (maximum_#_of_nodes x 32) + [(#_of_nodes / 1000) x 12] +
(#_of_nodes_w/children x 12)

Note: *The maximum number is used here because the Events to be parsed can be dynamic and vary per instance. Also, if the result of the equation (#_of_nodes/1000) is less than 1.0, you must subtract 1.0 from the result.*

Outbound Event Map Structure

The map of the outbound Event is created identically to the inbound Event map. Use the same method to calculate the size for the outbound Event as you used for the inbound Event.

Outbound Event Data Tree

When an outbound Collaboration is processed, the size of the outbound Event cannot be determined. However, a tree can be allocated based on the nodes in the **map_structure** that are required and have a minimum repetition greater than one. If the minimum and maximum repetitions are equal, space for the repeated nodes can be allocated. Otherwise, the space for the nodes is created as the nodes are written to.

As data is written to the outbound Event, space for that node is dynamically allocated. This operation allocates space equal to the size of the node plus 12 bytes for system tracking. Each time another repetition is added or an existing field is concatenated to an existing node, a **realloc** operation is executed.

You can use the inbound formula to determine the size of the outbound data tree.

Note: *To allow for the dynamic and unpredictable nature of the outbound data, dynamic allocations are used. Consequently, the allocation process can become time intensive. Memory can also become fragmented because of the number of **realloc** calls.*

Interpretation of Monk Code

In addition to the resources allocated during the parsing and population of Monk Events, space is also allocated for the interpretation of Monk code. Also note that some Monk-enabled components, like e*Ways and the Alert Agent, may not utilize Collaborations, but they still have Monk environments.

Each Monk object is allocated 16 bytes for control information. If the object is not a mediate type like an integer or character, another 88 bytes are allocated for special control information. This control space is allocated in groups of 1000 bytes. The actual value or data of each non-mediate type is stored via a special lookup table that requires 12 bytes plus the size of the data per object.

To arrive at the total RAM required, aggregate the requirements of all the Monk objects, including procedural code that is interpreted into the objects (includes the standard Monk libraries as well as user-defined ones).

4.4.3 Total Memory Requirement Estimation

After you have determined the total Monk environment size, estimating the total memory requirement is a simple task, as explained in this section.

Determine the number and size of non-Monk components as explained in **"Preliminary Estimates" on page 64**. This preliminary estimate is the sum total of the sizes of all files corresponding to all non-Monk components. Run-time observations of the actual RAM consumed can help you refine these numbers.

The total memory requirement is the sum of the preliminary estimate and total Monk environment size. Each specific operating system supported by e*Gate has its own run-time RAM requirements as well as any other application that will be run simultaneously on the system with e*Gate. These requirements need to be added to the e*Gate requirement to get the overall system RAM requirement.

Note: *This calculation will have to be done for each Participating Host defined in the production schema. Empirical observations have suggested that Participating Hosts operate more efficiently if at least 256 MB of RAM per CPU are available on multiple-CPU systems. This requirement supersedes any RAM calculation that yields a smaller amount.*

Finally, this calculation only gives the minimum amount of RAM needed. The actual amount depends on the OS and its memory allocation scheme, as explained under **"Estimating RAM" on page 63**.

4.5 Hard Disk Estimation

Estimating the eBI Suite required free hard disk space can be divided into two requirements:

- Component storage

▪ Operational data

4.5.1 Component Storage

The space required for the actual installation and for the corresponding configuration files, must be estimated in terms of workstations and hosts. A safe estimate is at least 4 GB (4000 MB) per workstation. The Registry Host requires at least 500 MB, and each Participating Host also requires 500 MB. The hard disk requirements for the eBI Suite's configuration files are hard to predict, but given the cost of the medium, 4 to 8 GB would be an appropriate start.

4.5.2 Operational Data

In terms of operational data, the following must be accurately determined for *each* IQ:

▪ Maximum number of Events, including active and journaled Events

▪ Average Event size

▪ Number of components publishing data (publishers)

▪ Number of subscribers that receive data (subscribers)

▪ Log file requirements

Publisher and subscriber configuration data each utilize 256 bytes of hard disk space.

Log File Requirements

The hard disk space required for log files is an important consideration. The estimation of the disk requirements for log files is very specific to each component that is creating a log file. Each component has its own unique number of messages that can be written to the log file, and users can add to that total by writing out messages from their Collaborations and functions.

Logging for any one component can range from almost nothing to over 180 MB per hour, with all debug messages turned on. Do live monitoring of the size of log files, with the debug flags that you assume to be the ones most commonly used, and project the size of the file before it is archived.

Estimating Operational Data Space Requirements

To estimate operational data hard disk space requirements

1 Multiply the number of publishers by 256 bytes.

2 Multiply the number of subscribers by 256 bytes.

3 Add the results of step 1 and step 2.

4 Add the average Event size to the sum resulted in step 3.

5   Multiply the result in step 4 by the maximum number of Events.

6   Add the log file requirements to the result of step 5.

Express all answers in bytes.

Note: *The previous steps must be completed for **each** IQ.*

The total operational data space requirement is the sum of the requirements for all IQs.

4.5.3 Total Disk Space Requirement Estimation

To estimate the total free disk space, use the following calculation:

The component storage requirement plus the operational data requirement

Additional space may be required for log files, but the requirement varies, depending on logging levels and the number of components writing log entries. A safe estimate is an additional 1 GB (1000 MB).

4.6 Configuring for Performance Optimization

The main intent of considering CPU, RAM, and hard disk configurations is to optimize performance of the eBI Suite. This section briefly offers suggestions for optimizing the use of resources discussed earlier. These suggestions will consider the following:

- Efficient use of the eBI Suite

- Parallel processing

- IQs and IQ Managers

- Monk code

- Hard disk space usage

4.6.1 Increasing Efficiency

The premise of the eBI Suite is the efficient processing of data and Events. As such, we recommend that Events are parsed only when absolutely necessary.

When appropriate, pre-format data files prior to sending this information to e*Xchange, e*Insight, or e*Gate. This will save considerable processing time and requirements. Also, consider consolidating the functionality of Collaborations. For example, you could combine the functions of different Collaborations into a single Collaboration.

As log files rapidly grow in size, it is recommended that you lower the journal expiration times. See the *e*Gate Integrator System Administration and Operations Guide* for details about log files and log file maintenance.

There are technical details that are discussed later in this guide (see **Chapter 5** and **Chapter 6**), but briefly stated, it is highly recommended that you employ subscriber

pooling, and that Event flows are segmented. For more advanced techniques, you can consider creating parallel routes and segmenting Event flows.

4.6.2 Optimizing IQs and IQ Managers

It is recommended that you use no more than a total of 50 different publisher and subscriber Collaborations per IQ and three IQs per IQ Manager. However, to maximize efficiency, it is advisable that you use only two IQs per IQ Manager, if possible. For extremely high throughput, you may consider using only one publisher and one subscriber per IQ.

The frequency with which a subscribing Collaboration queries an IQ greatly impacts system performance and may significantly decrease processing speed. For that reason, the polling frequency of each subscriber to an IQ must kept to a minimum.

Perform a clean-up of each IQ after it handles every 1000 Events or after every 10 min of elapsed time, whichever happens first.

4.6.3 Monk Functions

To increase processing speed, minimize the number of times an Event is parsed, or reduce the number of calls to **iq-put** and **iq-get** (Monk functions). For example, multiple data transactions can be bundled into a single Event (see **"Event Parsing" on page 92**). Test the efficiency by first bundling 100,000, then 50,000, and finally 200,000 data transactions.

When necessary or more efficient, rewrite Monk functionality as C functions, either within **.dll** files or in Collaboration scripts. For more information on Monk functions, see the *Monk Developer's Reference*.

4.6.4 Hard Disk Access

To optimize hard disk access times, observe the following general guidelines:

- Use nonvolatile caches on disks.
- Place IQ data on high-RPM disks.
- Use a multi-controller system with many disk partitions.
- Deploy IQs across multiple partitions.

4.7 e*Insight, e*Xchange, and e*Xpressway Requirements

This section lists the basic requirements for each of the e*Insight, e*Xchange, and e*Xpressway systems. For convenience, this section lists software, as well as hardware requirements.

4.7.1 e*Insight

The corresponding version of e*Gate must be installed before installing e*Insight. For example, e*Insight version 4.5.2 requires e*Gate version 4.5.2.

Hardware Requirements

The e*Insight GUIs must be installed on a Windows NT or 2000 workstation that includes the following minimum requirements:

- Pentium-class CPU, 300 MHz or higher
- 128 MB RAM
- 70 MB disk space
- 1024 by 768 pixel monitor resolution

Software Requirements

e*Insight requires that the following applications and components be installed before installing the e*Insight components:

- e*Gate version 4.5.2

Note: *For detailed information on e*Gate and other requirements, see the **e*Insight Business Process Manager Installation Guide**.*

- One of the following applications:
  - Oracle 8.1.6 or 8.1.7
  - Sybase 11.9
  - SQL Server 7 or SQL Server 2000

Note: *For sizing requirements associated with databases, see the user guides for the desired application. For complete details on e*Insight system requirements, see the **e*Insight Business Process Manager Installation Guide**.*

4.7.2 e*Xchange

The corresponding version of e*Gate must be installed before installing e*Xchange. For example, e*Xchange version 4.5.2 requires e*Gate version 4.5.2.

Hardware Requirements

The e*Xchange GUIs must be installed on a Windows NT or 2000 workstation that includes the following minimum requirements:

- Pentium-class CPU, 300 MHz or higher
- 128 MB RAM
- 70 MB disk space

Software Requirements

e*Xchange requires that the following applications and components be installed before installing the e*Xchange components:

- e*Gate version 4.5.2

Note: *For information on the required e*Gate components, as well as detailed information on other requirements, see the **e*Xchange Partner Manager Installation Guide**. This guide also lists all the platforms that e*Xchange supports.*

- Oracle 8i (Server, 8.1.6 or 8.1.7), SQL Server 7.0 (Server), SQL Server 2000, Sybase 11.9 (Server), or DB2 Universal Database
- Java Runtime Environment for Java 1.3.x (for the Validation Rules Builder)

Web Interface

- Java SDK version 1.3.x.

Note: *For sizing requirements associated with databases and other programs, see the user guides for the desired application. For complete details on e*Xchange system requirements, see the **e*Xchange Partner Manager Installation Guide**.*

4.7.3 e*Xpressway

The corresponding version of e*Gate must be used with e*Xpressway. For example, e*Xpressway version 4.5.2 requires e*Gate version 4.5.2.

Solution Provider

- Local installation of e*Xpressway Integrator Packager Tool
- Version 4.5.2 of the SeeBeyond eBI Suite
- A high-speed TCP/IP network connection

Trading Partner

Hardware:

- Dedicated, clean system
- Intel Pentium III or equivalent AMD Processor, 800 MHz minimum
- 512 MB RAM
- 1 GB free disk space for installation and operational data

Software:

- Windows 2000 with Service Pack 2 or Windows NT 4.0 with Service Pack 6a or later
- Microsoft Internet Explorer 5.0 or later
- A file archive utility, such as WinZip, Jar, or the equivalent

The trading partner also needs a TCP/IP network connection and a highly available, high-speed Internet service (for example, T1 or DSL).

Note: *Clean means a system with no applications other than an OS running on it. For trading partners, do not install OnRamp on any system that already has e*Gate installed.*

Trading Exchange

Hardware:

- Same as the trading partner

Software:

- e*Xpressway Integrator Server, version 4.5.2
- e*Gate version 4.5.2
- Windows 2000 with Service Pack 2 or Windows NT 4.0 with Service Pack 6a or later
- Microsoft Internet Explorer 5.0 or later
- Java Development Kit (JDK) Release 1.3 or later
- HTTPS, CGI, and Batch e*Ways
- Oracle 8i database
- Oracle SQL client on the installation system

The Trading Exchange also needs a TCP/IP network connection and a highly available, high-speed Internet service (for example, T1 or DSL).

4.8 System Requirements: Summary

This section summarizes eBI Suite deployment system requirements.

Registry and Participating Hosts

SeeBeyond recommends the following minimum Registry and Participating Host hardware requirements for a typical current e*Gate configuration of 20 to 30 interfaces:

Windows NT

- 1 GB (1000 M) RAM
- 20 GB disk space
- Pentium III-class CPU, 866 MHz

UNIX

- 1 GB RAM
- 20- to 30-GB hard disk space
- CPU, 400 to 450 MHz

> *Note:* *A second CPU is recommended if you are using the SeeBeyond Standard*
> *IQ Manager (that is, if you are **not** using the SeeBeyond JMS IQ Manager or*
> *MQSeries JMS e*Way Connection).*

Throughput performance can vary depending on the size and complexity of data. Additional memory or CPUs may be necessary, depending on the specific throughput requirements. e*Gate also needs a TCP/IP network connection.

Client Systems

SeeBeyond recommends the following as a comfortable configuration for client (Windows NT for GUI) systems:

- 512 MB RAM (minimum)
- 8-GB hard disk space
- Pentium III-class CPU, 700 to 866 MHz
- 19-in. monitor

> *Important:* *You must install the e*Gate GUI applications on a Windows system to configure*
> *and monitor UNIX e*Gate components. For a complete list of e*Gate system*
> *requirements (hardware and software) and supported platforms, see the **e*Gate***
> ***Integrator Installation Guide**.*

Additional eBI Suite Applications

Take into account the system requirements for e*Insight, e*Xchange, and e*Xpressway (see **"e*Insight, e*Xchange, and e*Xpressway Requirements" on page 69**) if you are using any of these applications.

Going Forward

Once you have finished all elements of your deployment planning, including having a complete determination of your hardware requirements, you are ready to go on to the system design and development phase of the project.

System Design and Development: For a discussion of the next phase your deployment project, including system design architecture and development, see **Chapter 5**. In this phase, you broaden and fill in the details of the general designs and overall model you created during planning.

The next chapters in this guide treat the eBI Suite deployment project under the following topics:

- **Testing and Transition to Production:** For a discussion of the testing, transition (go-live), and maintenance (fine-tuning) phases of your system deployment, see **Chapter 6**. These phases follow the system design and development phase.

- **Helpful Tips: Chapter 7** "Frequently Asked Questions" gives you some helpful hints and tips for best practices.

Designing and Developing the eBI Suite Environment

This chapter explains how to design and develop a complete, functioning eBI Suite based on your deployment analysis and planning.

In This Chapter

- **"An Overview of eBI Suite Design" on page 74**
- **"Distributed Architecture Considerations" on page 76**
- **"Methodology Considerations" on page 84**
- **"Designing Your System" on page 96**
- **"Optimizing Your System" on page 104**
- **"System Development Considerations" on page 119**
- **"Case Study Examples" on page 133**

5.1   An Overview of eBI Suite Design

After the analysis and planning phase has been completed, your next major step is the system design and development phase. In many ways, this work is the heart of the eBI Suite deployment operation. During this phase, you flesh out the essential system architecture that implements your business plans and processes.

Designing the deployment of an eBI Suite environment requires a series of successive refinements applied to your initial summary plan ("Functional Requirements Specification" document as outlined in **Table 2 on page 54**). Your design must start with the broadest view of the system then proceed to the details.

Applying this top-down approach to deploying an eBI Suite environment results in the most effective application of its technology to the integration of your existing systems and applications. Total system design and development include the following basic steps:

- Planning general hardware configuration
- System design methodology
- Software installation and development

- System optimization

However, keep in mind that these steps are not necessarily an exact sequence. The entire design and development operation requires that you occasionally "back-track" to earlier steps and look forward to later steps, to implement the correct design refinements your system requires.

Figure 11 shows where the system design and development phase fits into the overall eBI Suite Deployment operation.

Figure 11 System Design and Development Phase

Phase 1:

Analysis of Requirements

Phase 2:

Development Planning

Phase 3:

System Design and Development

Phase 4:

Pre-Transition Testing

Phase 5:

Transition to Production

Phase 6:

Post-Transition Maintenance
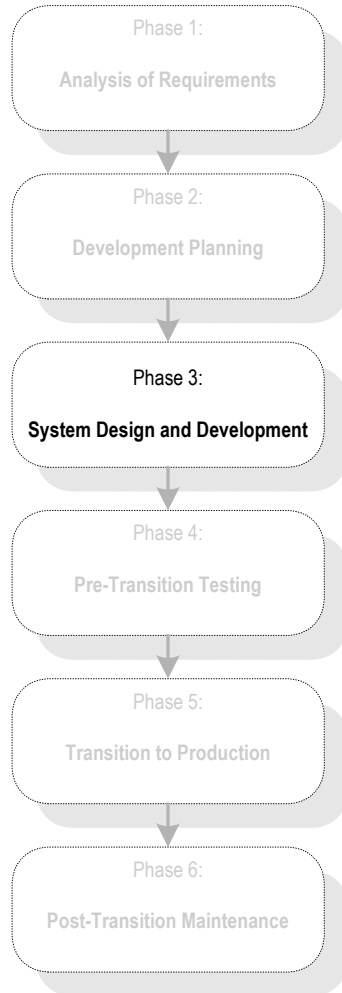
System Design

Because you can distribute a single eBI Suite environment over as many hosts as you need to provide sufficient computing power, this chapter guides the decisions you must make to deploy an effective e*Gate environment, including:

- Choice of the number of hosts to employ

- Choice of the number and types of schemas and components to build

The chapter also presents a methodology for designing an eBI Suite environment. The methodology involves the following well-defined steps:

- Describing the communication topology

- Designing the hardware topology

- Designing the component topology

- Planning the eBI Suite components

- System optimization

System Development

Development proceeds after completion of design tasks and consists of a list of tasks to create each component of the eBI Suite. The section on development explains how you create the task list, including the completion order for the tasks.

The entire system design and development phase methodology is illustrated in later sections with examples for the following SeeBeyond products:

- e*Gate

- e*Insight

- e*Xchange

- e*Xpressway

Case Study Examples

- **"Case Study 1: Web Order Scenario" on page 133**

- **"Case Study 2: Expanded Web Order Scenario" on page 139**

- **"Case Study 3: Tracking Timecards and Payroll Scenario" on page 144**

- **"Case Study 4: Receiving and Purchasing Scenario" on page 149**.

5.2 Distributed Architecture Considerations

The power of the eBI Suite lies in its fundamental design that includes:

- Distributed architecture

- Central management of computing

- High availability and firewall management

This section explains e*Gate's distributed network architecture and how to take advantage of its specific features in your deployment.

5.2.1 Distributed Architecture in e*Gate: Overview

A common view of software systems starts with a box representing a computer host. Programs or processes are added to the computer host and are represented as smaller boxes inside the bigger box.

Multiple software systems are typically spread out over several physical hosts with no relationship or connection between the hosts. Figure 12 shows the conceptual

relationship among several different software systems that are commonly built to support business needs.

While it is possible to connect many different types of systems such as those in Figure 12, it is inconvenient and costly to manage the connections without a central point of access.

In addition, economies of scale gained through reusable components are unlikely to exist in the typical hub-and-spoke architecture that these types of systems require.

Figure 12   Common View of Software Systems
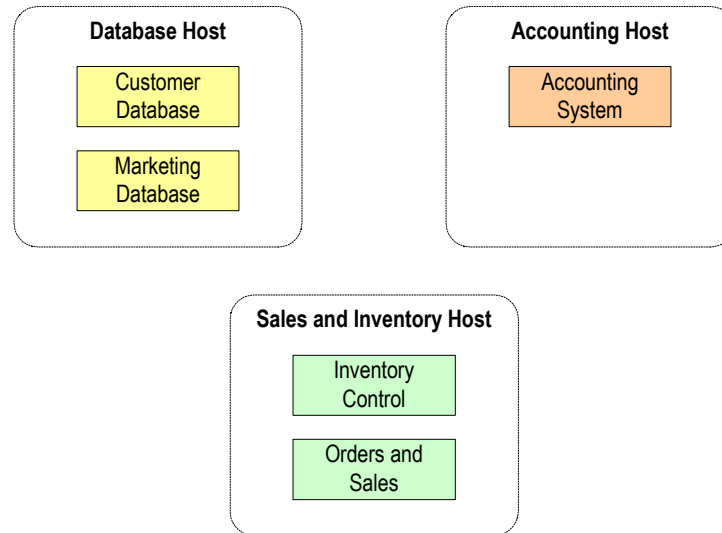
The eBI Suite turns this typical view around. The eBI Suite can be understood as encompassing all participating computer hosts. From this viewpoint, the eBI Suite becomes the connection that brings many disparate computer hosts and processes together.

As a result, diagrams describing a deployment of the eBI Suite show the system as a large box encompassing the systems that it connects.

Although the eBI Suite is represented as a large box, this portrayal is not meant to suggest that the system runs on its own dedicated host. The power of the eBI Suite is that its components can be distributed over several hosts as needed. The components communicate with each other and with a GUI that provides a central point of access to an integrated system.

Using the e*Gate environment as an example, Figure 13 shows computer hosts connected through the eBI Suite as boxes *inside* the main box.

Figure 13 e*Gate Distributed Environment
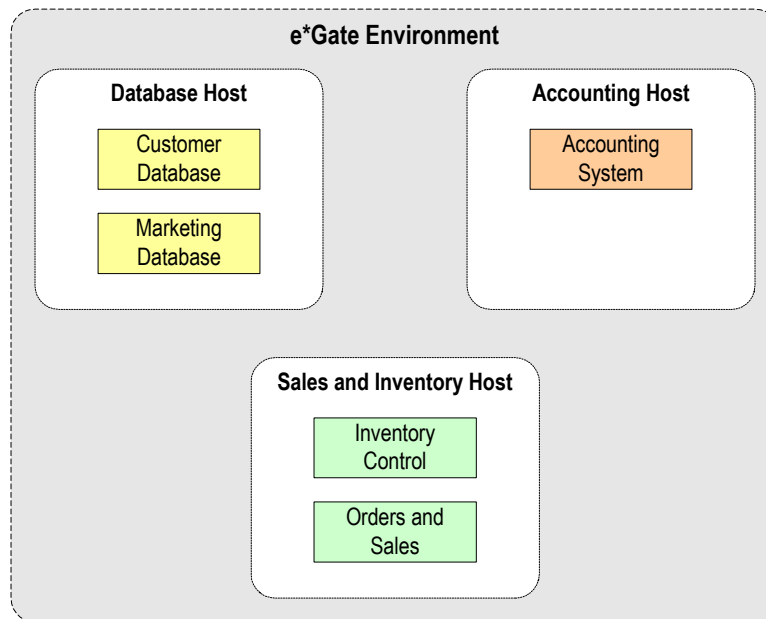
Multiple Participating Hosts

In e*Gate, the various components are managed in this distributed-network environment. All system components, including Control Brokers, have logical names that are independent of physical host (computer) location. At the configuration level, the system's publish-and-subscribe information is dependent on logical names, not host names.

Participating Hosts have one property that sets the host name. e*Gate components find each other purely through host names and port numbers. As a result, it is easy to reassign host names or ports, or move e*Gate components to other systems without any change to the basic e*Gate configuration.

5.2.2 Basic Architecture

You can scale an existing eBI Suite environment containing e*Gate simply by adding more memory, processors, and computer hosts to the total system. Any of these actions results in incremental benefits. Two examples are:

- If your company acquires a new business unit and needs to integrate its systems to an existing configuration, you network these systems to the existing e*Gate hosts and add new components to service the acquired systems.

- If your business experiences growth in computer traffic, and you need more computing power to service it, you can add another processor to an existing host. Also, you can add one or more hosts then move or duplicate some of the existing components to any new host.

In both of the previous examples, the existing e*Gate components do not change. As you read the diagrams in this chapter, understand that e*Gate represents the "big box." One or more computer hosts are shown as smaller boxes in the big e*Gate box.

For example, you can configure Participating Hosts to refer to *any* Registry Host, primary or secondary, as your installation requires. Also, you can point multiple Participating Hosts to the same Registry.

Schema and Component Organization

The eBI Suite components are organized into schemas. A schema is a configuration scheme that contains all the modules and configuration parameters that control, route, and transform data as it travels through the overall system.

A schema also maintains the relationships between its internal components, including the publish/subscribe information that is at the heart of e*Gate's data transportation process.

The number, design, location, and component makeup of your schemas is a function of your overall design methodology, as explained under **"Methodology Considerations" on page 84**.

Role of Control Broker

The Control Broker component manages schema operation in e*Gate. An e*Gate environment requires that you can run only one Control Broker per host per schema, and the Enterprise Manager enforces this restriction. However, a single host can support multiple schemas and run more than one Control Broker.

5.2.3 High Availability Features

You can minimize loss of critical operations by distributing the operations over multiple systems. Using this setup, you can design the eBI Suite so that if one system has any problems, another can take over the critical operation entirely.

This section provides a summary of how these features operate. For more information on implementing high availability in e*Gate, see **Chapter 8**.

*Note: e*Gate has not been fully tested for UNIX high availability products.*

System Registry

The e*Gate Registry holds all the necessary information for every component in the system to run. All Participating Hosts authenticate with the Registry through the schema name and Control Broker's logical name. As a result, all e*Gate Participating Hosts have the potential to run any e*Gate Control Broker and schema.

The key idea is that no configuration is tied to the files on the Participating Host. Even if you lost all systems in a given environment except the Registry Host, you could install new systems as Participating Hosts and then give them the names of the original systems or change the host names configured in the e*Gate schema. This would reproduce all the original configurations by connecting to the Registry through the same logical component names.

Registry Replication

You can install the e*Gate Registry in a replicated configuration. To do so, designate a primary Registry that sends all schema changes to one or more secondary Registries. Also, you can assign the e*Gate Control Brokers and other components to more than one Registry.

The components attempt connecting to the first Registry in their Registry list. If this action fails, the components attempt the next Registry in the list. If the Registry processor is not on the same system as any Participating Host, you can use this feature to reduce the Registry connection as a point of failure.

This architecture allows you to employ a distributed Registry with Registry Replication for added system redundancy and software-based high availability capability. **Figure 14 on page 80** illustrates a typical eBI Suite network with these features.

Note: *For more information on Registry replication and the distributed Registry, see the* ***e*Gate Integrator System Administration and Operations Guide***.

Figure 14   Overview: e*Gate Network with Distributed Registry
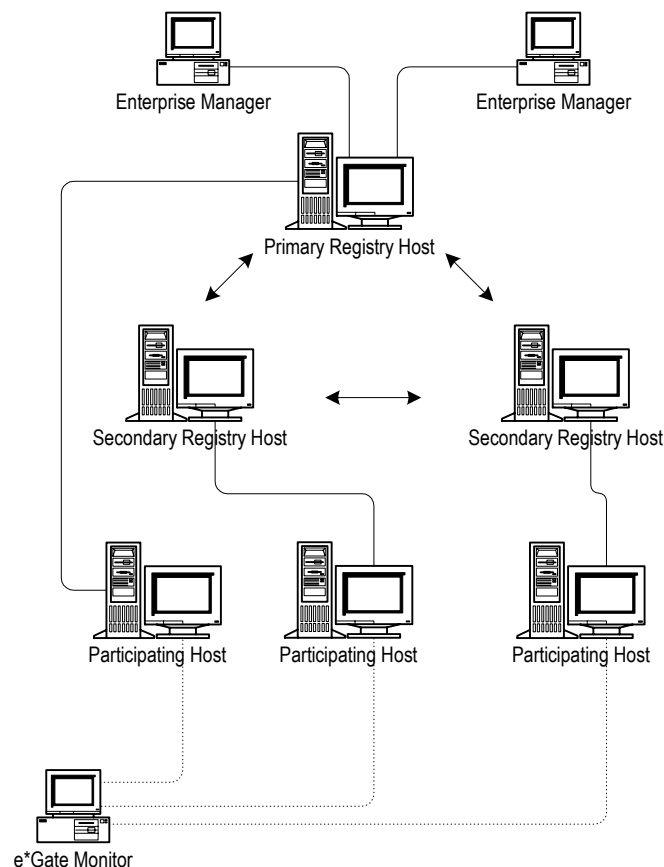
Clustering in e*Gate

You can also take advantage of Microsoft clustering software in designing your system. You may want critical operations performed by hardware to be backed up by additional

systems, where necessary. See the **Appendix D** for an explanation of how to design and use this clustering software.

Note: *For general information on high availability issues involved in upgrading from DataGate (e*Gate 3.5/3.6) to e*Gate 4.0 or later, see* **Appendix C***. For more information, see the* **e*Gate Integrator Upgrade Guide***.*

5.2.4 Network Port and Firewall Considerations

e*Gate components exchange data and control information using a TCP-based protocol. This transport requires reservation ports on the systems where the components reside. Table 5 shows the default listen ports used by e*Gate components.

Table 5   Default e*Gate Port Usage

Component	Default Port
Control Broker	4000
Registry	23001 23002 23101
IQ Manager	24053 (+ if other ports)
SNMP Agent	1501

The reason there are two ports that connect to the Registry is similar to the NETD process on UNIX (see the appropriate UNIX user's guides for details). That is, if one of the Control Brokers fails during the initial connection to the Registry, it may cause the initial connection port (23001) to hang.

If, during that time, the Registry goes down, it cannot come back up because the initial port it attempts to bind to is unavailable. This problem occurs because the malfunctioning Control Broker process has control of that port.

The secondary reason for the two-port connection Registry is for scaling, particularly, if in the future, the e*Gate environment can dynamically allocate ports, for example, if the system has more than 1000 components.

All other e*Gate components initiate TCP connections and thus do not need a reserved number. Figure 15 shows components of e*Gate and how they are governed by the Registry or Control Broker.

Figure 15 e*Gate Component Relationships
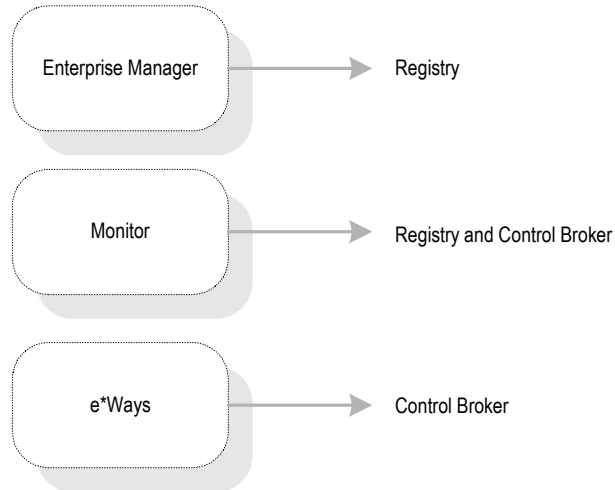
Figure 16 shows a typical firewall configuration.

Figure 16 Communicating Through a Firewall
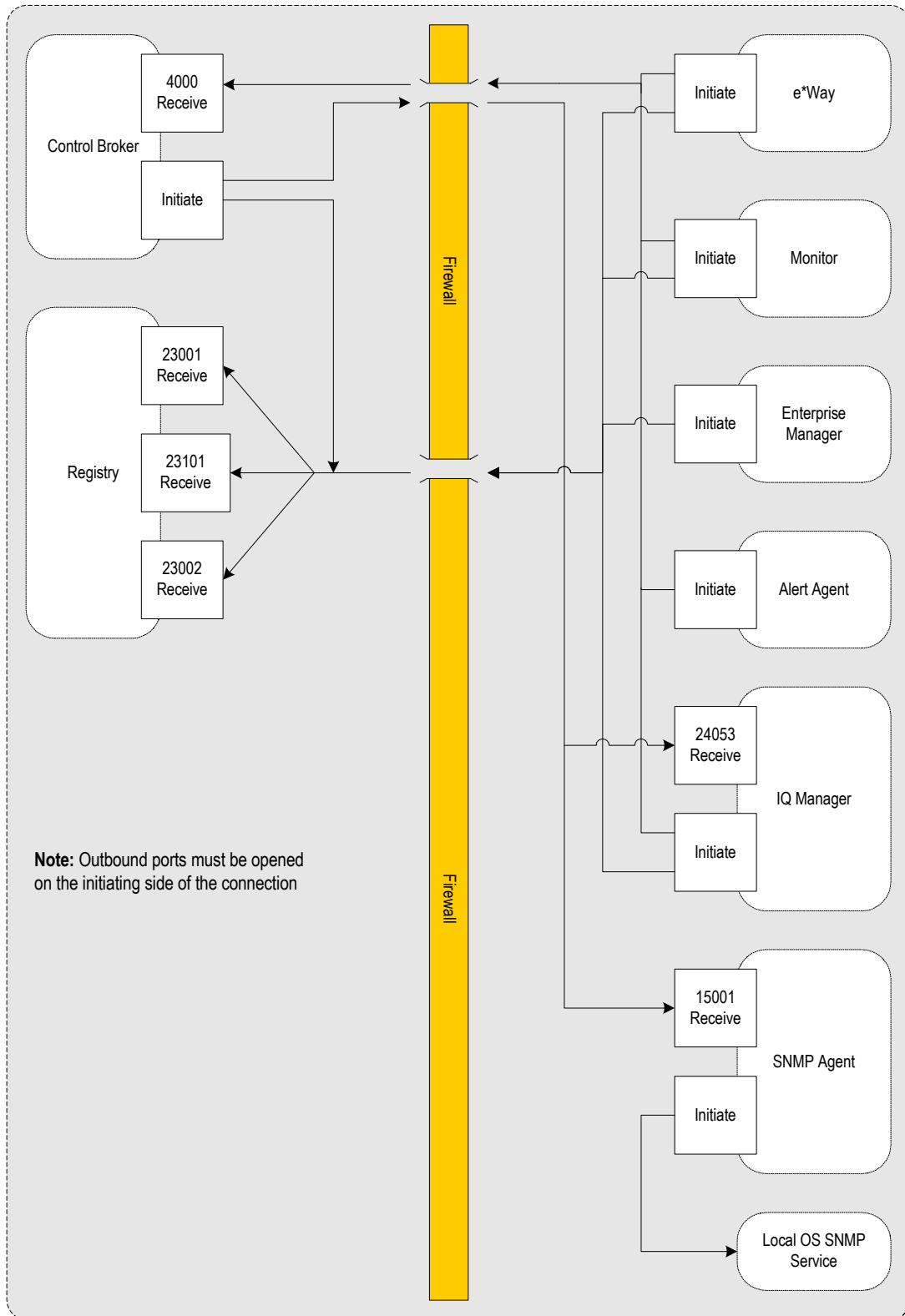
5.2.5 Clustering and Storage Area Network Considerations

The logging features of certain network volume management products (such as *VERITAS Volume Manager*™) have a negative impact on IQ performance. It is advisable to turn these logs off for volumes that host IQ files.

5.3 Methodology Considerations

Methodology means the ways or methods to figure out how to design the eBI Suite to best meet your business and information system (IS) needs.

5.3.1 What is Topology?

Topology refers to the pattern of connections between interrelated objects. Topology considers only relationships between objects and ignores the location of the objects.

Because the eBI Suite is centrally managed, the location of a Participating Host is insignificant in designing the system. Understanding the meaning of topology is important in providing a conceptual framework for discussing design considerations for your total system.

Elements of Topology

A topology exists between the following elements:

Computer Systems Related by Communication
This is a data-flow topology because the only concern is which system is communicating with which other system. Communications topology is therefore only logical because it has no reality in hardware.

Computer Hosts Related by Physical Network Connections
This is a hardware topology because it concerns the relationship between physical computer hosts.

e*Gate Components Related by Publication and Subscription
This is a component topology because it concerns the pub/sub relationship between components.

Sample Topologies

The figure on this page shows three sample topologies. The first two examples are identical because distance and location are irrelevant in describing a topology. Only relationships matter and the relationships described by the arrows connect the same systems in the same way between the first two examples. The third example is different

from the first two because it contains an additional connection from System C to
System B.

Figure 17   Examples of Topologies
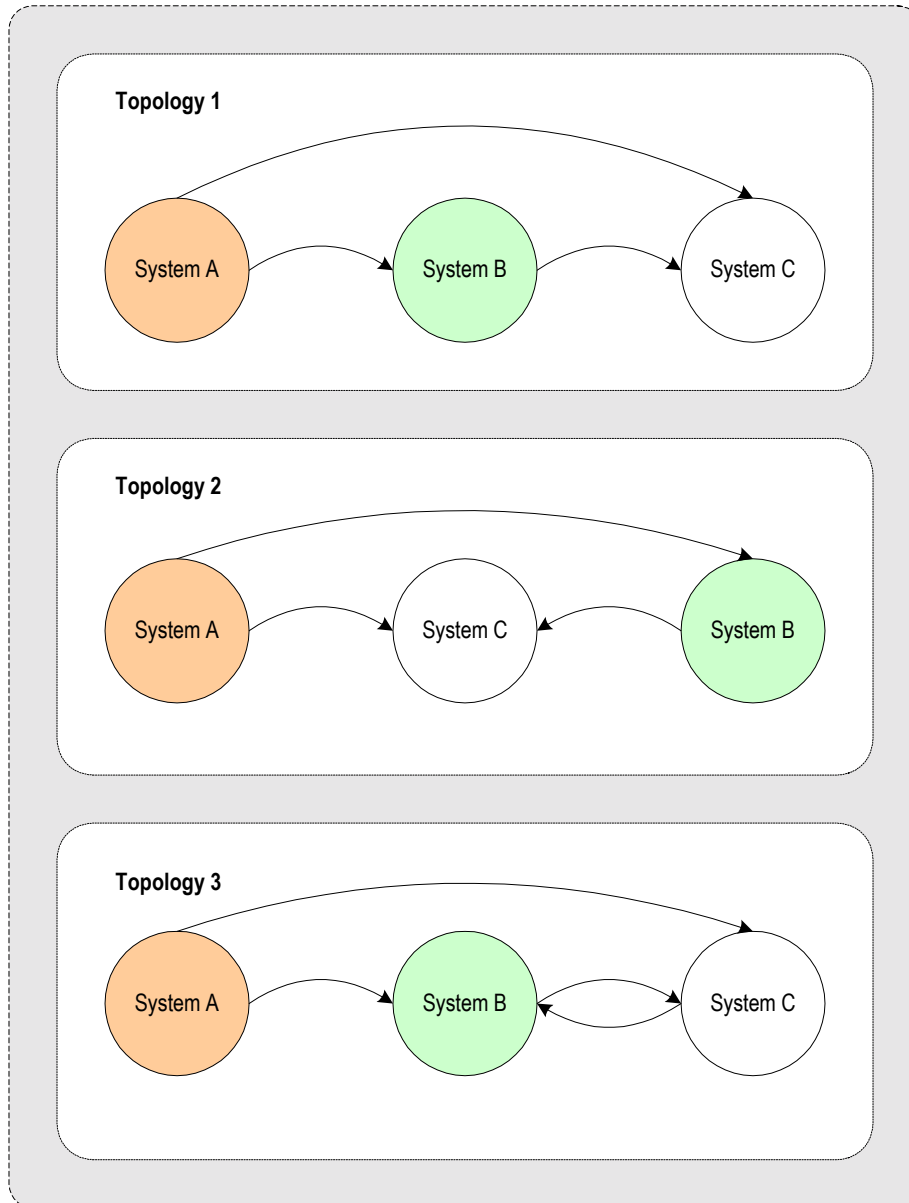
In these diagrams, the direction in which each external system exchanges data with
another is identified with an arrow connecting the data publisher to the data subscriber
(see the previous figure). The arrow points in the direction that data flows.

5.3.2 **Three Basic Steps**

The basic steps in designing an eBI Suite are:

- Identify all external systems to be connected.

▪ Define the configuration of eBI Suite components.

▪ Define the configuration of hardware and network connections.

This section explains these steps in detail.

Identifying External Systems

The first step in designing an eBI Suite is to identify all the external systems to be connected to each other through the system. The resulting set of interconnected systems is called the *communication topology.* The communication topology exists without regard for the hardware hosts where components execute and without regard for the format of data exchanged between systems.

Configuring eBI Suite Components

The second step is to define a configuration of e*Gate components, for example, e*Way Intelligent Adapters, BOBs, e*Insight Engines, and IQ Managers, to run on the respective hosts in the *hardware topology*. The component arrangement is called the *component topology.*

Efficiencies at this stage are gained by choosing the simplest ETDs and Collaborations. Defining the most efficient component topology depends upon the relationship of data formats. Therefore, defining Event Types is an integral part of designing the component topology.

In addition, you also want to determine and define the interaction between e*Gate, e*Insight, e*Xchange, and e*Xpressway, if you are using any of these applications.

Hardware and Network Connections

The third step is to define a configuration of hardware and network connections that enable the external systems to communicate as required by the communications topology. This hardware configuration is called the *hardware topology.*

As explained earlier, the eBI Suite is designed to run as a distributed system with central management. Only network performance and the demands on each host are relevant considerations in defining hardware topology. Because of the distributed architecture of the total system, the hardware topology is not rigidly defined. It can be adjusted as needed when system demands change. For example, increased demands on the e*Gate environment can be met by distributing processing across more CPUs.

For more information on how to determine and meet your hardware requirements, see **Chapter 4**.

5.3.3 Performance Considerations

As with all computer systems, performance improvements can be achieved by optimizing memory usage and computational efficiency. In this section we consider impacts of the following eBI Suite performance-related issues:

▪ Virtual memory

- Event parsing

- Subscriber pooling

- Hardware distribution

Before delving into this level of detail, this section outlines the basic operation of e*Ways, BOBs, and IQs. The rest of this section explains these components then provides some performance and application guidelines.

Basic e*Way Operation

An e*Way is used to communicate between an external system and the rest of the e*Gate environment. An e*Way provides both communication and data transformation capabilities.

Communication and Collaboration Functions

An e*Way is architecturally divided into two parts, the Communication and Collaboration halves. In **Figure 18 on page 88**, the e*Way is represented by a split box. The Communication part is the left half of the box. The Collaboration half is the right half of the box.

These two parts of an e*Way function as follows:

- **Communication half** is responsible for communicating with the external system. Through the use of configuration parameters and user-programmed functionality, the Communication half can be configured to meet any business requirements.

- **Collaboration half** is responsible for exchanging Events with the rest of the e*Gate environment. The Collaboration half can execute one or several Collaborations on any Event. Collaborations are fully configurable and can be as complex or as simple as needed.

The simplest transformation you can use in the Collaboration half is called the *Pass Through Service*. This Collaboration Service receives an Event from, or sends an Event to the Communications half without transforming the Event in any way. Using the Pass Through Service, the Event sent on by a Collaboration is an exact copy of the Event received.

Figure 18 Basic e*Way Operation
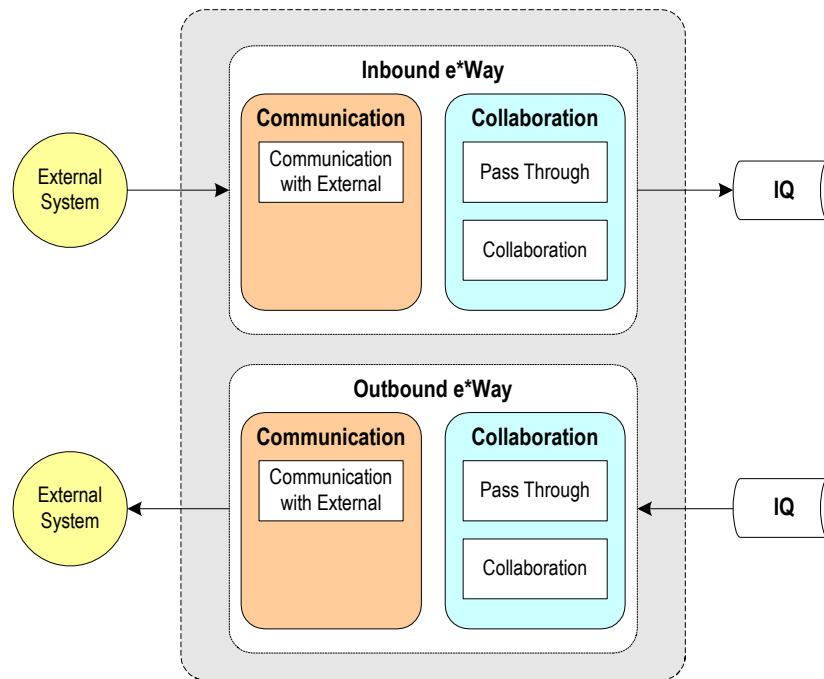
Figure 18 shows both an inbound and an outbound e*Way. In practice, a single e*Way can be used to support both inbound and outbound Events. Generally, it is better design practice to configure inbound and outbound e*Ways.

These e*Way types function as follows:

- **Inbound e*Ways:** For an inbound Event, the Event first passes to an e*Way's Communication half, that is, the e*Way *subscribes* to the external system. A function can execute to transform the Event, after which the results flow to one or more Collaborations in the Collaboration half, or the Event can simply "pass through." The results of a Collaboration are *published* (sent) to an IQ.

- **Outbound e*Ways:** For an outbound Event, an Event is fetched from an IQ and processed by an e*Way's Collaboration half. In the Collaboration half, the Event can be transformed by one or more Collaborations or can go through the Pass Through Service. The result of the Collaboration is passed to the Communication half where it is published to the External system.

The use of IQs guarantees delivery of an Event, or else notification if the Event cannot be delivered. Under no circumstances is an Event ever lost. See **"Basic IQ Operation" on page 89** for more information on IQs and their functions.

Each e*Way has associated documentation explaining how to operate and configure the e*Way. Refer to the appropriate e*Way user's guide for detailed information on configuring an e*Way.

Note: *Subsequent figures provide a simplified version of the e*Way diagrams shown here. Although most diagrams show a single Collaboration, this does not imply a limitation on e*Way capabilities. Many Collaborations are possible.*

Basic BOB Operation

A BOB is used to move data within the e*Gate environment. Unlike an e*Way, a BOB provides only data transformation capabilities. BOBs provide intermediate data transformation capabilities or allow for helping with load distribution. A BOB subscribes to one or more IQs, fetches Events from these same IQs, processes Events through one or more Collaborations, and publishes the result to one or more additional IQs.

Like those in an e*Way, BOB Collaborations are fully programmable and can be as complex or as simple as needed. In **Figure 19 on page 89**, a BOB is represented as a rounded box. Unlike an e*Way, which is architecturally split into two halves, a BOB contains just a single part to execute Collaborations. One or many Collaborations can be configured in a single BOB.

Note: *Subsequent figures show a simplified version of the BOB diagrams in* **Figure 19 on page 89**. *These figures sometimes show a single Collaboration for ease of clarification; this is not be a limitation of a BOB. Many Collaborations or Pass Through services are possible.*

Figure 19   Basic BOB Operation
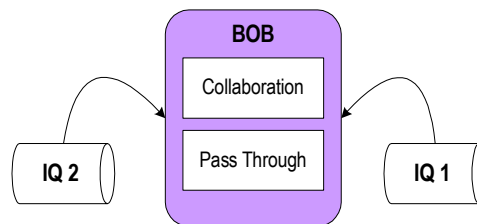
Basic IQ Operation

An IQ is an on-disk repository of transient data. Events written to an IQ stay there until picked up by all subscribers to the same Event Type. Because Events stay in the IQ only until passed on, they are considered transient.

Events sent through e*Gate are guaranteed not to be lost. Either they succeed in arriving at their destination or they are stored on disk while an acknowledgment of the delivery failure is returned. Storing Events in IQs makes this functionality possible.

Note: *In e*Gate release 4.5 or later, you can ensure the once-only delivery of every Event. See* **"Using Guaranteed Exactly Once Delivery" on page 113** *for details.*

Because of this feature, *within e*Gate* every BOB and e*Way must publish to at least one IQ and subscribe to at least one IQ. This architecture provides support for transferring Events internally. Your design considerations determine how many IQs each e*Way and BOB publish and subscribe to.

Also, an IQ can be configured so that an Event fetched by a single subscribing process is removed from the IQ. For more information on this feature, see **"IQ Subscriber Pooling" on page 92**.

Note: *For simplicity, figures which illustrate a configuration of e*Ways or BOBs do not display the IQ unless the IQ is relevant to the performance issue under discussion. However, keep in mind that at least one IQ is always present when a BOB and an e*Way exchange Events.*

Virtual Memory

Each process requires memory for its executable code and for its data. Components in e*Gate that require executable processes include:

- e*Ways
- BOBs
- IQ Managers
- Control Brokers
- Collaboration Services (including Java, C, and Monk)

Note: *In e*Gate, these types of components are also called **modules**.*

Because the entire eBI Suite, including e*Gate, e*Insight, e*Xchange, and e*Xpressway, is distributed across an integrated network of computers, e*Gate memory usage affects the entire system. Extending memory by moving information between the RAM and the hard disk is called *memory swapping*. Memory management for e*Gate processes requires memory swapping in the host computer's RAM, as this section explains.

Memory Allocation

Once loaded, the RAM space required by executable code remains fixed. However, data memory is dynamically allocated and can grow. If a Collaboration processes an Event that is too large for the currently available memory, additional memory is allocated to the process to meet the increased need.

Ideal performance requires that all memory used by all processes fit into physically available RAM. This ideal circumstance is rarely achieved because most systems trade some of their speed for increased flexibility in memory management. This management allows for memory swapping and reserves a region of hard disk storage space to use as an extension of RAM.

Memory Swapping Process

If physical memory available on a computer is, for example, 50 MB but the memory required by a program and its data is 100 MB, the program and data that it operates on cannot fit into the available memory. In this circumstance, the operating system loads just a portion of the program and start execution on a portion of data.

Figure 20 shows a process that uses nine blocks of memory but can only fit three blocks into the RAM.

Figure 20 Memory Swapping
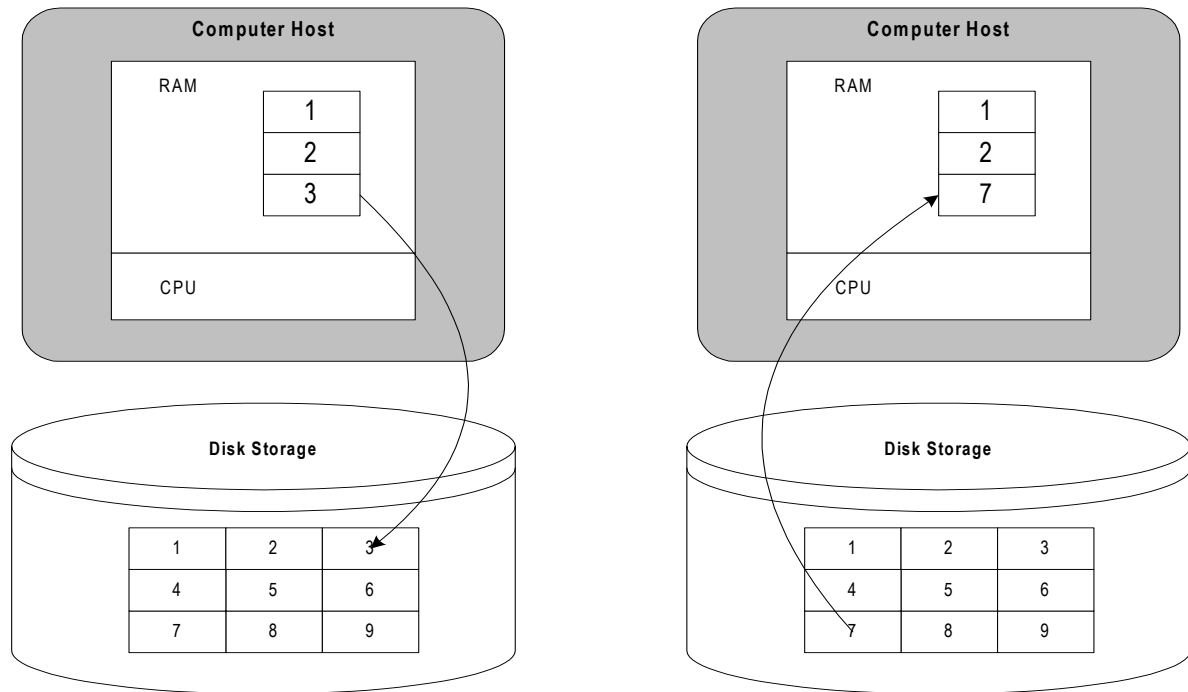
Step 1: Write block 3 to disk   Step 2: Read block 7 from disk

When the computer is instructed to access executable code that does not exist in the portion of code that was loaded, or when the computer must use data has not been read in, the operating system intervenes as shown in the previous figure. It chooses some block of data or code in memory (block 3 in the previous figure) and writes it to the disk, thus freeing up space in memory. It then finds the required block of data or code (block 7 in the previous figure) on the disk and reads it into the newly available memory. This process is memory swapping, and the blocks of data are called "memory pages".

Memory swapping by the operating system is costly in time because accessing the disk is comparatively slow. This swapping can be avoided completely if all code and data fit into the memory available on the host.

Memory Estimates

For performance reasons, memory allocated by the system on behalf of a process is reused by that process as ETDs are freed. On some systems, memory is never released and returned to the system. If more memory is required by a process than the current "high-water mark," additional memory is allocated, and the high-water mark is increased.

Thus, memory demands from an eBI Suite start out at a certain level but increase if large Events are processed. This process simplifies memory management and speeds performance but could require that total, long-term memory usage be well understood and that the configuration be designed to avoid memory swapping whenever possible. When accounting for the space taken by data, you must base your estimates on space determined up to an estimated high-water mark.

Event Parsing

In e*Gate, when a Collaboration Rule script processes an Event, it must first match the contents of the Event to an ETD that describes how the Event is structured. If the Event contains a name field and an address field, for example, the ETD describes where those fields can be found.

Parsing and Collaborations

An e*Gate installation provides extensive libraries of ETDs to describe many standardized Event Types including Events complying with the EDIFACT, X12, and HL7 standards. Events defined in these and other standard schemes can be quite complex and contain hundreds or thousands of individual nodes.

The process of matching an Event to its definition, or *Event parsing*, must take place before individual fields are referenced in a Collaboration Rule. Parsing an Event enables a Collaboration Rule to resolve a path name like:

input%CustOrder.Customer[0].fullname

into the data that is found in that location, for example:

"George Washington"

Matching an ETD to an Event involves inspecting the contents or format of the Event for certain kinds of data or delimiters. If the ETD allows for optional nodes, parsing can be much more complex, because many more possibilities must be considered before determining whether an optional node exists within the given Event.

Parsing and Memory

When parsed, each node within an ETD occupies a certain amount of memory. The memory location holds identifying information about the node and its contents. More nodes in an ETD translate into more computation and more memory usage.

Therefore it makes sense to use an ETD that defines only those fields actually referenced in the Collaboration Rule. If you are using an ETD provided as part of a library, you can copy it under a new name then prune its unnecessary nodes. When deploying e*Gate, create the simplest ETDs needed for each Collaboration Rule script.

IQ Subscriber Pooling

Distributed processing eliminates bottlenecks. Because the eBI Suite is a distributed system, no single process can block the execution of other processes.

However, the component topology in e*Gate is related to the Event Types that must be processed. Typically, most or all Events of a particular type take the same route through the e*Gate environment. If there is an Event Type that is especially complex, and there are many Events of that particular type, there could be times when the processes supporting that route cannot keep up with the demand.

Eliminating Data Bottlenecks

IQ Subscriber pooling solves this problem by permitting routes through duplicate Collaborations, possibly on separate hosts, so that no single route is able to create a bottleneck. The following figure shows how components might be arranged without using subscriber pooling.

Figure 21   Components Without Subscriber Pooling
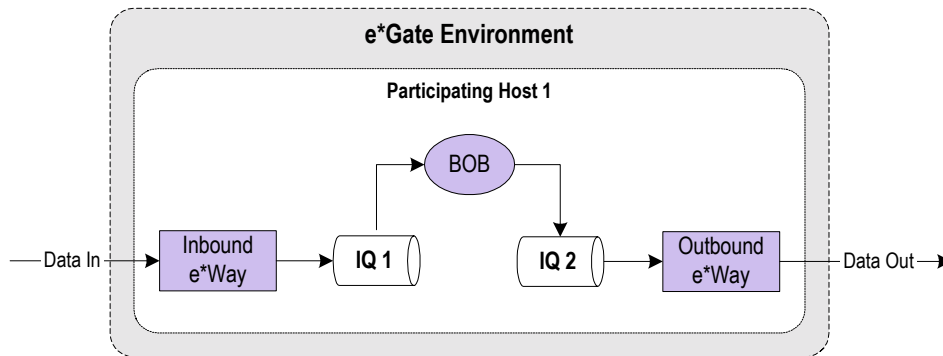
In this arrangement, the inbound e*Way receives data from an external system. It writes Events to IQ 1. BOB 1 fetches Events from IQ 1, processes the Events through its Collaboration and publishes the new Event to IQ 2. The outbound e*Way fetches Events from IQ 2 and publishes them to its external system.

While the arrangement shown in Figure 21 does not restrict the processing of other Events, BOB 1 can cause a bottleneck if many complex, computation-intensive Events must be processed through its Collaboration.

To break up the bottleneck, use IQ subscriber pooling and distribute the processing burden over more than one BOB. When subscriber pooling is used, all BOBs must be configured to execute the same Collaboration. Figure 22 shows subscriber pooling with two BOBs running on the same Participating Host.

Figure 22   Components with Subscriber Pooling
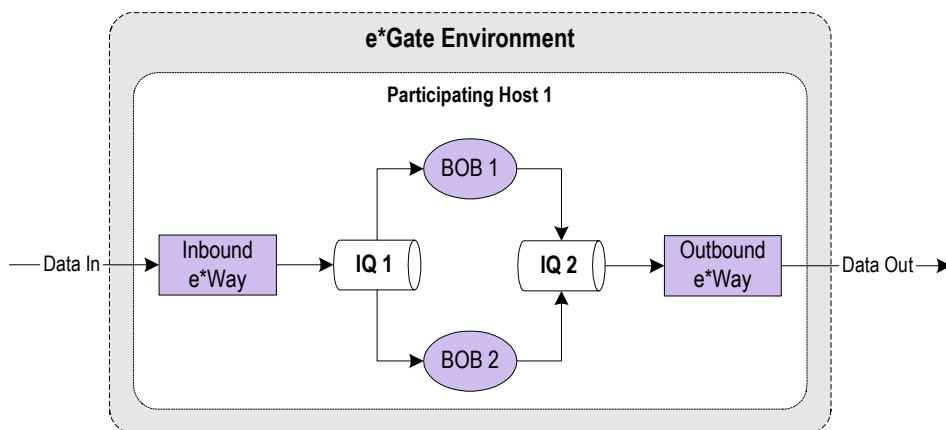
If the reason for the bottleneck in Figure 21 is because BOB 1 must wait for a reply from the inbound e*Way. For example, the e*Way has to query an external database or reference a file at a distant location on a network, and then has to run BOB 2 on the same Participating Host is an effective way to distribute the load. When BOB 1 is waiting for specified responses from the e*Way, BOB 2 runs using available CPU cycles to process other Events.

On the other hand, what if the reason for the bottleneck is because there is not enough CPU power on Participating Host 1 to meet the need? In other words, what if BOB 1 is

not waiting for the inbound e*Way's response to an external system but is forced to wait for CPU cycles *from its own host*?

In this case, adding a second BOB to the same Participating Host does not improve matters because there is no extra CPU power available. Instead, you can add a BOB but have it run on a *different* Participating Host, as Figure 23 shows.

IQ subscriber pooling and distributing components over several Participating Hosts provides the following efficiencies:

- When you use subscriber pooling, you can add as many additional, but identical, BOBs and distribute them over as many Participating Hosts as you need.

- By adding BOBs and Participating Hosts, you can distribute the load and increase the computing power of your e*Gate environment as the need arises.

Figure 23   Subscriber Pooling over Multiple Hosts
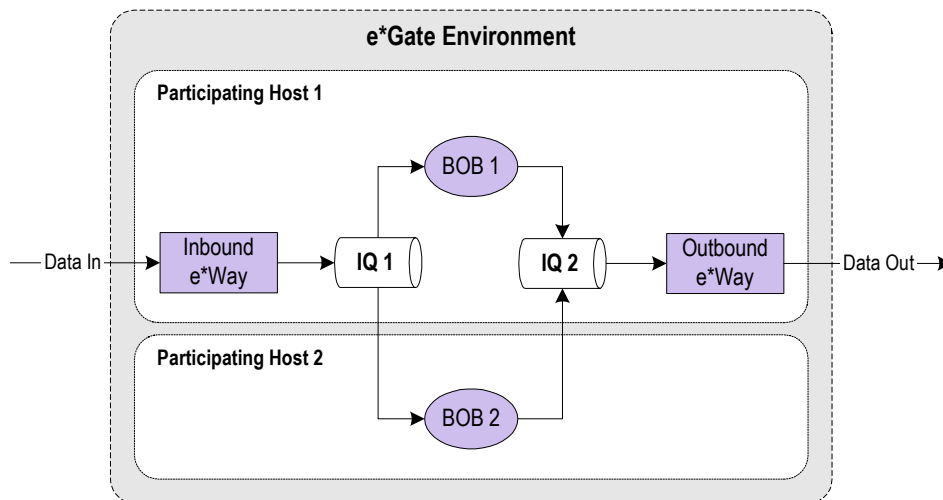
Subscriber Pooling Options

When subscriber pooling is disabled, an Event must be picked up by *all* subscribing Collaborations before it is removed from the IQ. When subscriber pooling is enabled, an Event is removed from the IQ (when it expires) after it has been picked up by the *first* subscribing Collaboration.

To take advantage of distributed processing shown in **Figure 22 on page 93** and Figure 23, subscriber pooling must be enabled. Then an Event that is picked up by BOB 1 is not processed by BOB 2, or any other BOBs that can process Events from the same IQ. In other words, the Event is processed *exactly once* as required, and not multiple times.

Note: *There are some situations when subscriber pooling can impede performance. See* **"Event Serialization and Delivery" on page 111** *for details.*

Hardware Distribution

The hardware distribution of an eBI Suite depends upon the following considerations:

- Physical location

- Hardware burden
- Isolation concerns of other software systems

Distributing Hosts

If your organization has a central office along with branch offices, you can install e*Gate on one or more hosts in the central office *and* on one or more hosts at any of the branch offices. Every host, local or remote, can be a Participating Host. Components that belong on any particular host can be those components most closely related to the external systems that they service, but this is not a requirement. When necessary, you can distribute a BOB or IQ Manager on any networked host.

If a single Participating Host becomes overburdened because of Event volume or complexity, one or more components can be moved to a different host to distribute the processing load. The Participating Host receiving the share of the computing burden can exist in the same location or be remote. In either case, ensure that the network connection can sustain the traffic that you are routing to the new host.

Management Requirements

Different software systems have different management requirements. To avoid conflicting requirements, run some software systems on a separate Participating Host. For example, some database systems undergo periodic backup and put a heavy load on the host input/output during the backup time. This burden can reduce the performance of the entire eBI Suite, if the demand from the database backup is too great.

For more information on how to determine and meet your hardware requirements, see **Chapter 4**.

Performance Summary

You can employ the following guidelines to improve the efficiency of the eBI Suite:

- Eliminate unnecessary parsing by using the simplest ETDs that are able to meet your requirements (see **"Optimizing Event Parsing" on page 108**).
- Eliminate bottlenecks by distributing processing (including BOBs) required for complex Event Types across multiple hosts and using IQ subscriber pooling.
- Reduce or eliminate memory swapping by:
  - Increasing the physical memory in the host
  - Distributing the e*Gate environment across multiple hosts
  - Moving non-e*Gate processes to other hosts
- Reduce the processing burden on a single host by distributing e*Gate components across multiple Participating Hosts.

5.4 Designing Your System

As explained earlier in this chapter, the topology of e*Ways, IQs, and BOBs constitutes the foundation of your system design (see **Figure 22 on page 93**). As you plan an e*Gate environment to meet your business and IS needs, how you set up and order these components determines the basic architecture of that system.

Another important consideration is accommodating the constraints of specific communication technologies inherent in an external system (as well as business organization needs). This section explains how to design system topology, taking all these factors into account.

Note: *During the design operation, it is a good idea to allow for the later implementation of any methodologies that can improve or optimize your system's performance. See* **"Optimizing Your System" on page 104** *for details.*

5.4.1 Determining e*Way Topology

It is preferable to use at least one inbound and one outbound e*Way for each external system interface when two-way data flow is needed. You could need additional outbound and inbound e*Ways depending on:

- Number of interfaces with each external system/application
- Volume of data each interface handles
- Time window of processing for each external system/application

External System Interfaces

One system can have multiple interface capabilities. An external system could need several interfaces depending on the application or applications involved, the number of applications, the system architecture, and the type of operation involved. For example, one system with three different applications requires six e*Ways if you want an inbound and outbound interface with each application.

The type of system itself can require more than one interface. Some systems demand multiple interfaces because of their basic configuration or architecture. In such cases, you must use an inbound and outbound (if necessary) e*Way for each interface.

You must allow for the interface technology of the connecting application or system according to its own interface needs. If the technology of an external system demands at least two interfaces, use at least two e*Ways or sets of e*Ways for each interface, depending on the need for inbound/outbound data flow.

Volume of Data

The volume of data is a critical consideration in assessing needs for the number and arrangement of e*Ways. You must allow for peak data volume handling needs and plan your e*Way topology accordingly.

For example, at peak data-handling periods, if you determine the data volume can overwhelm a single e*Way or set of e*Ways, you must plan for one or more additional e*Ways. If the added e*Way or e*Ways are not enough, more are required. See **"IQ Subscriber Pooling" on page 92** for additional details on how to handle data volume and distribution within the e*Gate environment.

Time Windows

Depending on how an external system processes information, data can only be available from that system to e*Gate at certain times. Time windows affect data volume. Obviously, if a flood of data is available at certain times and little or no data at others, you must first consider the data volume impact (see the previous section).

In addition, set up your e*Way design to allow for this data availability. Even if one or more e*Ways interfacing with this "schedule-driven" data flow does little or no work much of the time, it is best to set up a dedicated set of e*Ways and IQs (and BOBs, if necessary) that only handles this interface. Otherwise, conflicting data schedules could overburden any of the e*Ways in the flow. Also, dedicating one set of e*Ways to a single "schedule-driven" application makes maintainability and troubleshooting much easier.

5.4.2 Determining BOB Topology

The primary purpose of e*Ways is connecting with external systems and applications. However, you could also want to have Collaborations associated with e*Ways doing one or more transformations of data (using Collaboration Rules) along the way. Every task added to an e*Way slows its response time. Too many tasks overload an e*Way and impede data communication to/from the external interface.

The solution to these problems is to create BOBs within your e*Gate environment. These components perform data transformations and decrease the processing load from your e*Ways. This feature frees e*Ways to handle connectivity and data-flow processes with a minimum of impediments.

The number and arrangement of BOBs you need depends on the following factors:

- **Number of data transformations**: If the same Collaboration runs in multiple e*Ways, the system can simplify by having a BOB execute the Collaboration.

- **Data urgency and availability**: If an e*Way spends an large amount of time processing a complex Collaboration, having a BOB execute that Collaboration permits the e*Way to focus on communication.

- **Amount of data**: If a route from one external system to another transfers a large amount of data with complex transformations in between, using one or more BOBs can allow you to distribute the processing over multiple Participating Hosts.

- **Coordinated processing needs**: If Events from multiple sources must be joined into a single data flow before being sent on, a BOB can be created and configured to help this happen.

Number of Data Transformations

You can use BOBs to eliminate duplicate data transformations. The main purpose of BOBs is to handle Event transformations in the place of e*Ways. An e*Way functions in two halves that operate as follows:

- A communications half coordinates interfacing with an external system. This half faces outward and links up with communication from outside e*Gate.

- A Collaboration half transforms data and passes it on to the rest of e*Gate. This half faces inward and ensures correct linkage with the e*Gate environment

If several e*Ways execute the same Collaboration, you can reorganize your system to have a BOB perform the Collaboration rather than each of the e*Ways (see **Figure 24 on page 99**). The upper part of the figure shows three e*Ways that execute Collab 1, Collab 2 and Collab 3.

If Collab 1, Collab 2, and Collab 3 are identical and need to change, the e*Gate programmer must make the same changes to all three Collaborations, an error-prone and tedious task. Moreover, to execute this configuration, three Collaboration Services must be loaded, one for each Collaboration. This design increases the memory demands and processing overhead in the Participating Host.

The lower part of **Figure 24 on page 99** shows how the system can be reorganized to eliminate the duplication. Three Collaborations become a single Collaboration executing in a BOB, which is easier to manage and more efficient to run. Where the e*Ways formerly executed identical Collaborations, they now simply pass the Event through (Pass Through Service) unchanged.

Note: *See the appropriate e*Way user's guide for a full description of the rules of processing performed by that e*Way.*

Figure 24   Eliminating Duplicated Collaborations
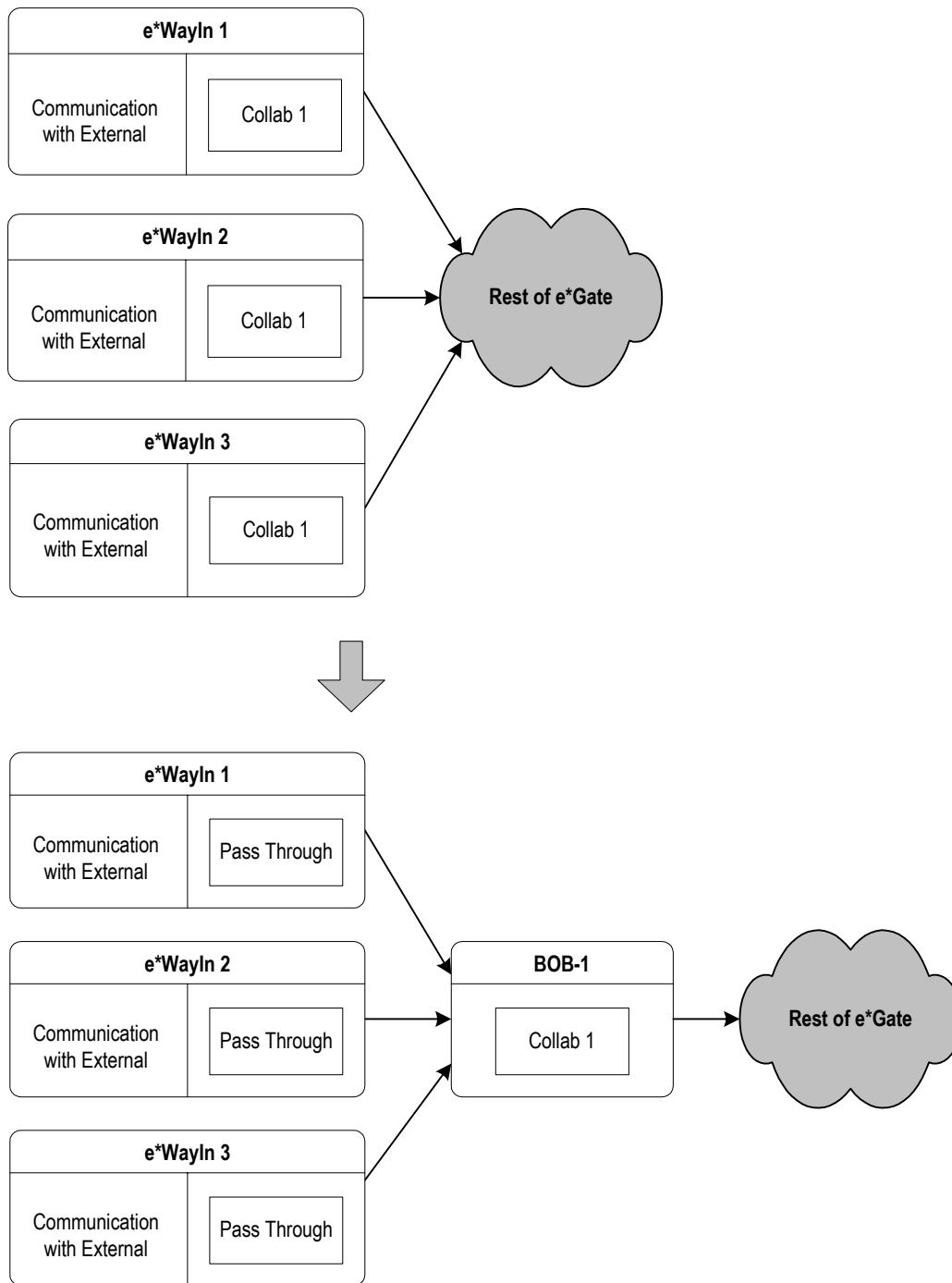
Data Urgency and Availability

You can use a BOB to speed the response time of an e*Way. The urgency and need for certain types of data also affects your number and arrangement of BOBs. For example, data that only goes into an archiving database has little urgency and does not need to be readily available. On the other hand, stock-market quotes must have immediate availability and are urgently needed on a constant basis.

e*Way Architecture Considerations

The architecture of an e*Way is designed to guarantee data delivery if possible or to guarantee appropriate negative acknowledgment if delivery does not succeed. This operation happens as follows:

- For an Event sent by an external system an acknowledgment is returned to the external system if the e*Way successfully processes the Event.

- Successful processing means that the appropriate Collaboration has executed *and* that the resulting Event has been published to the IQ.

- Message processing must successfully complete *before* the acknowledgment is sent.

The requirements of guaranteed delivery, when combined with a Collaboration that takes a long time to run, can cause the e*Way to fail to acknowledge successful receipt of the Event soon enough. The external system sending the Event could think that the Event was not successfully sent.

In effect, the Collaboration becomes a bottleneck preventing the timely delivery of an acknowledgment. To solve this problem, you can make the e*Way execute simple Pass-Through processing and have a BOB execute the Collaboration. Figure 25 shows how this type of design works.

Figure 25   Eliminating Delayed Acknowledgments
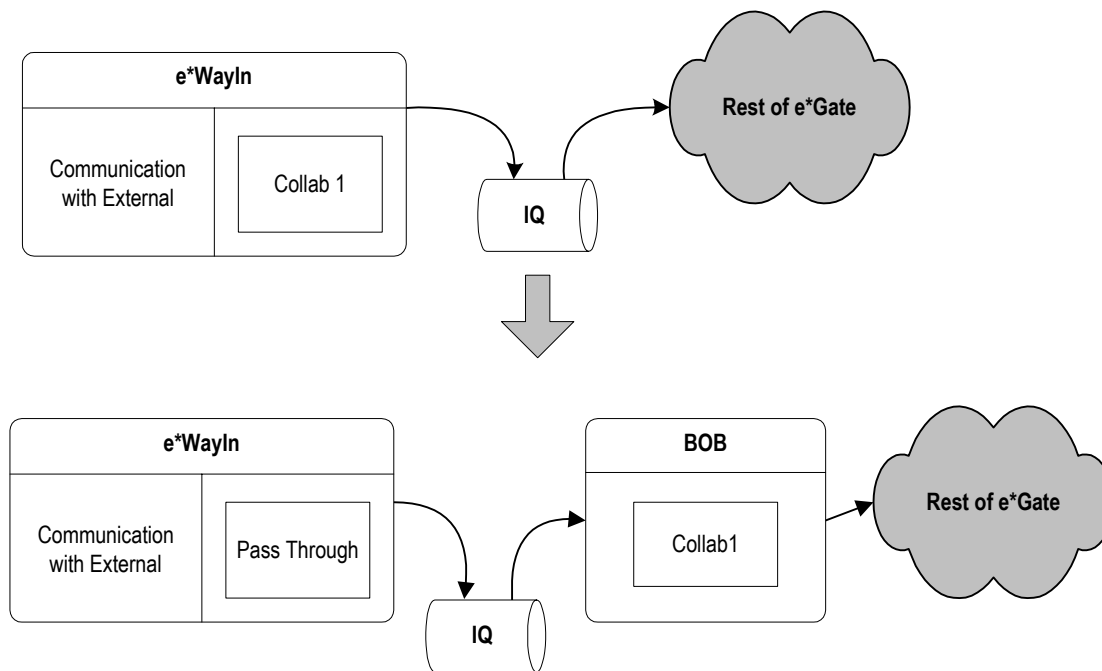
In the upper part of the previous figure, the e*Way executes Collab 1 then writes the result to the IQ before the rest of e*Gate environment does any processing. In the lower part of Figure 25, the e*Way passes the Event through unchanged then writes it to an IQ.

The BOB fetches the Event from the IQ, and applies Collab 1 to it before sending it on to the e*Gate environment. The net change is that the Event is written to an IQ *before* Collab 1 is applied, allowing the Event to be acknowledged sooner.

Amount of Data

You can use BOBs to distribute processing when you have a large amount of data. The BOBs are able to aid one another in a load-sharing arrangement that helps speed their general processing work. For more information on load-sharing techniques, see **"Using Parallel Data Threads" on page 105**.

Data volume considerations for BOBs are basically the same as those for e*Ways. See **"IQ Subscriber Pooling" on page 92** for an explanation of how that e*Gate feature can help you in handling data volume needs.

Multi-Source Transformations

You can use a BOB to join the results of two or more e*Ways. BOBs can bring together data from many different inbound e*Ways and do multiple transformations on this data if necessary, before sending it along to an outbound e*Way (see the following figure).

Figure 26   BOBs and Multi-Source Transformations
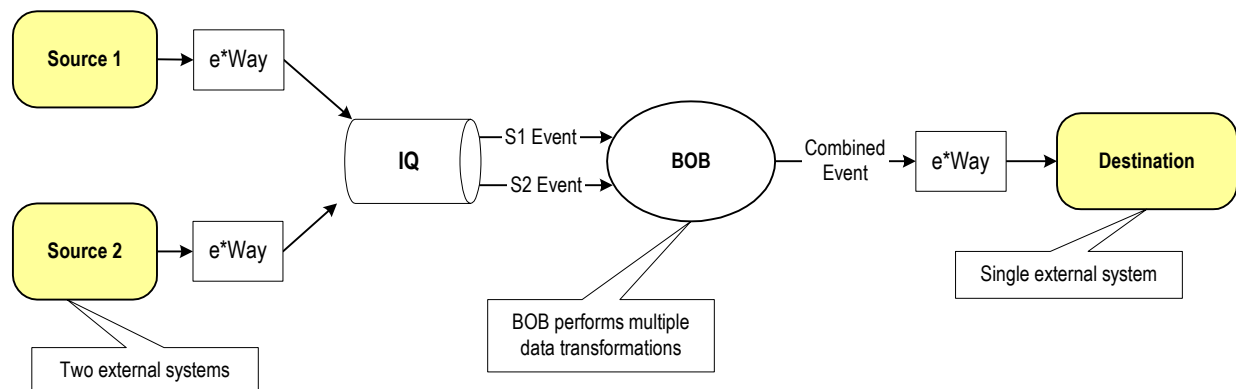
In the previous figure, the Events sent to the external system by the outbound e*Way are comprised of data from both of the other external systems. The BOB gathers data from each source. Using the scenario outlined in Figure 26, you can also send the data to multiple outbound e*Ways, if necessary, using the same BOB.

For more examples, see the case studies under **"Case Study Examples" on page 133**. Also, see **"System Topology and Business Organization" on page 103** for details on system design and business organization.

5.4.3 Determining IQ Topology

This chapter has already treated the primary use of IQs in load balancing and how to design their topology under **"IQ Subscriber Pooling" on page 92**. The following additional considerations are helpful in determining your IQ topology:

- If you have data flowing in from one source but going out to several destinations, use one IQ to control this flow.

- If you have information from many destinations that must be consolidated into a single flow, use one IQ to control this flow.

- If neither of the previous situations is a concern, you can use several IQs for load balancing in a subscriber pooling setup.

- If you have sensitive data (for example, confidential corporate records) that you want to keep out of the rest of the system, dedicate one or more IQs only to this data.

- If possible, do not use more than three IQs per one IQ Manager. For the most efficient results, configure the IQ Manager and its associated IQs on the same computer.

Note: *For more information on IQs and high availability, see the* **Appendix D**. *For complete information on IQs and how to configure them, see the* **e*Gate Integrator Intelligent Queue Services Reference Guide**.

5.4.4 Using e*Gate Java Features

Release 4.5 of e*Gate (and later) allows you to create ETDs and Collaboration Rules using the Java programming language, as well as e*Gate's original language, Monk. Using the SeeBeyond Java Message Service (JMS) IQ Manager and Service features can significantly improve your system performance.

Note: *In e*Gate release 4.5.1 and later, the JMS IQ Manager and Service are configurable on* **all** *e*Gate-supported platforms.*

If your available human resources are Java-oriented and have more experience using this language, you can use these features to streamline your deployment without the "ramp-up" time required for programmers to learn a new language.

For a sample case study using Java ETDs and Collaborations, see section **"Case Study 3: Tracking Timecards and Payroll Scenario" on page 144**.

For complete information on how to use the GUI Java features in e*Gate, see the *e*Gate Integrator User's Guide*.

5.4.5 Accommodating External System Constraints

In designing overall system topology, you must factor in external system constraints. Ensure that your number and arrangement of e*Ways and BOBs takes these system and application needs and limitations into account.

First Example

For example, you have a Siebel system used for placing new customer orders that must immediately go to an SAP system whose purpose is to fulfill those orders. The orders must be filled and the products shipped out right away. No data transformations are necessary along the way. For this purpose:

- Dedicate a single inbound e*Way to the Siebel system.

- Dedicate a single outbound e*Way to the SAP system.

- Link the two e*Ways via a single IQ.

Dedicate this arrangement only to this purpose. Using such a setup, orders can proceed quickly and efficiently. In this case, your business process is enhanced, and the needs of the external systems are satisfied.

Second Example

On the other hand, in another example, you have a Siebel system that needs to synchronize customer orders every six hours. These orders must flow into an SAP system on the same schedule. Using that schedule, feedback is necessary from the Orders Fulfillment Department back to the Customer Order Department. Data transformations are required in both data flows.

An arrangement to meet these needs could look something like Figure 27.

Figure 27   Scheduled Two-Way Order System
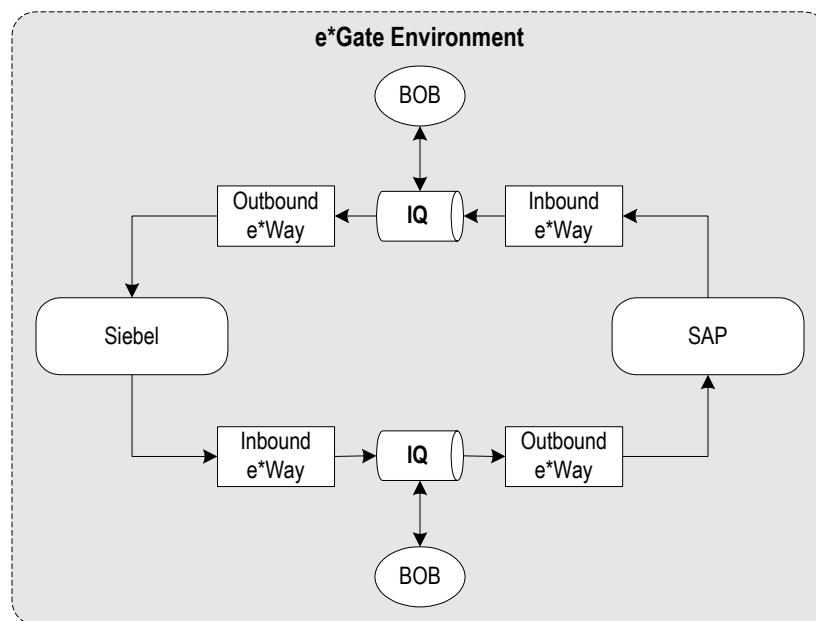
As in the previous example, it is best to dedicate this system (Figure 27) only for this purpose. In such a setup, orders can proceed as scheduled, with data appropriately transformed in both directions. The BOBs help to synchronize and control the data flow. In this case, once again, your business process is enhanced, and the needs of the external systems are satisfied.

5.4.6 System Topology and Business Organization

Early on in the design process, you are thinking about technical considerations, such as data volume, external system interfaces, and the number of data transformations you need. Once you have allowed for all these factors, you must start to consider the "bigger picture." System design topology must also take into account your company's organizational needs. Good planning for these considerations greatly enhances your e*Gate environment's long-term maintainability and supportability.

Right Topology for Your Business

No one recipe can take care of everyone's business organization needs. However, the following general considerations help:

- Set up at least one (or two for a two-way data flow) e*Way for each organizational unit of any importance in your business. These units could be teams, departments, management groups, or any major unit in your organization. This setup allows for easier maintainability and accountability. If there is a need or problem with the system in one group, that group can be responsible for taking care of it. Several groups sharing one e*Way or set of e*Ways could lead, later on, to a lack of accountability.

- Allow for as much needed redundancy as possible. If a data connection is vitally important to your business or if you need it to be open continually for any reason, make sure you have enough e*Ways on hand to back up this connection in case it needs maintenance. This allowance makes supportability easier and minimizes problems in case of trouble.

- Set up your schema structure to reflect related processes. In other words, if you create more than one schema, assign components with common purposes and functions to each schema. For example, you could set up a schema to handle accounts payable operations and another to handle accounts receivable. Also, if you have many processes contributing data to a single database, configure all these processes to take place in a single schema.

- Major operations or business processes with a high priority require additional e*Ways and, if necessary, BOBs. After you determine the e*Way demands for your peak usage, add one or two more e*Ways to your total number. Remember that peak demands can sometimes be exceeded. Allowing for this possibility enhances total system supportability and also maintainability.

5.5 Optimizing Your System

Once you have set up your overall eBI Suite design, you then need to go back over the entire setup and determine where you can optimize and improve system performance. This important step ensures the highest-quality results once your system goes into operation.

However, keep in mind that when you do your original system design, it is good practice to allow for the optimization plans explained in this section. System optimization is not just a single step. It is instead an ongoing process that you can implement in various ways during all phases of deployment and beyond.

This section provides general guidelines on system optimization, listed topically by each area and application where key improvements can be realized.

Determining an Optimization Plan

All the optimization methods described in this section improve data throughput. In deciding on and choosing optimization plans, you must first determine whether the improvement you seek is worth the cost. Different types of cost can be:

- Monetary, for example, using faster hard disk controllers.

- Performance, not in data throughput but in other ways of measuring performance, such as:

  ◆ Increased latency, that is, total throughput improves, but individual transactions during low-volume periods can be delayed longer than they would be without the optimization method

  ◆ Increased complexity could improve, for example, total throughput, but the schema is more difficult to support

In other words, when designing your eBI Suite environment, do not plan to implement as many different methods as possible. Instead, decide on acceptable levels of data throughput, latency, complexity, and cost. Next, find a balanced mix of these optimizations to achieve desired efficiencies.

The rest of this section provides tips on optimization and how to balance this mix, along with examples.

5.5.1 Using Parallel Data Threads

One of the simplest ways to improve e*Gate performance is to set up parallel data threads through your system. This principle works in basically the same way as adding more lanes to a freeway, allowing traffic to move faster. See Figure 28.

Figure 28   Using Parallel Data Threads
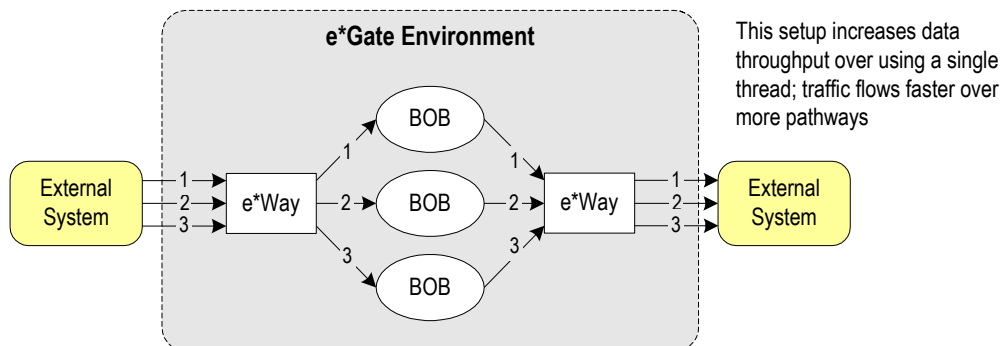
The type of data distribution shown in the previous figure is called *load balancing*. You can create and configure as many different data threads and components on as many different systems as your overall resources allow. Test your results, because if you are straining these resources, adding more threads can decrease instead of increasing performance quality.

Multi-threaded BOBs: Keep in mind that you can adjust the number of threads per BOB as well as the number of BOBs. See **Figure 29 on page 106**.

Figure 29 Using Multiple Threads per BOB
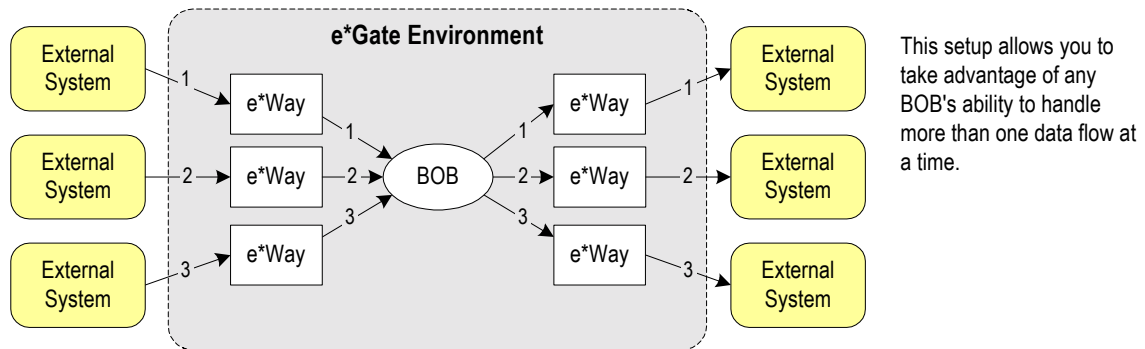
Depending on your needs, you can use more BOBs with fewer threads or vice versa. Determine which of these setups you need as follows:

- Use several, single-threaded BOBs if you are using a single-threaded library, for example, HTTP, or if you need to start and start BOBs individually as necessary.

- Use fewer, multi-threaded BOBs for multi-threaded libraries, for example, database libraries.

In general, fewer BOBs and more threads use system resources more efficiently than fewer threads and more BOBs.

5.5.2 Improving IQ Performance

The following list provides some helpful tips on how to improve IQ performance in the eBI Suite:

- Empirical evidence has determined that using two to three IQs per IQ Manager produces optimum results.

- If the total number of your CPUs is not an issue, it is advisable to add IQ Managers whenever possible. Keep track of the CPU use of the IQ Managers and decide how many you can add based on that calculation.

- Make sure you create an IQ Manager on the same host computer as its associated IQs to allow for unrestricted access between the two components.

- When an IQ Manager puts an Event into an IQ, it briefly locks the IQ from all other publishers. Using multiple IQs reduces the possible number of lock contentions for IQs, across all publishers to those IQs. See Figure 30.

Figure 30 Optimizing IQs: Using Multiple IQs
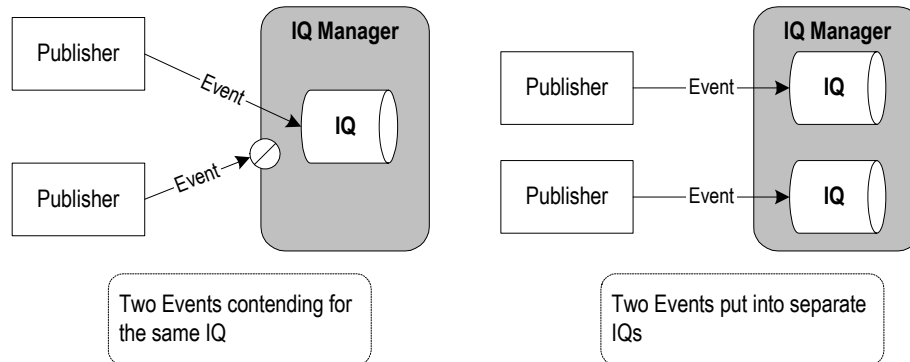
- Hardware vendors like Sun Microsystems, Hewlett Packard, and IBM have hard disk storage solutions that utilize nonvolatile caches on the disk controllers. Although this is an expensive solution, it can significantly improve IQ performance, because IQ Managers usually write a large volume of small-set bytes as part of their updating Event states in the IQ.

- Try to reduce the number of puts and gets to the IQ as follows:

  - Try batching Events together. Whenever you can afford to batch Events then do so. See **"Batching Events" on page 110** for more information.

  - Use IQ subscriber pooling to streamline system processing by making it parallel. See **"IQ Subscriber Pooling" on page 92** for more information.

Note: *Subscriber pooling may not be appropriate if Event serialization is a priority. See* **"Event Serialization and Delivery" on page 111** *for details.*

- If you do not need to retain Events in an IQ for long time periods, set the Events' expiration for a shorter time period. Then, set the IQ Manager to clean up its IQs at multiple times per hour or per day. Whether you do IQ cleanup per hour or day depends on the system's total data volume and peak times.

Note: *Do not set IQ cleanup to less than a 5-min interval. Verify that the IQ Events' expiration period is longer than the IQ cleanup time or Events can expire during cleanup. Also, keep in mind that the larger the number of Events an IQ handles, the longer the IQ Manager takes to clean up its IQs.*

The cleanup process reduces the size of live index files in the SeeBeyond Standard IQs, making the IQ-put actions go faster. Otherwise, these files can keep on growing and taking up more hard disk space, causing performance degradation.

Handling High Volume Throughput

If you need to handle high-volume throughput in any e*Way or BOB in your system, and you have a small number of sources and source Events, use the following guidelines:

- Use one IQ per publisher. This setup helps increase the throughput and also reduces risk. If an IQ becomes unavailable, only one link to an internal component or external application is affected or goes down.

- For extra-high data throughput, use one IQ per publisher and Event Type. See the example shown in the following figure.

Figure 31   Optimizing IQs: One IQ per Publisher and Event Type
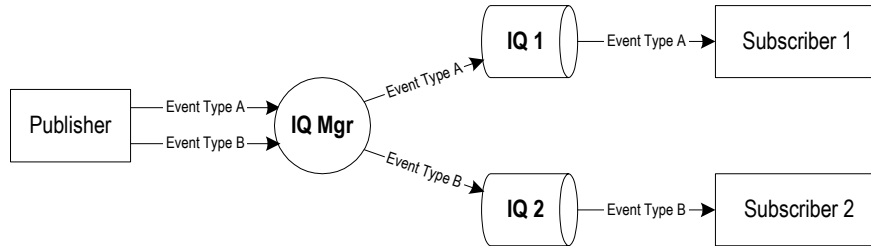
- Too many subscribers using the same IQ can stifle the responsible IQ Manager. To avoid this problem, create multiple IQs and, if necessary, an Event Type for each IQ. This setup takes advantage of multiple threads in the IQ Manager. The following figure shows an example.

Figure 32   Optimizing IQs: One IQ and Event Type per Subscriber
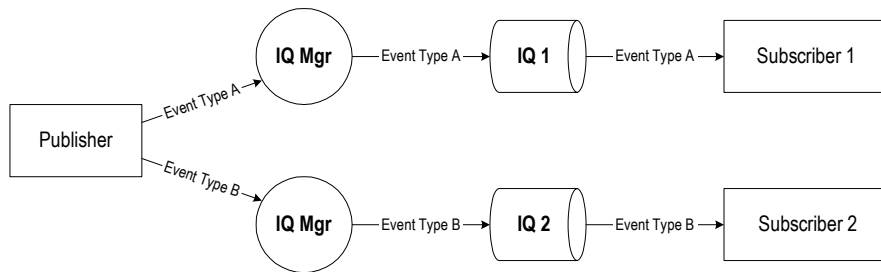
In any case, the maximum number of client connections per IQ Manager is approximately 50. The total number of client connections equals the total number of Events handled by the IQ Manager regardless of direction. For example, in the previous figure, the IQ Manager has four client connections.

5.5.3 Optimizing Event Parsing

Whenever an eBI Suite Collaboration utilizes an Event for any kind of Service other than Pass Through, the Collaboration must first parse the Event. To do this operation, the system reads and processes structural information within that Event.

Every parsing action takes time and system resources. To optimize performance, you must avoid unnecessary parsing. Verify that each Collaboration is only parsing every Event as much as necessary to do the assigned transformation and no more.

Avoiding Excessive Parsing

Here is a checklist of steps you can take to avoid unnecessary Event parsing:

- **Choose efficient functions,** as in the following examples:
    - Use the Monk function **db-struct-bulk-insert** to transport Events into a database e*Way (for example, an Oracle e*Way) works more efficiently than **db-sql-execute**.
    - The Monk function **$event-clear** causes all available memory for an ETD to be released. If the Event's ETD is to be reused in the same Collaboration, instead of **$event-clear**, use a statement like:

      (copy " " ~evt%root:0-END)

Note: *For more information on Monk functions and statements, see the **Monk Developer's Reference Guide**.*

- **Grouping too few records** in a bulk-insert statement is inefficient, but too many has a significant message-parsing overhead. There is a good average to aim for, which is probably data-dependent, but has been observed in many tests to be approximately 500 records per insert region.

- **Consolidating nodes** where possible. Create new ETDs based on original (full) ETDs by consolidating the original ETD nodes in these new ETDs. Make sure the parent node has its length set to the total lengths of its children. See the following figure.

Figure 33   Node Consolidation
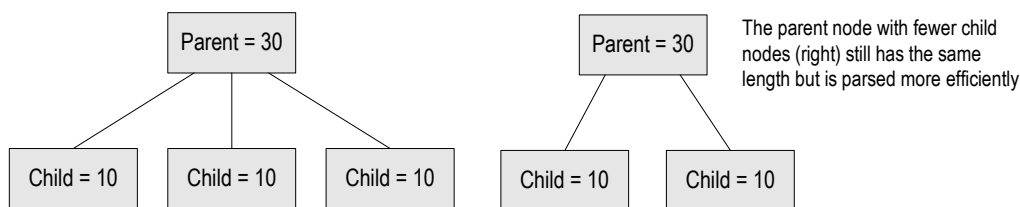
For example, if you only need to process the data in the first child node, you could consolidate the nodes in the way shown in the previous figure. Consolidate nodes in this way only if processing data in the other nodes is not necessary.

- **Stopping unnecessary parsing:** If child and leaf nodes are not referenced, eliminate those nodes from the ETD. Also, use the Pass Through Service whenever possible. For more information, see **"Event Parsing" on page 92**.

- **Checking node lengths:** When parsing a fixed-length Event generated by a database ETD builder, check all node lengths carefully. Try to make the parent node or unspecified length the same as the "value" node and not 0 (zero). With e*Gate version 4.1.0 especially, this setting has a significant positive effect on the speed of message parsing.

  For example, if node 0 represents the rest of the data in an Event, and we know we only need 100 bytes more of this data, it would be more efficient to represent this node as node 100.

- **Avoiding multiple IQ read-writes:** Configuring multiple small read-write actions in a single IQ uses inordinate system resources. Try to batch as many small Events

as possible into a single larger Event that contains a repeating node. For more information, see **"Batching Events" on page 110**.

- **Consolidating Collaborations** wherever possible. Also, the use of additional BOBs does not always improve performance; there is a point where returns diminish. Test added BOBs and their Collaborations, and if their use degrades performance, consolidate them wherever possible. For example, if you have two or more BOBs doing the same operation, you could combine them and their Collaborations into a single BOB and Collaboration.

- **Avoiding expanding ETDs**: Dynamically growing ETDs, that is, those that add more repeating nodes, are expensive in terms of RAM management. Only create these types of ETDs if it is absolutely necessary.

Batching Events

Bundling multiple Events into a single Event when it enters e*Gate is called *Event batching*. You then design the system to split them apart when they leave e*Gate. This process restores the Events to their original separate state.

For example, you could have a system with Events entering the system separately as shown in Figure 34. This setup causes a separate disk write for each of the three Events when it is handled by an IQ.

Figure 34 Non-Batched Separate Events

Each parent node is actually an Event "root" node that can contain as many or few child nodes as desired
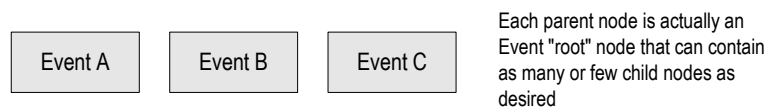
To improve efficiency, you can create a Collaboration (in the inbound e*Way or in a BOB) to join them into a single Event when they enter the e*Gate environment. The resulting Event appears as shown in Figure 35. This setup causes only a single disk write for the batch Event when it is handled by an IQ.

Figure 35 Events Batched into One Event

All three original Events are now bundled together as children of a new "root" node parent in a single new Event
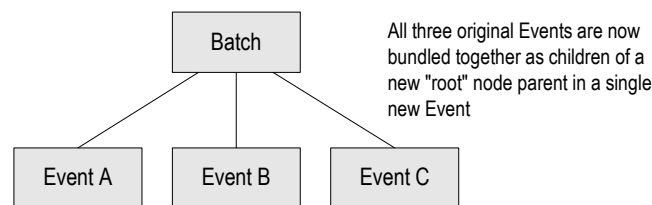
Then, you can have the batched Event sent through the e*Gate environment in any way you want, keeping the contained Events together. To unbundle a batched Event before it leaves the system, create another Collaboration (in the outbound e*Way or in a BOB) to split it apart into its original separate Events.

Advantages of Batching

The advantage of Event batching is speed. In most cases, batching Events greatly improves data throughput and, as a result, overall system performance. Publishing

fewer, larger Events to IQs uses less system-resource overhead than publishing more, smaller Events. Actual experience has shown that as many as 10,000 smaller Events can be efficiently batched into a single "parent" Event.

Note: *Carefully test a Collaboration's batching script or program before using it. A flawed batching process can cost you all the benefits Event batching has to offer.*

Event Serialization and Delivery

Events can take different routes through an e*Gate environment. As a result, there is no guarantee that they all arrive at their final destination in the same order they started out with. If Event order is not important, you can use Event batching, as well as IQ subscriber pooling as desired, to improve system performance and move Events along independent threads through e*Gate.

Note: *See* **"IQ Subscriber Pooling" on page 92** *for more information on IQ subscriber pooling.*

For example, suppose Events arrive at e*Gate in the following order: 1, 2, and 3. Event 1 could take an independent side trip to an external database while Events 2 and 3 proceed directly. Thus, the final order they are received in could be 2, 3, and 1. If order is not important, this processing topology is no problem.

But what if Event 1 carries information needed to interpret Event 2? Then, there could be problems if you do not guarantee that Events end up in the same order they had when they started.

Guaranteeing Serialization

Ensuring *Event serialization* means guaranteeing, through your system design, that Events are received in the order they are sent. Of course, Event serialization is only a consideration in situations where the order of Event processing is critical.

Using a more specific example, if you are processing account transactions where credits are entered first before debits (especially at high speeds), naturally you want Events processed in exact time sequence. Otherwise, a debit Event that arrives too soon could "bounce" because the destination system does not yet know the money is there.

The following figure shows a sample non-serial Event sequence.

Figure 36 Non-Serial Event Sequence
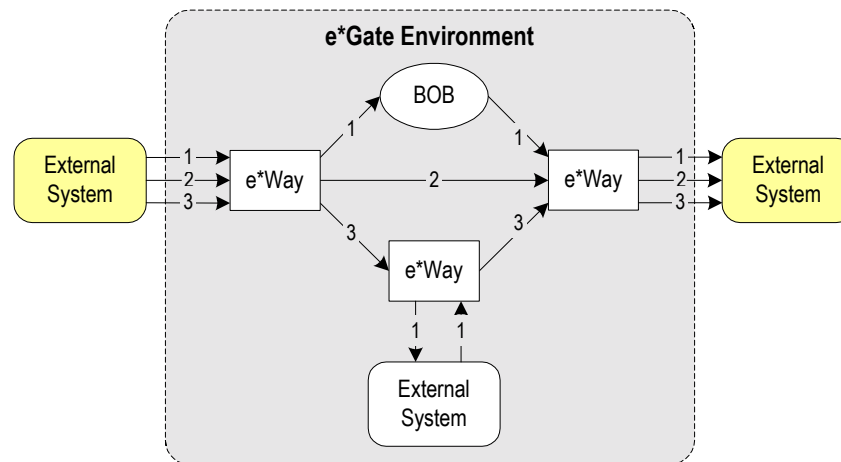
In the previous figure, Event 1 was sent first but may not arrive at its final destination ahead of the other two Events. Avoid too many such "detours" for Events when they must arrive in sequence. See **"Using Parallel Data Threads" on page 105** for information on load balancing and how to allow for Events' being processed in parallel threads.

Using e*Gate features, you can guarantee Event serialization by:

- Sending all Events you want to keep in serial order to a single IQ

- Assigning these Events to a single Event Type

If, in Figure 36 the Events leave e*Gate in 1-2-3 order, that is, in the same order they entered, they have been serialized or *sequenced*.

Using Event Linking and Sequencing

Using e*Gate's Event Linking and Sequencing (ELS) feature ensures that your data arrives at its destination in the desired sequence, after leaving e*Gate. This feature is a part of the Java Collaboration Service (available in e*Gate release 4.5 or later).

This feature allows you to create Event sequences with greater complexity and flexibility than those available only in Monk. For example, Java Collaborations allow you to transform one Event Type to many, many to one, or many to many instead of the simple one-to-one correspondence available with Monk.

ELS is configurable using the Java Collaboration Rules Editor in e*Gate. The case study example that begins later in this chapter shows an example using this feature. See **"Case Study 3: Tracking Timecards and Payroll Scenario" on page 144** for a sample deployment of ELS in e*Gate via a Java-enabled configuration.

For more information, see the following references:

- *e*Gate Integrator Collaboration Services Reference Guide* for an explanation of the Java Collaboration Service and ELS

- *e*Gate Integrator User's Guide* for an explanation of Java Collaborations and the Java Collaboration Rules Editor

Using Guaranteed Exactly Once Delivery

e*Gate allows you to use the system's Guaranteed Exactly Once Delivery feature that ensures the once-only delivery of each Event. If your system requirements dictate that every Event that leaves e*Gate be delivered to its destination system *only* once, you can use this feature to achieve that result.

Note: *The use of Guaranteed Exactly Once Delivery may slow system performance somewhat. Only configure this feature where its need is dictated by specific business requirements.*

XA-aware features are available using the SeeBeyond JMS IQ Manager and Service. See the *SeeBeyond JMS Intelligent Queue User's Guide* for detailed information on the JMS IQ Manager and Service and how to configure Guaranteed Exactly Once Delivery.

Possible IQ-pooling Problems

Using IQ subscriber pooling can cause problems with Event serialization. Once Events go into an IQ "pool," Collaborations pick them up as they are able and not in time sequence. For situations where Event serialization is necessary, the subscriber pooling feature may not be desirable. In such cases, consider Event sequencing when deciding whether to use this feature.

5.5.4 Monk Optimization

There is no definitive formula optimizing Monk-related operations in the eBI Suite. Keep in mind that the best way to optimize programming is to first program the code so that it works correctly, even if not efficiently. Once you have a program that operates well, then you can concentrate on optimization.

Note: *This section assumes that the reader at least has some familiarity with Monk programming. For more information on using Monk, see the* **Monk Developer's Reference***.*

The following list explains some handy Monk optimizations you can apply in your deployment:

- **Internalize slow functions:** Use the C Collaboration Service to write a C-language function that replaces appropriate Monk functions with those contained in compiled .**dll** files. Then, use the Monk **Invoke** function to load the .**dll** files.

- **Minimize path access:** Using strings or byte offsets instead of paths speeds Monk processing considerably. In cases where Monk maps a string, have it map the string

or byte offset directly instead of by path. For example, suppose you need to find an employee number in an ETD. The following line shows the map by path:

```
employee
  (
  firstname 5
  lastname  6
  IDnum     14
    (
    number       10
    type         1
    checkdigit   1
    )
  )
```

The following line shows the map by string:

```
input%employee.IDnum.number
```

The following line shows the map by byte offset:

```
input%employee[13,10]
```

For better efficiency, use the second type of mapping shown in the examples above. The byte offset method only applies to fixed-format ETDs.

▪ **Eliminate duplicated collaborations:** If more than one Collaboration serves the same purpose, consolidate them into a single Collaboration. See **"Number of Data Transformations" on page 98** for details.

▪ **Precalculate node lengths in fixed ETDs:** This optimization streamlines Monk's mapping processes. It does so by letting the system know beforehand how many bytes of data to pass on to a fixed-length parent node with fixed-length children. The children make up a parent by the sum of their sizes. Give the parent a length equal to the sum of its children's length (see the following figure).

Figure 37   Precalculating Node Lengths

Parent = 30

The parent node informs the environment how many bytes of data its child nodes possess

Child = 10   Child = 10   Child = 10
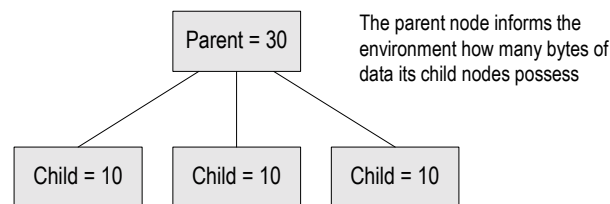
This operation is similar to the node-parsing example shown in **Figure 33 on page 109** (see that section for related information). This type of shortcut can improve mapping speed by several times.

Similarly, you can ask the system to map by byte offset. For example, suppose you want the system to map a zip code in an address in a Level 2 node of a fixed-format ETD. You could "find" the zip code as follows:

```
input%msg.region.company.zip
```

Or if you know the zip code is always at byte 68, you could direct the system to it as follows:

```
input%msg[68]
```

By using the second method shown above, you allow the system to go to desired point directly (and more efficiently), instead of its having to completely map every node on the path.

▪ **Avoid unnecessary memory allocations:** Reading numerous blocks of data from a file and appending it to a Monk string variable can hurt the Monk program's performance. The amount of allocation and reallocation Monk has to do in such cases consumes extra system resources. A good alternative is to pre-allocate a large block of memory space using the **make-string** function and by using a combination of **string-copy!** and **string-length!** functions to manage the data.

Here are some examples of these functions:

```
(define xxx (make-string 10))
(string-copy! xxx 0 "1234567890")
(display xxx)(newline)
(string-copy! xxx 4 "abc")
(display xxx)(newline)
(string-length! xxx 5)
(display xxx)(newline)
```

In the previous example, the code writes the following string to a log file after line three:

```
1234567890
```

In the previous example, the code writes the following string to the log file after line five:

```
1234abc890
```

In the previous example, the code writes the following string to the log file after line six:

```
1234a
```

Using functions in this way prevents Monk from doing any further reallocation to the string variable to come up with the same results.

▪ **Use variables in loop conditions:** Often you must use loops where the stopping criteria depends on the length of a string or the number of times a certain node repeats. In these cases, pre-calculate the length of the number of iterations in a Monk variable and use the variable in the stopping criteria. This optimization prevents Monk from having to recalculate the same value, when it can be stored in a variable once then referenced later on.

In the next example, the variable *number_of_repetition* stores the actual repetition count. The purpose of the program is to avoid evaluating the *count-rep* expression every time the test condition of the **do** rule is not evaluated to **#t** (true).

```
(set! number_of_repetition (count-rep
~input%root.node_that_repeat))
(do ((index 0 (+ index 1))) ((>= index number_of_repetition))
(display (string-append "Repetition Data [" (number->string index)
    "] = "))
(display ~input%root.node_that_repeat[<index>])
(newline)
```

▪ **Use Monk map caching:** In some situations, the same Monk ETD and the same data are used by several functions, either sequentially or hierarchically. Here, you can allow the first function that maps the input data to cache the input ETD. Then, this

function can globally share this cache with the remaining functions that use the same ETD. The rest of the data transformations must then be converted to database-access poll functions to bypass the input data mapping process.

Monk Map Caching Examples

```
(define bbdl_process_request
 (let ((input ($make-event-map bbdl_request-delm
     bbdl_request-struct))
        (output ($make-event-map bbdl_response-delm
            bbdl_response-struct))
     )
   (lambda (message-string)
     ($event-parse input message-string)
     ($event-clear output)
     (begin
        (define cached-struct ($make-event-map bbdl_request-delm
            bbdl_request-struct))
        (set! cached-struct input)
        (sub-xlate)
        :
        :
```

This operation expends more system resources than the sub-call shown in the next example.

The previous example creates a Monk ETD and assigns the function's input variable to it. The cached ETD contains everything as input.

```
(define sub-xlate
 (let ((input ($make-event-map bbdl_request-delm
     bbdl_request-struct))
        (output ($make-event-map bbdl_response-delm
            bbdl_response-struct))
     )
   (lambda ()
     ($event-clear output)
     (begin
        (set! input cached-struct)
        :
        :
        :
        (set! input output)
```

This sub-call, does not contain the expensive **$event-parse** operation and instead uses the cache, expending fewer system resources

The previous example uses the database-access poll function. The input variable gets assigned the cache. At the end of the function, the input must be set to output.

5.5.5 Optimizing Performance Using Hardware

Here are a few brief tips on how to optimize your general eBI Suite hardware performance:

- Disk speed is often a limiting factor; use the fastest hard disks possible.

- Linear scalability (increasing CPU yields, therefore increasing data throughput) can only be observed when disk speed is not a limiting factor.

- Keep IQs and their IQ Managers on the same computer. Separating them onto different systems can degrade performance.

- Use fast hard disks with battery-powered caches if possible. This improvement is especially beneficial to IQ performance (see **"Hard Disk Access" on page 69**).

Note: See **Chapter 4** *for detailed information on how to optimize your eBI Suite hardware setup.*

5.5.6 e*Insight Engine Optimization

This section explains e*Insight Engine features you can use to optimize your system performance. These features are:

- **e*Insight Engine Affinity** allows e*Insight Engines in a multi-engine e*Gate schema to cache information about particular Business Process Instances (BPIs) as they flow through an e*Gate schema. Using e*Insight Engine Affinity can possibly improve performance. Using this topology, if an Engine is shut down for some reason, the instances associated with that Engine will not finish being processed until the Engine is manually restarted using the e*Gate Monitor GUI.

- **Instance Caching** lets you use a single e*Insight Engine to cache information about BPIs. Using this feature, you can reduce system disk-access instances and therefore improve general system performance. Employ this feature with any use of the e*Insight Engine, even it you are only using a single Engine in an e*Gate schema.

- **Adding additional e*Insight Engines** combined with Instance Caching offers the benefits from reduced disk access and load balancing. e*Insight Engine Affinity allows an e*Insight Engine in multi-engine e*Gate schema to start and finish particular BPIs as they flow through a business process and e*Gate schema. Using this topology, if an e*Insight Engine is shut down for some reason, the instances associated with that Engine will not finish being processed until the Engine is manually restarted using the e*Gate Monitor GUI.

For more information on how to set up, configure, and use all these features, see the *e*Insight Business Process Manager Implementation Guide*.

5.5.7 e*Xchange Optimization

Trading Partner Profile Caching

If you are deploying a large number of trading partners, this feature minimizes database lookups for partner profile information, which can be costly in a high-volume environment.

The system default settings for this feature are:

- Expiration Time of trading partner profile in Cache
- Maximum trading partner profiles in Cache

Figure 38 shows the GUI that allows you to access these settings.

Figure 38   e*Xchange System Defaults—Editing Window

These settings define how many profiles are held in memory and when they are expired if not used. This feature also operates, using a least-used algorithm, so if all the memory slots allocated are used, and a new profile is loaded, the least-used profiles in memory are removed.

Multiple Batch e*Way Instances

When dealing with high volume batch connections, consider using more than the standard base Batch e*Way. Using the batching facility, you can create a multiple Batch e*Way and give each one a unique instance number within the batch. For example, in a deployment with ten batch e*Ways, you can assign each one a unique instance number between 1 and 10 (see **Figure 39 on page 119**).

Figure 39   e*Way Editor for Batch e*Way
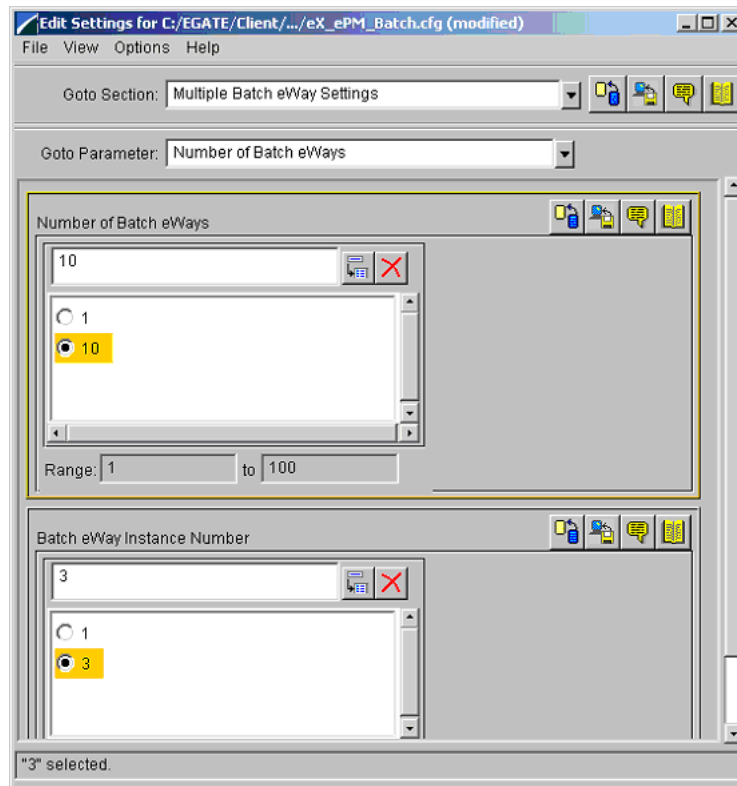
When batch messages are being processed, e*Xchange applies a modular calculation to split transactions across multiple e*Ways. This process can greatly improve performance. It is also possible to set the maximum number of messages in a batch, which can improve the batch processing of messages.

5.6   System Development Considerations

This section discusses some important guidelines for e*Gate development, that is, how to set up and configure e*Gate for your deployment.

5.6.1   Overview of e*Gate Development

As described earlier, setting up e*Gate requires a step-by-step approach to system design, architecture, and planning for component interaction. Once you have created a basic system design, you must build on that plan by developing your complete e*Gate system. Then use the e*Gate GUIs to create and set up all the needed system components.

You must configure most e*Gate components before the system can use them correctly. Configuring e*Gate components utilizes the same GUIs as system setup. Access e*Gate's system development and setup features via the Enterprise Manager GUI.

e*Gate GUIs

In addition to the Enterprise Manager, e*Gate uses the following GUIs for setting up, configuring, and editing its components:

- ETD Editors (Monk and Java)
- Collaboration Rules Editors (Monk and Java)
- Collaboration-ID Rules Editor (for backwards-compatibility with e*Gate Version 3.6 only)
- e*Way Editor

Each of these GUIs has its own window and pane features as well as an online Help system. Also, all the GUIs, including the Enterprise Manager, have properties and other dialog boxes that aid in the e*Gate component creation, editing, and configuration operations.

Setup Steps

The **Figure 40 on page 120** shows basic e*Gate setup steps in the form of a road map.

Figure 40   e*Gate Setup Road Map
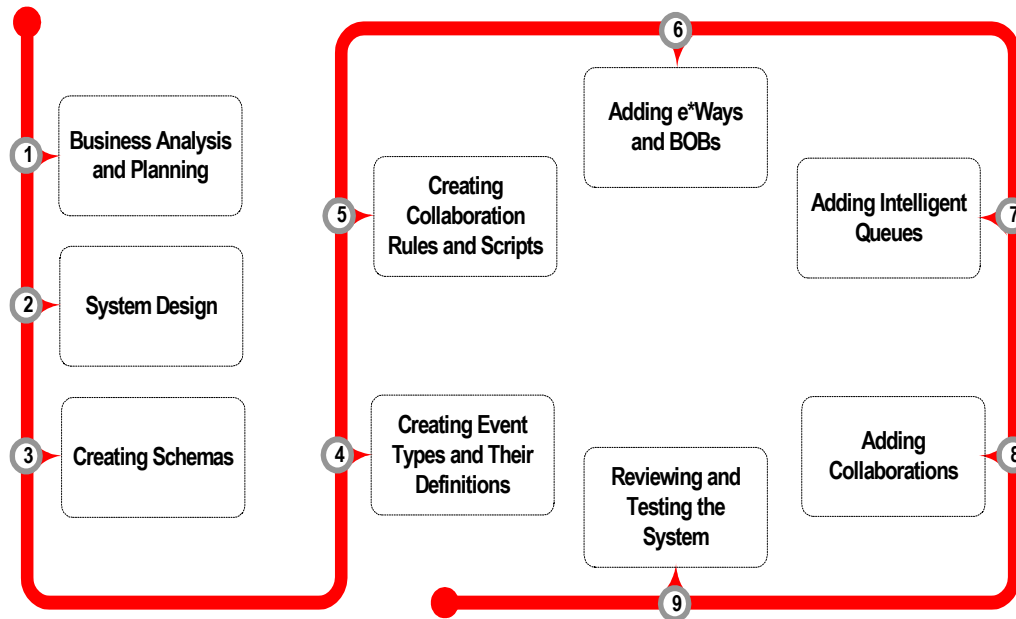
As discussed earlier, analysis, planning, and system design are basic phases of the deployment project. In addition, actual system development requires the following steps:

- Creating schemas
- Creating Event Types and their definitions (ETDs)
- Creating Collaboration Rules and scripts

- Adding e*Ways and BOBs
- Adding IQs
- Adding Collaborations
- Reviewing and testing the system

See **Chapter 6** for complete information on system testing and review. For specific information on how to create, configure, and operate the e*Gate components in the previous list, see the *e*Gate Integrator User's Guide*.

5.6.2 Setting Up Users, Roles, and Privileges

The e*Gate Access Control List (ACL) feature defines users' privileges to take specified actions on components. This is e*Gate's basic security feature that controls development activities, such as creating or modifying components, and operation activities, such as starting or shutting down components. For example, using the ACL, you can permit one or more users to create or modify certain e*Gate components and other users to only start up or shut down certain e*Gate components.

Note: *ACL security features explained in this section only apply to e*Gate release 4.1.2 or later. For details on e*Gate's ACL features, including assigning schema-level security in release 4.5 or later, see the* **e*Gate Integrator System Administration and Operations Guide***.*

Role-Based Security

The ACL is defined through assigning roles to users. Privileges such as creating e*Ways, modifying ETDs, or reorganizing IQs are assigned to a role. In turn, roles are associated with one or more users. Users can then access objects according to the privileges they receive through the role.

When deploying an e*Gate environment, you define roles principally according to your users' predefined responsibilities; that is, what each user is expected to be able to do. To set up users, roles, and privileges, you must consider:

Access
Whether you wish to permit one or more users to access portions of the schema or the whole schema

Privileges
Whether you wish to permit one or more users to exercise all privileges (actions) or a portion of privileges pertaining to any component or category of components

The next section contains an example showing how to deploy the ACL security feature.

Example—Supply Chain Scenario

In this example, the Supply Chain System consists of these subsystems:

- Orders

- Inventory

- Shipping

Implementing this scenario involves the components shown in the following figure.

Table 6   Supply Chain Scenario Components

e*Way	Collaboration
Orders subsystem	
ewOrderIn	colOrderToInv
ewOrderOut	colCustResp
Inventory subsystem	
ewInventIn	colInventResp
ewInventOut	colInventQuery
Shipping subsystem	
ewShippingIn	colShipOrder
ewShippingOut	colShipOrderResp

In this example, you want "developer" users (use the e*Gate default Administrator role as a guide) to be able to make changes to the e*Ways and their Collaborations. However, you only want them to carry out these changes in their own departments.

Define the following developer roles:

- DevOrder

- DevInvent

- DevShipping

Then assign view and edit privileges to the e*Ways as follows:

- Orders subsystem e*Ways to the DevOrder role

- Inventory subsystem e*Ways to the DevInvent role

- Shipping subsystem e*Ways to the DevShipping role

Also, if desired, you can set up one or more "operator" roles for these departments in the same way (use the e*Gate default Operator role as a guide). You can then associate the remaining privileges (start, shutdown, suspend, continue, reload, and status) with the e*Ways and with each of your operator roles. Users with these roles are only able to do routine operations with the e*Ways and cannot make changes.

If you create a new role with view permissions associated with each component, the user will be able to change the logging levels and debug flags. This is especially useful for a "night administrator" or any other user that needs log level authority without allowing them to change the configuration.

Finally, you can assign only operator's roles to certain users, only developer's roles, or both roles to desired users. These users are then be able to control the e*Ways only according to their assigned roles.

5.6.3 Modeling Business Processes with e*Insight

Once you have e*Insight installed and running, along with e*Gate, this application can help you with the design phase of your deployment. You can use e*Insight to build graphical models of all your business processes.

The e*Insight Main window GUI offers you GUI tools you can use to create a working business-process model for your work flow. Using its Configuration mode, this application allows you to do the following operations:

- Create new business processes
- List these processes
- Diagram process models
- Document your business processes

Advantages of e*Insight Models

Once a business process model is in place, you can set properties and attributes for each activity then generate the components that make up the actual integration layer. Through this configuration process, the activities created in e*Insight automatically generate the appropriate e*Gate components. You can model and view a summary of the process flow or drill down to the detailed implementation of the activities.

eBusiness Integration

e*Insight has been designed and developed to provide tight, seamless integration of systems directly out of the box, to accelerate eBusiness Integration. Used with e*Insight, business process management requires little configuration and transformation development work in order to integrate back-office, Internet, or B2B applications into a total business process.

With e*Insight's automatic services in place, the more flexible process-driven integrations require no greater effort to implement than traditional messaging integrations.

Full UML Compliance

e*Insight is fully Unified Modeling Language (UML) compliant in its graphical display of business processes as activities, decision gates, sub-processes, and flow-control arrows.
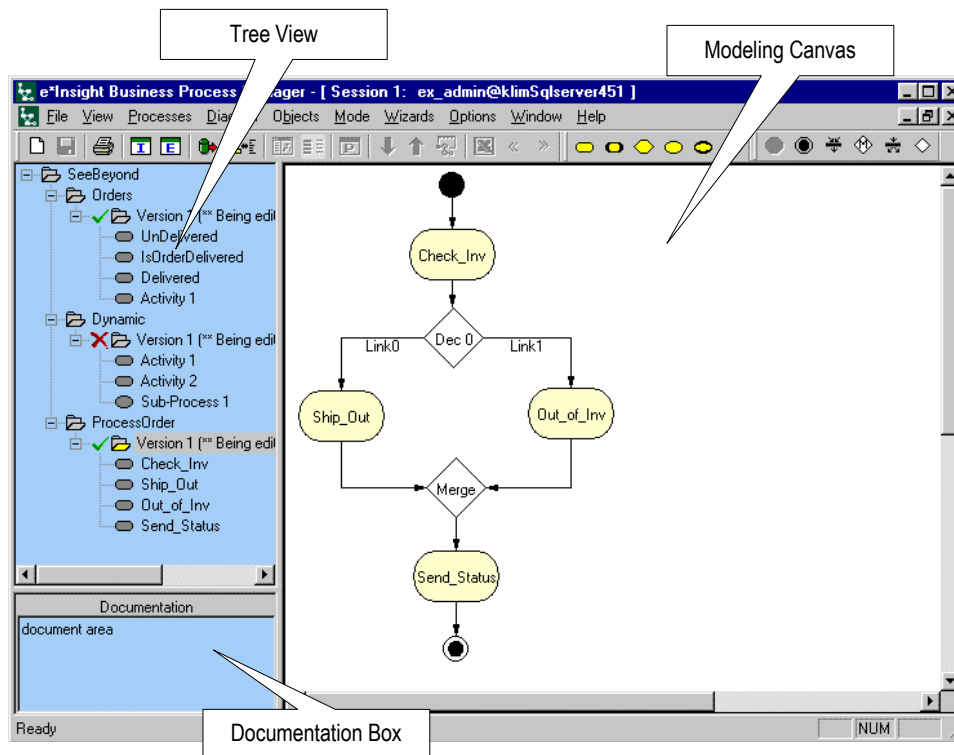
e*Insight Operating Modes

The e*Insight application has the following modes of operation:

- Design Mode (see **Figure 41 on page 125**)
- Monitoring Mode (see **Figure 71 on page 173**)

For more information on how e*Insight's Monitoring mode can help in your deployment operation, see **"Transition to Production" on page 172**.

Figure 41 e*Insight Main Window (Design Mode)

Use the e*Insight Configuration mode to create your graphical business process model. for more information on how to use this GUI, see the *e*Insight Business Process Manager User's Guide*.

e*Insight GUI Features

As shown in the previous figure, the Main window's Design Mode has the following GUI features:

- **Tree View** displays a hierarchical representation of all the business process models within e*Insight and their associated activities. This view allows you to see at a glance what is currently being displayed on the modeling canvas.

- **Documentation Box** manages comments and free-text descriptions about the business process version and its elements. This documentation is directly associated with the business processes overall and each of its activities can be used to support compliance documentation.

- **Modeling Canvas** is the portion of e*Insight where you graphically create the business process model in the form of a UML Activity diagram.

The ability to graphically model and implement a business process improves understanding of the process by both business and technical users while increasing the flexibility to adapt to change. As you elevate your business logic into the process layer, you can directly change business processes to reduce both the time to implement new processes and the effort to modify existing processes.

Design Management

As your business processes evolve in response to changes, the e*Insight database facilitates your efforts by managing all versions of business processes as they change over time. You can manage multiple versions of business process templates, exporting and importing them to and from e*Insight as needed.

Figure 42 shows a graphical example of an e*Insight business process model.

Figure 42   Business Process Model Example
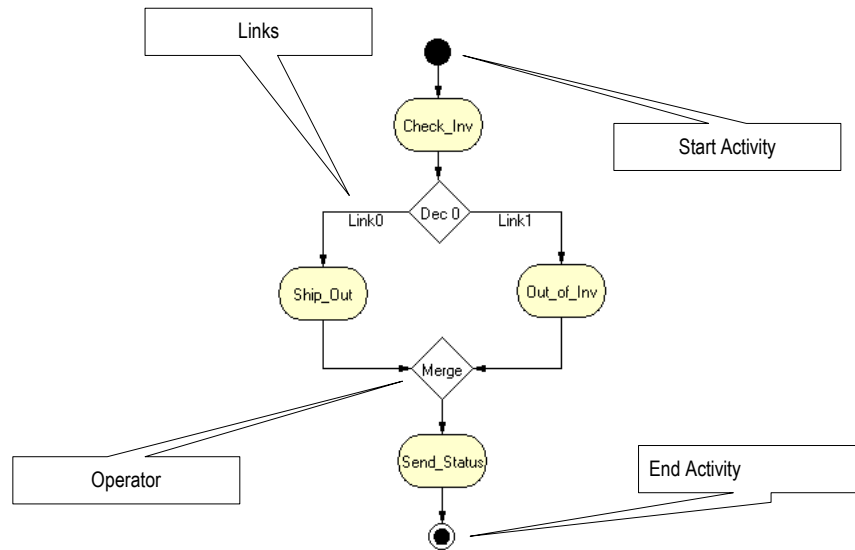
e*Gate Implementation

Once you have finished modeling your business processes using the e*Insight Configuration mode, you can then generate the e*Gate components from within e*Insight.

Automatic Component Generation

e*Insight automatically generates the reusable e*Gate integration components for each Activity needed to implement the actual business process. Because this implementation is automated (and therefore accelerated), no effort trade-off decisions are necessary. Direct system integration and process management become one.

Once you have created the business process, any system designer can set the Activity properties and generate the e*Gate integration schema to support the business process model's activities. Through this configuration process, the Activities created in e*Insight are transformed into either e*Ways or BOBs with Collaborations in e*Gate.

Transition from e*Insight to e*Gate

You can easily transition your business-process models from e*Insight to e*Gate, using the following e*Insight features:

- **e*Gate Configuration** GUI allows you to review the configuration settings for all the business process-related Activities and next choose which e*Gate components to generate.

- **Activity Properties** GUI allows you to configure how each specific Activity is implemented.

See the *e*Insight Business Process Manager Implementation Guide* for more information on implementing e*Insight features in e*Gate.

Defining Component Relationships

The following table clarifies the relationship of e*Insight components to their corresponding e*Gate components:

Table 7 Component Relationships: e*Gate to e*Insight

e*Insight Component	e*Gate Component
Business Process Template	Becomes a schema. A specific schema is a collection of e*Gate components (Hosts, e*Ways, IQs, and so on) configured to work together as a logical group. A schema can include the components for one or more business processes.
Activities	Become e*Ways, BOBs, and Event Types within a schema.
Operators	Include business logic that is run within the e*Gate e*Insight Engine.
Links	Become the publish/subscribe relationships that determine the routing between components and other participants.
Attributes	Become the input and output values used in Collaboration Rules to interact with external systems (external to e*Gate).
Business Process Instances	Become e*Gate Events, that is, actual messages being processed through the system.

After you have fully implemented your e*Insight business processes in e*Gate, you can monitor these processes as you watch them operate in real time.

Note: *For more information on e*Insight and e*Gate implementation, see* **Chapter 6***.*

For detailed information on implementing e*Insight with e*Gate, see the *e*Insight Business Process Manager Implementation Guide*.

5.6.4 Overview of e*Xchange Implementation

Implementing an e*Xchange system is the process of translating the vision of the business analyst into a functioning system. Once the analyst has determined that a certain business task must be accomplished with e*Xchange, it is the job of the implementor to make this a reality.

You implement e*Xchange by using the e*Xchange GUIs to enter the relevant data into the e*Xchange database. Then you combine the e*Xchange e*Gate components with other e*Gate components you add to create a complete e*Xchange schema.

The e*Xchange components are mostly pre-configured and do not require any (or very slight) modification by the implementor. The components that you add are completely user-defined. However, the e*Xchange GUIs and this guide provide a framework for integrating these user-defined components into a working e*Xchange system.

Types of e*Xchange Implementations

The e*Xchange system is designed for the large-scale integration of information systems, both inside and outside of an enterprise, in order to run and monitor business processes. The details of the business processes themselves depend on the nature of the business.

Not every business process takes advantage of every feature built into e*Xchange. Therefore, some e*Xchange implementations can use a simplified **eXSchema**.

Implementation Road Map

Clearly, each type of implementation involves a different approach. However, at a high level, there are certain similarities.

In general, the work of implementing an end-to-end scenario with e*Xchange involves taking what is created in e*Xchange and integrating it into a working e*Gate schema. e*Gate powers every e*Xchange scenario, and a successful e*Xchange implementation is dependent on a successful e*Gate implementation.

To give you an overview of the complete process, the implementation road map shown in **Figure 43 on page 129** contains high-level steps for a full e*Xchange implementation. This road map is further refined and given more detail in the sections that follow.

Figure 43   e*Xchange Implementation Road Map
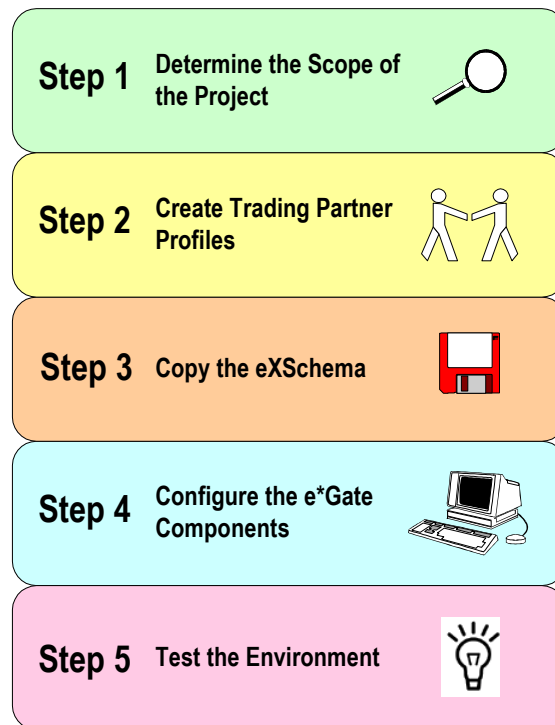
Step 1: Determine the Scope of the Project

Determine the type of implementation

The tasks involved in implementing e*Xchange differ depending on the type of implementation.

Analyze the business process

The business analyst must perform the standard tasks of analysis to develop a clear representation of the business process. It is a good idea to have diagrams of the process and a list of the data that must be tracked within the business process. Your detailed plans and diagrams provide highly beneficial starting points for working with the e*Xchange GUIs.

For more information on analyzing your business processes, see **Chapter 3**.

Note: *e*Insight can help you in analyzing and diagramming your business processes. See* **"e*Insight Deployment" on page 46** *for details.*

Step 2: Create Trading Partner Profiles

1   Create the custom validation Collaborations you need. For X12 protocol implementations, use the Validation Rules Builder tool to help create these validation Collaborations.

2   Enter the trading partner information into the e*Xchange database.

Step 3: Copy the eXSchema

When beginning an integration project, make a copy of the e*Xchange schema, **eXSchema**, that is installed from the CD-ROM. Do not make any modifications to **eXSchema** itself; keep it as a template. Make changes to the copy of the **eXSchema** that you create. Use this copy as your starting point in e*Gate for supporting e*Xchange.

Use the following procedure to create a copy of the **eXSchema**:

1 Open the **eXSchema** in the e*Gate Enterprise Manager GUI.

  ◆ Start the e*Gate Enterprise Manager.

  ◆ Log in to **eXSchema**.

2 Export the **eXSchema** to a file **c:\eGate\client***eXSchema backup file name*.

  ◆ Select **Export Schema Definitions to File** from the **File** pull-down menu.

  ◆ In the **Select archive File** dialog box enter *eXSchema backup file name* in the **File name** text box, then click **Save**.

3 Create a new schema using the eXSchema export file as a template.

  ◆ Select **New Schema** from the **File** pull-down menu.

  ◆ Enter *new e*Xchange schema name* in the text box.

  ◆ Mark the **Create from export** check box.

  ◆ Click **Find** and browse for the *eXSchema backup file name* file created in step 2.

  ◆ Click **Open**.

The Enterprise Manager creates a copy of the **eXSchema** with the schema name you entered.

Step 4: Configure the e*Gate Components

Configuring the e*Gate components forms the majority of the integration work done. In this step, you can:

  ▪ Add and configure the e*Ways that send data into and out of the e*Xchange system

  ▪ Make all user-configurable associations in the e*Gate GUI

Step 5: Test and Tune the System

It is a good idea to test the system in stages. For example, make sure that one activity works correctly before you try to run the entire business process. One good approach is to start with the "upstream" activities at the beginning of the business process, and work your way down to the last activity.

Once you have the entire system working, make adjustments, as necessary, to improve performance. For detailed information on all these implementation steps, see the *e*Xchange Partner Manager User's Guide* and *e*Xchange Partner Manager Implementation Guide*.

5.6.5 # Overview of e*Xpressway Implementation

This section contains lists of basic steps and considerations you and your trading partners must take into account to design your e*Xpressway implementation methodology.

Trading Exchange Web Site

e*Xpressway Integrator Server hosts the Trading Exchange Web site that provides the following functioning areas:

- **Administration**, where your administrator host controls Trading Exchange membership and the contents of the download packages

- **Public,** where trading partners can register and request information

- **Members-only,** where authorized trading partners can download and test the e*Xpressway customized OnRamp Solution Packages

Setting Up Your e*Xpressway Web Site

You, as the Trading Exchange customer, purchase e*Xpressway Integrator Server, allowing you to host e*Xpressway. This gives you full control over all hosting features.

Using the Server: e*Xpressway Integrator Server provides Web-based tools that let you manage your own Trading Exchange Web site. You can develop solutions for the transaction file format differences that may exist between you and your trading partners.

e*Xpressway Integrator OnRamp Overview

e*Xpressway Integrator OnRamp is an e*Gate-based application used by your trading partners to facilitate data transport. You must ensure that the OnRamp is customized to create an e*Xpressway Integrator OnRamp Solution Package for each of your trading partners. Once a customized Solution Package is uploaded to the Web site, a trading partner can then download and use it to access your Trading Exchange.

e*Xpressway Integrator OnRamp Schema

Each Solution Package is actually a small-footprint e*Gate schema (e*Gate version 4.5.2 or later) that enables communication from one URL to another, over the Internet. See **"Schema and Component Organization" on page 79** for more information on e*Gate schemas. Every Solution Package contains all the modules and configuration parameters needed to control, route, and transform data as it travels through the e*Gate environment. These Solution Packages are maintained by the Configuration Manager.

Web Site Management

The e*Xpressway Web site has the following tools that allow you to manage it on a continuing basis:

- **OnRamp Customization Management** page lets you view the OnRamp Solution Packages your Solution Provider (for more information this role, see **"Working with a Solution Provider" on page 132**) has uploaded, the trading partner to which

they are assigned, and the OnRamp Solution Package translation information. This page also lets you download or delete OnRamp Solution Packages and view notes.

▪ **OnRamp Solution Tools** page provides everything your Solution Provider needs to create OnRamp Solution Packages for additional trading partners, including the OnRamp software environment and all executables that Solution Providers need to create custom OnRamp solutions. The OnRamp Solution Tools consist of:

◆ **OnRamp template schema** are included in the base package to give Solution Providers a head start on creating made-to-order OnRamp Solution Packages.

◆ **Packager Tool executables** that your Solution Provider can use to create OnRamp Solution Packages for uploading to your Web site and downloading by your trading partners.

Working with a Solution Provider

Your Solution Provider (probably a consultant) must meet the following requirements:

▪ Unrestricted access to e*Gate, including the e*Ways

▪ Expertise in customizing e*Gate schemas

▪ Expertise and resources relevant to your likely trading partners' applications and platforms

Via your Web site, you provide the e*Xpressway Packager Tool to your Solution Provider. Using this tool, the Solution Provider can create OnRamp Solutions Packages for your trading partners.

Provide your Solution Provider with all the information and resources necessary to create the OnRamp solutions your trading partners need. Using the Packager Tool, the Solution Provider must create types of OnRamp Solution Packages for different categories of trading partners. These Solution Packages can be reused by all the trading partners who fall within the various categories.

Note: *Ensure that the Solution Provider thoroughly tests every OnRamp Solution Package before uploading it to the Web site. Complete testing before uploading ensures that trading partners do not have problems later on. Trading exchange sample schemas are also included on the CD-ROM for testing purposes.*

Trading Partners: Getting Started

The trading partners take the following general steps to begin using e*Xpressway:

1 Trading partners hear about e*Xpressway from the Trading Exchange and decide to use it. They are made aware of the URL for the Web site.

2 A trading partner enters the Web site's public area and applies to the Trading Exchange.

3 The trading partner receives approval (or denial) e-mail from you, depending on your decision.

4 The trading partner enters the Web site's member area and takes the following steps:

- A readiness assessment in preparation for installing

- Downloading and installing e*Xpressway Integrator OnRamp

- Selecting, downloading, and configuring a customized OnRamp Solution Package

- Returning to the Web site and signing up by supplying their e*Xpressway OnRamp URL

- Awaiting notification that they have had their URL added to the Trading Exchange

- Testing communication to and from your Trading Exchange

- Troubleshooting (if needed)

- Going live

5.7 Case Study Examples

The rest of this chapter provides the following examples of deploying the eBI Suite:

- **First Scenario** uses e*Gate and e*Insight; see **"Case Study 1: Web Order Scenario" on page 133**.

- **Second Scenario** uses e*Gate, e*Insight, and e*Xchange; see **"Case Study 2: Expanded Web Order Scenario" on page 139**.

- **Third Scenario** uses e*Gate with Java features, including the Guaranteed Exactly Once Delivery of Events; see **"Case Study 3: Tracking Timecards and Payroll Scenario" on page 144**.

- **Fourth Scenario** uses e*Gate with Java features, including ELS; see **"Case Study 4: Receiving and Purchasing Scenario" on page 149**.

Note: *See* **Chapter 6** *for a continuation of each of these examples into the transition-to-production phase of deployment.*

5.7.1 Case Study 1: Web Order Scenario

The eBI Suite provides the tools to create a supply-chain system that supports customer and corporate demands. In particular, e*Insight provides the ability to configure business rules through a GUI and to track individual items as they proceed through the supply chain. The scenario discussed in this section provides an example.

Background

Electronic commerce provides a competitive advantage to those companies that can provide a functioning electronic forum. Customers have come to expect rapid response time to orders processing and comprehensive orders tracking. An integration project that models business processes has a potentially high return on investment.

Modeling a business process like a supply chain frequently requires the integration of several different types of systems. The eBI Suite is ideally suited to the task because it is flexible, general, and centrally managed.

This section shows by example how to determine the components you can build to create a supply-chain system with e*Insight.

Functional Requirements

Electronic business process modeling applied to the supply chain for the Web Order scenario integrates these systems:

- Web orders
- Inventories
- Shipping
- Customer billing and accounting
- e*Insight database (configured as order tracking)
- Customer response

Designing Communication Topology

As explained earlier (see **"Identifying External Systems" on page 86**), communication topology is the relationship among external systems without regard for the connections provided by the eBI Suite. It is basically a representation of the communication requirements met by the system.

The requirements described under **"Functional Requirements" on page 134** lead to the Web Order scenario communication topology shown in Figure 44.

Figure 44 Communication Topology for Business Process Modeling: Case 1
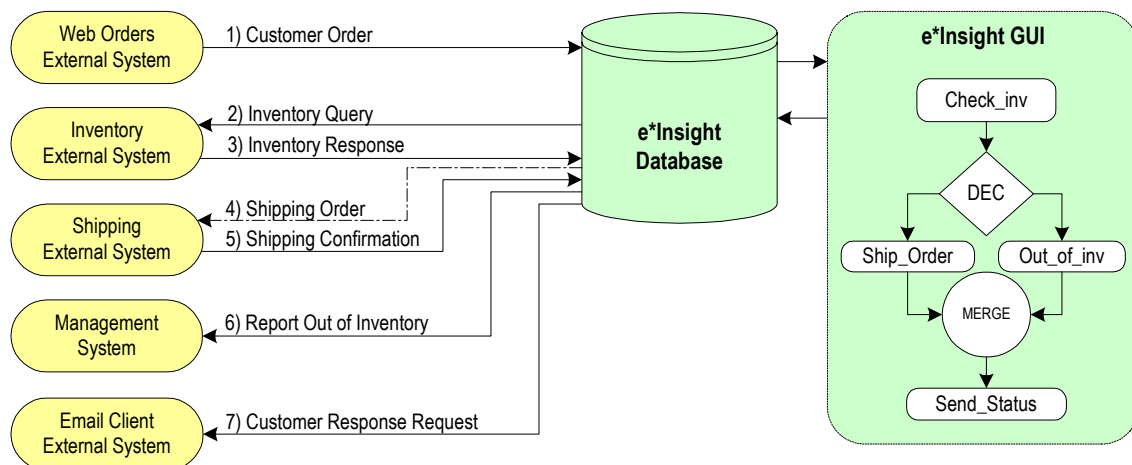
The communication topology shown in the previous figure includes an e*Insight database and GUI. Although these systems are included in the topology they are not separate items because they are all installed as part of the total eBI Suite.

Business Process Implementation

As shown in Figure 44, e*Insight implements the following business process:

1 The Web order system sends each customer order Event to the e*Insight database.

2 e*Insight sends a query Event to the inventory system to verify the availability of items desired.

3 The inventory system sends a response Event to the e*Insight database.

4 If goods are available, the e*Insight database sends a shipping-request Event to the shipping system.

5 The shipping system sends a response Event to e*Insight when the customer order is shipped.

6 The e*Insight database sends a billing-request Event to the accounting system.

7 The e*Insight database sends a customer-response-request Event to the customer response system to generate confirming email.

Designing Component Topology

A component topology is the relationship among the e*Gate, e*Insight, and e*Xchange components that support that topology. It is influenced by the Event Types that are exchanged among components.

The figure below shows the component topology in the Web Order scenario.

Figure 45 Component Topology for Business Process Modeling: Case 1
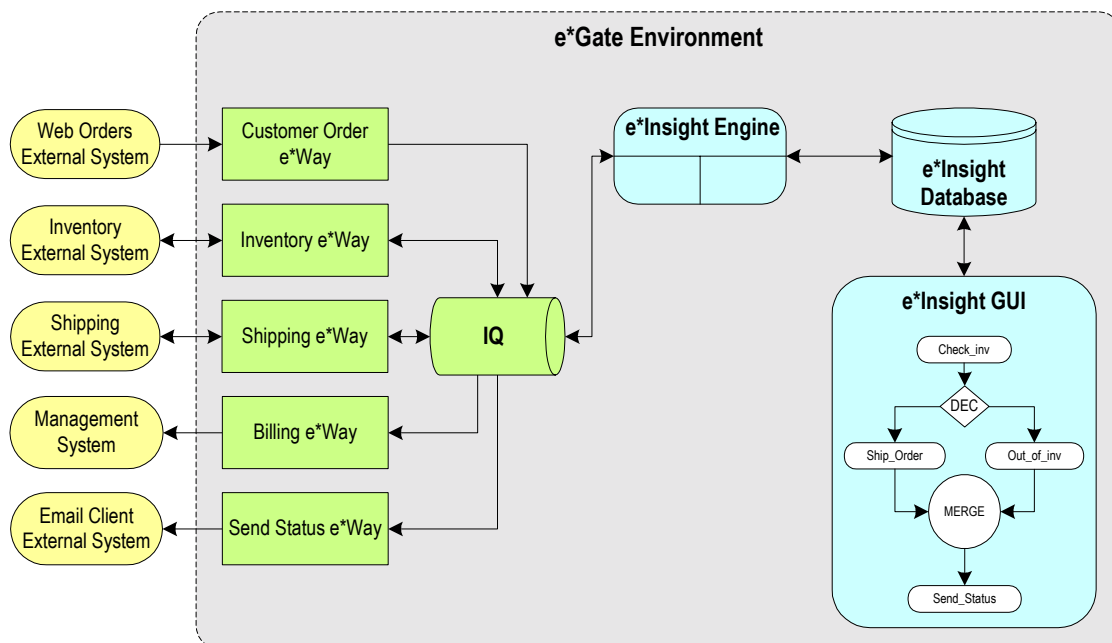
In the component topology shown in the previous figure, all e*Ways publish their data to the same IQ. This decision represents the simplest deployment but could also create a performance bottleneck. Two alternatives are:

- Create multiple IQs managed by one IQ Manager

- Create multiple IQs managed by more than one IQ Manager

If there is sufficient RAM to support more than one IQ Manager without causing memory swapping it would probably be advantageous to do so.

Collaboration Components

The final part of establishing the component topology is designing the following Collaboration components:

- ETDs

- Collaboration Rules/scripts (Monk and Java)

- Monk function definitions

ETDs: Each data flow in a diagram such as **Figure 44 on page 134** is associated with one or more Event Types. Along a specific data route, where the content of Events at several stages along the route is related, it is sometimes possible to use the same Event Type for more than one step along the route.

In the Web Order scenario example, the format of data exchanged between external systems is of the same type as that written to (or read from) the IQ. This is frequently not the case but leads to simplification of the overall system where it occurs.

Note that, for performance reasons (see **"Event Parsing" on page 92**), it is often advantageous to define *more* Event Types if by doing so you can define *simpler* Event Types that take less CPU time to parse, using less memory. The following figure lists the Event Types required by the Web Order scenario.

Table 8   ETDs for Case 1

Name of Event Types	Contents
etd_CustOrder	Order ID, customer information such as name; order information such as items and quantity.
etd_InventoryReq	Order ID, item, and quantity information.
etd_InventoryResp	Order ID and inventory response.
etd_ShippingReq	Order ID, customer information, item, and quantity information.
etd_ShippingResp	Order ID and shipping response.
etd_BillingReq	Order ID, customer information, item, quantity and price information.
etd_CustomerResp	Order ID, customer information, and response type (for example, order sent or items on back order).

Collaboration Rules: Collaborations take an input Event and convert it into an output Event. As described earlier, the input and output Event Types for each Collaboration are identical in the Web Order scenario example.

Typically, there is one Collaboration for each inbound Event and one Collaboration for each outbound Event. This amounts to seven Collaborations for the example.

Collaboration Rules and Event Types passed between the e*Insight engine and its database are not included in the next table because they are under the control of the e*Insight database. The table does list the Collaborations required for the Web Order scenario system.

Table 9   Collaborations for Case 1

Collaboration Rules	Input Event Types	Output Event Types
cr_CustOrderIn	etd_CustOrder	etd_CustOrder
cr_EXch2Invent	etd_InventoryReq	etd_InventoryReq
cr_Invent2EXch	etd_InventoryResp	etd_InventoryResp
cr_EXch2Shipping	etd_ShippingReq	etd_ShippingReq
cr_Shipping2EXch	etd_ShippingResp	etd_ShippingResp
cr_EXch2Billing	etd_BillingReq	etd_BillingReq
cr_EXch2CustResp	etd_CustomerResp	etd_CustomerResp

Monk function definitions: Each e*Way uses several Monk functions to define its specific behavior. The **Exchange Data with External** function is used to process data inbound from the external system. The **Process Outgoing Message** function is used to process data sent by the e*Way to the external system.

For more information on these functions, how to create them and how to configure the e*Way to use them, see the documentation specific to the e*Way being used.

The next table lists the e*Ways that need to have an **Exchange Data with External** function defined and provides sample functionality for what the Monk function is required to do.

Table 10   Monk Functions for Inbound e*Ways: Case 1

Inbound e*Ways	Exchange Data with External Functionality
Customer Order e*Way	Execute required business rules applicable to fetching customer orders from the Web server.
Inventory e*Way	Execute business rules applicable to transforming the query response provided by the inventory system.
Shipping e*Way	Execute business rules applicable to transforming the query response provided by the shipping system.

The next table lists the e*Ways that need to have a **Process Outgoing Message** function defined and provides sample functionality for what the Monk function can be required to do.

Table 11   Monk Functions for Outbound e*Ways: Case 1

Outbound e*Ways	Process Outgoing Message Functionality
Inventory e*Way	Execute business rules applicable to making a request of the inventory system.

Table 11   Monk Functions for Outbound e*Ways: Case 1 (Continued)

| Outbound e*Ways	Process Outgoing Message Functionality
Shipping e*Way	Execute business rules applicable to making a request to ship customer goods.
Billing e*Way	Execute business rules applicable to making a request to bill the customer.
Send Status e*Way	Execute business rules applicable to sending a response to the customer.

Designing Hardware Topology

The hardware topology is the distribution of components across hardware systems. In the e*Insight example, the database must segregated from the e*Ways, for reasons described earlier. As a result, you need two Participating Hosts:

- The first Participating Host runs the database software required to support the e*Insight database. Because of the close relationship between the e*Insight database, its engine, and GUI, it is best to run all these components on the same host.

- The second Participating Host runs the e*Ways that communicate with the external systems and the IQ Manager.

Figure 46 shows the final configuration.

Figure 46   Hardware Topology for Business Process Modeling: Case 1
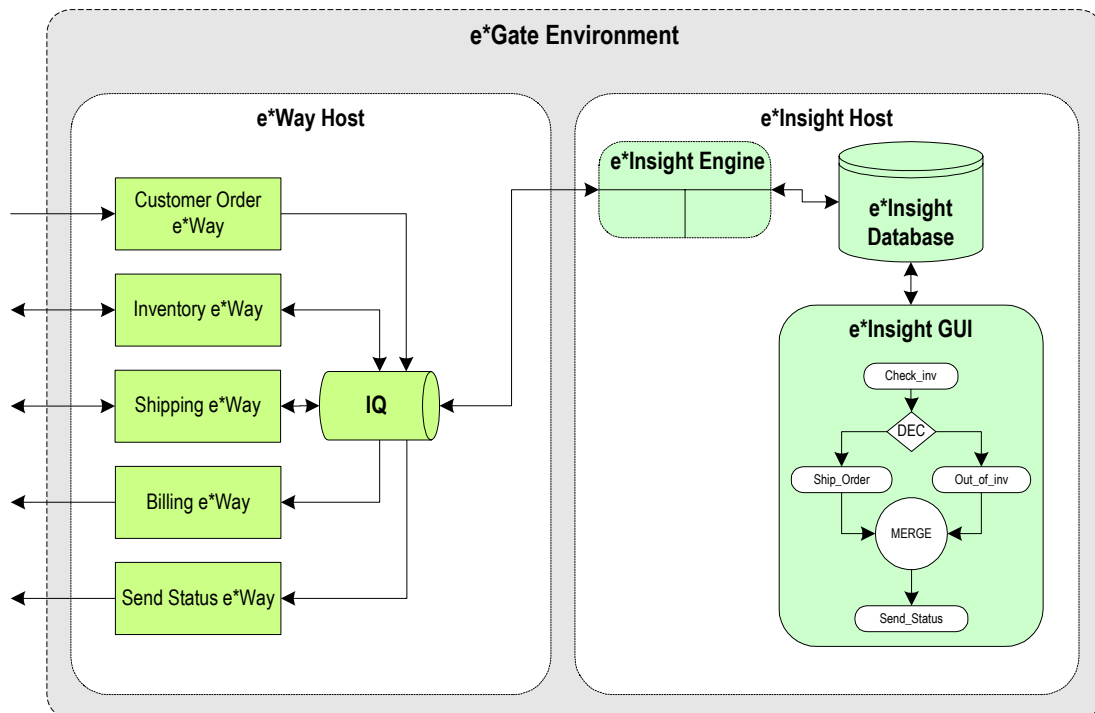
5.7.2 **Case Study 2: Expanded Web Order Scenario**

Electronic commerce involves not only integrating systems within an organization but also systems between organizations. In other words, transmitting information between trading partner is an important type of integration that can bring additional benefits. The example provided in this section expands the previous scenario to create a new scenario including this integration.

Background

As before, the e*Insight system creates and configures the e*Ways that support the business process that is modeled. This case study builds on the previous example by showing how the e*Insight engine module is also incorporated into the supply-chain project to provide secure data exchange with security organizations.

Functional Requirements

Electronic business process modeling applied to the supply chain for the Expanded Web Order scenario integrates these systems:

- Web order
- Inventory
- Shipping
- Customer billing and accounting
- e*Insight database (configured as order tracking)
- Customer response
- Trading partner manager

Designing Communication Topology

As explained earlier (see **"Designing Communication Topology" on page 134**) communication topology is the relationship among external systems without regard for the connections provided by the eBI Suite. It is basically a representation of the communication requirements met by the system.

There is an additional external system that must be integrated into the scenario, using e*Xchange. This application is the system connecting to the trading partner's systems. Incorporating the trading partner's external systems leads to the communication topology in Figure 47.

Figure 47 Communication Topology for Business Process Modeling: Case 2
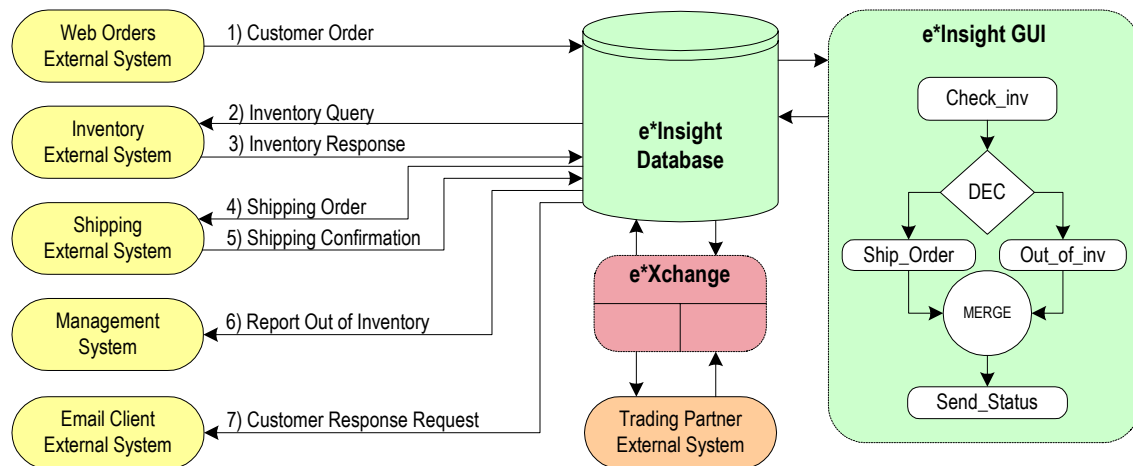
The communication topology shown in the previous figure includes the e*Insight database, the e*Insight GUI, e*Xchange, and the trading partner (external system) components.

Business Process Implemented

As Figure 47 shows, the eBI Suite implements the following business process:

1 The Web order system sends a customer-order Event to the e*Insight database.

2 e*Insight sends a query Event to the inventory system to verify the availability of the items desired.

3 The inventory system sends a response Event to the e*Insight database.

4 If goods are available, the e*Insight database sends a shipping-request Event to the shipping system.

5 The shipping system sends an Event to e*Insight when the customer order is shipped.

6 If goods are unavailable, the e*Insight database sends an Event to the management system informing it of the state of the inventory-and-billing-request Event to the accounting system.

7 If specific goods are unavailable, an automatic order is placed to one of the trading partners by sending an Event to the external system servicing the trading partner's system. The e*Xchange application manages communications with the trading partner.

8 The e*Insight database sends a customer-response-request Event to the customer response system to generate confirming e-mail.

Designing Component Topology

A component topology is the relationship among the e*Gate and e*Insight components that support that topology. It is influenced by the Event Types that are exchanged among components.

The following figure shows the component topology in the Expanded Web Order scenario.

Figure 48   Component Topology for Business Process Modeling: Case 2
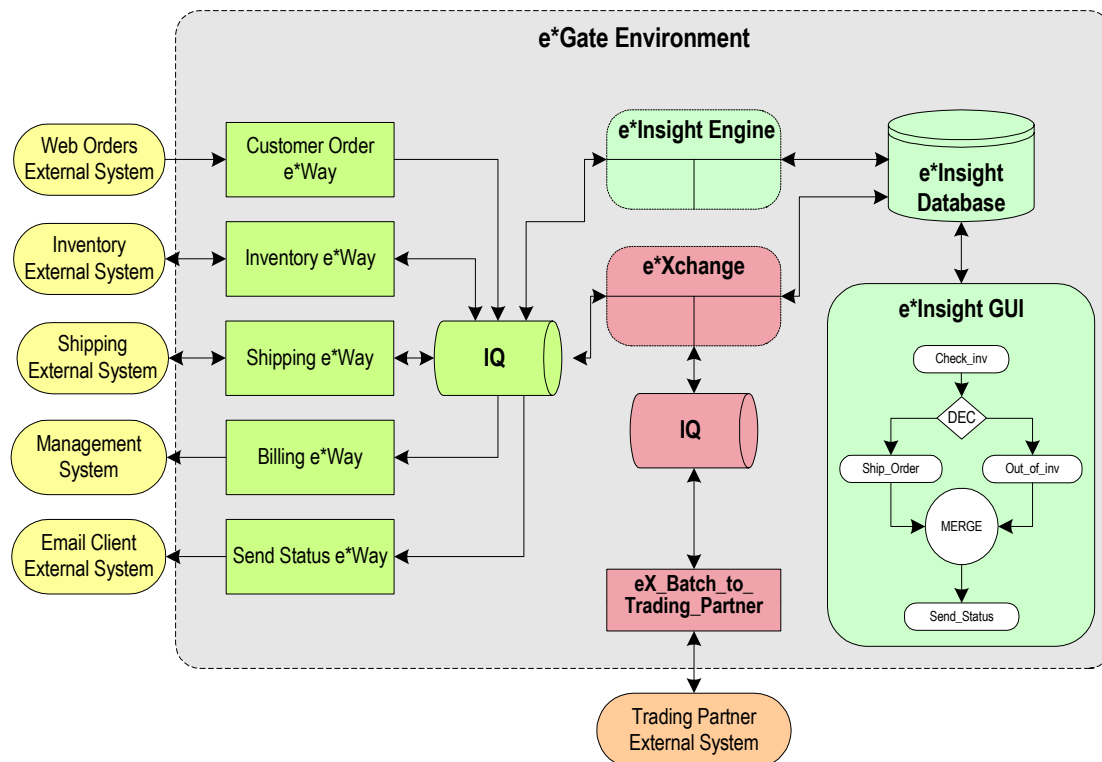
In the component topology shown in the previous figure, all e*Ways originally created by e*Insight publish their data to one IQ. e*Xchange and its supporting components communicate through a second IQ. Also, e*Xchange supports communication with the trading partner via an additional e*Way.

In terms of the hardware topology, it makes logical sense for the two IQs to exist on separate systems. Therefore, there are at least two IQ Managers, also on separate systems.

Collaboration Components

The final part of establishing the component topology is describing the following Collaboration components:

- ETDs
- Collaboration Rules/scripts (Monk and Java)
- Monk function definitions

ETDs: Each data flow in a diagram such as **Figure 47 on page 140** is associated with one or more Event Types. Along a specific data route, where the content of Events at several stages along the route is related, it is sometimes possible to use the same Event Type for more than one step along the route.

In the Expanded Web Order scenario example, the format of data exchanged between external systems is of the same type as that written to (or read from) the IQ. This is frequently not the case but leads to simplification of the overall system where it occurs.

Note that, for performance reasons (see **"Event Parsing" on page 92**), it is often advantageous to define *more* Event Types if by doing so you can define *simpler* Event Types that take less CPU time to parse using less memory. The following table lists the Event Types required by the Expanded Web Order scenario.

Table 12   ETDs for Case 2

Name of Event Types	Contents
etd_CustOrder	Order ID, customer information such as name; order information such as items and quantity.
etd_InventoryReq	Order ID, item, and quantity information.
etd_InventoryResp	Order ID and inventory response.
etd_ShippingReq	Order ID, customer information, item, and quantity information.
etd_ShippingResp	Order ID and shipping response.
etd_BillingReq	Order ID, customer information, item, quantity and price information.
etd_CustomerResp	Order ID, customer information, and response type (for example, order sent or items on back order).
etd_Batch	Inventory ID and information about goods supplied by the trading partner.

Collaboration Rules: Collaborations take an input Event and convert it into an output Event. As described earlier, the input and output Event Types for each Collaboration are identical in the Expanded Web Order scenario example.

Typically, there is one Collaboration for each inbound Event and one Collaboration for each outbound Event. This amounts to seven Collaborations for the Expanded Web Order scenario.

Collaboration Rules and Event Types passed between the e*Insight engine and the e*Insight database are not included in the next table because they are under the control of the e*Insight database. This table does list the Collaborations required for the Expanded Web Order scenario system.

Table 13   Collaborations for Case 2

Collaboration Rules	Input Event Types	Output Event Types
cr_CustOrderIn	etd_CustOrder	etd_CustOrder
cr_EXch2Invent	etd_InventoryReq	etd_InventoryReq
cr_Invent2EXch	etd_InventoryResp	etd_InventoryResp
cr_EXch2Shipping	etd_ShippingReq	etd_ShippingReq
cr_Shipping2EXch	etd_ShippingResp	etd_ShippingResp

Table 13 Collaborations for Case 2 (Continued)

Collaboration Rules	Input Event Types	Output Event Types
cr_EXch2Billing	etd_BillingReq	etd_BillingReq
cr_EXch2CustResp	etd_CustomerResp	etd_CustomerResp

Monk function definitions: Each e*Way uses several Monk functions to define its specific behavior. The **Exchange Data with External** function is used to process data inbound from the external system. The **Process Outgoing Message** function is used to process data sent by the e*Way to the external system.

For more information on these functions, how to create them and how to configure the e*Way to use them, see the documentation specific to the e*Way being used.

The next table lists the e*Ways which need to have an **Exchange Data with External** function defined and provides sample functionality for what the Monk function can be required to do.

Table 14 Monk Functions for Inbound e*Ways: Case 2

Inbound e*Ways	Exchange Data with External Functionality
Customer Order e*Way	Execute required business rules applicable to fetching customer orders from the Web server.
Inventory e*Way	Execute business rules applicable to transforming the query response provided by the inventory system.
Shipping e*Way	Execute business rules applicable to transforming the query response provided by the shipping system.

The following table lists the e*Ways that need to have a **Process Outgoing Message** function defined and provides sample functionality for what the Monk function can be required to do.

Table 15 Monk Functions for Outbound e*Ways: Case 2

| Outbound e*Way	Process Outgoing Message Functionality
Inventory e*Way	Execute business rules applicable to making a request of the inventory system.
Shipping e*Way	Execute business rules applicable to making a request to ship customer goods.
Information e*Way	Execute business rules applicable to informing management about shortages in inventory.
Send Status e*Way	Execute business rules applicable to sending a response to the customer.

Designing Hardware Topology

The hardware topology is the distribution of components across hardware systems. In the e*Insight example, the database must be segregated from the e*Ways, for reasons

described earlier. Also, communicating with the trading partner forms another logical separation. As a result, you need three Participating Hosts:

- The first Participating Host runs the database software required to support the e*Insight database. Because of the close relationship between the e*Insight database, its engine, and GUI, it is best to run all these components on the same host.

- The second Participating Host runs the e*Ways that communicate with the external systems and the IQ Manager.

- The third Participating Host runs e*Xchange, as well as the e*Ways and IQ Manager that enable communication with the trading partner.

Figure 49 shows the final configuration.

Figure 49   Hardware Topology for Business Process Modeling: Case 2
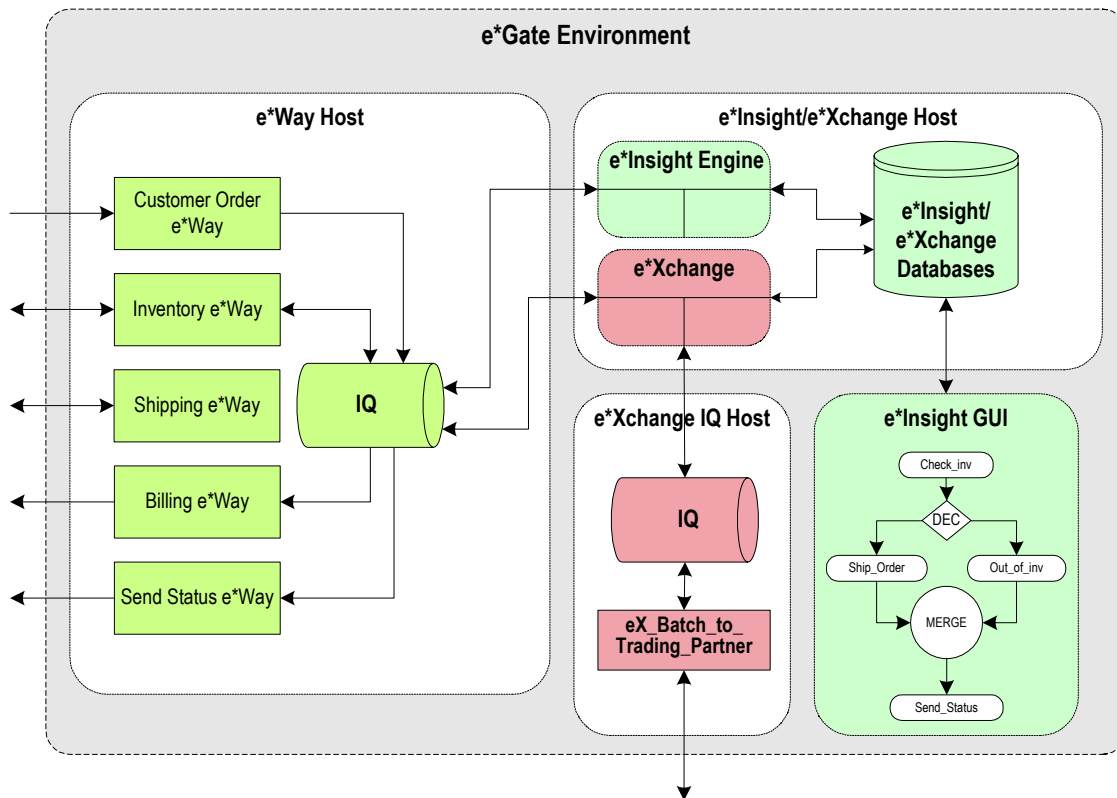
5.7.3 Case Study 3: Tracking Timecards and Payroll Scenario

A company's Human Resources (HR) department needs a more effective system for keeping track of employees' hours, their timecards, and ensuring accurate payroll. The department has two systems for performing these operations, but these systems are extremely dissimilar. Data must be delivered accurately for payment to be exact. This scenario shows how e*Gate can help solve these problems using the Java-based Collaborations.

Background and Functional Requirements

This HR Department has two systems. System A keeps track of the hours employees work per week; System B is responsible for paying them. The data from system A is in a delimited format and must be converted to B's fixed format. In addition, system B can only accept four fields: first name, last name employee number and amount. system A does not have an "amount" field, so one must also calculate the value for the amount field before sending it to B.
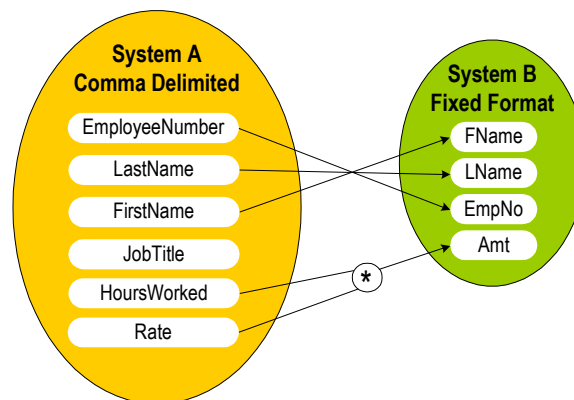
e*Gate Solution

The proposed e*Gate solution makes use of the e*Gate release 4.5 (and later) enhanced Java Collaboration Service to transform the data from system A's format to system B's format. e*Gate is very flexible about where the actual transformation processing can occur as the data moves from system A to system B.

Designing Communication Topology

The following figure graphically depicts the data to be sent from a simple timecard system to a simple payroll system, showing the HR Department's business need.

Figure 50   HR Business Need

Designing Component Topology

This solution uses the Multi-Mode e*Way as the main transformation component and two simple file e*Ways to bring data into and send data out from the e*Gate environment. Figure 51 shows all the components and their relationships in the e*Gate schema.

Figure 51 Tracking Timecards and Payroll Scenario Overview
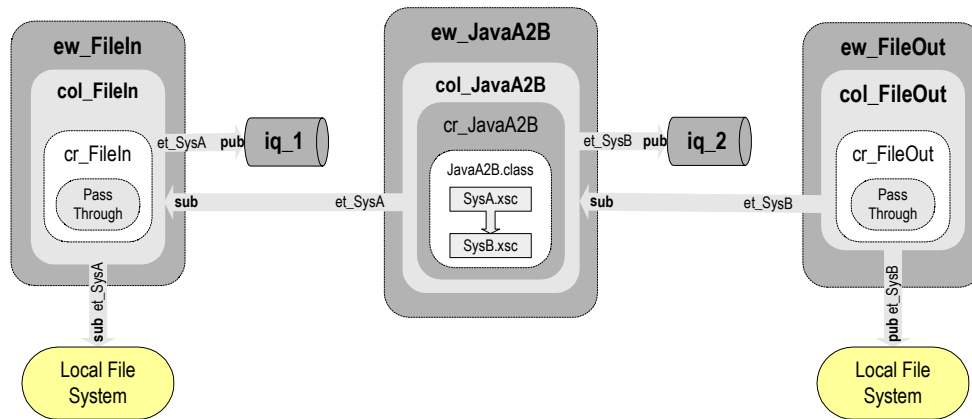
Notes on the Tracking Timecards and Payroll Scenario Overview

1 **ew_FileIn** brings data from System A into e*Gate.

   The **col_FileIn** Collaboration in the **ew_FileIn** e*Way subscribes to a location on the local file system. It polls this location for a text file with extension **".fin"** containing data from System A, packages the data as a **SysA** Event, and then publishes the Event to the **iq_1** IQ.

2 **ew_JavaA2B** (Multi-Mode e*Way) changes the data format and calculates the amount.

   The **col_JavaA2B** in the **ew_JavaA2B** subscribes to **SysA** Events published by **col_FileIn**. It uses the Java Collaboration Rule **cr_JavaA2B** to change **SysA** Events into **SysB** Events. This rule uses the **JavaA2B.class** which implements the transformation. **cr_JavaA2B** also computes the **SysB** amount field by multiplying the hours and the rate from **SysA**. Finally, **col_JavaA2B** publishes the **SysB** Event to the **iq_2** IQ.

3 **ew_FileOut** writes the transformed data out to local file system.

   The **cr_FileOut** Collaboration in the **ew_FileOut** e*Way subscribes to **SysB** Events published by **col_JavaA2B**. The **cr_FileOut** Collaboration Rule uses the Pass Through service to move the data without modifying it. When a **SysB** Event is retrieved, the e*Way packages it as a text file and writes it to the specified location on the local file system, completing the end-to-end scenario.

The following table lists all the components for the schema. Substitute the name of the system running the schema for *host-name* where applicable.

Table 16 Tracking Timecards and Payroll Scenario Components

Component	Logical Name	Settings
Schema	JavaE2E	
Control Broker	*host-name*_cb	
IQ Manager	*host-name*_iqmgr	Start Up = Auto
Event Type	et_SysA	
	et_SysB	

Table 16 Tracking Timecards and Payroll Scenario Components (Continued)

Component	Logical Name	Settings
Java ETD	SysA.xsc	Package Name = SysApackage
	SysB.xsc	Package Name = SysBpackage
Collaboration Rule	cr_FileIn	Service = Pass Through Sub = et_SysA Pub = et_SysA
	cr_JavaA2B	Service = Java Instance1 et_SysA In Trigger Instance2 et_SysB Out
	cr_FileOut	Service = Pass Through Sub = et_SysB Pub = et_SysB
Java Collaboration Rule Class	JavaA2B.class	Source = Instance1 Destination = Instance2
Inbound e*Way	ew_FileIn	Executable = stcewfile.exe Config file = ew_FileIn.cfg Start Up = Auto Collaboration = col_FileIn
Outbound e*Way	ew_FileOut	Executable = stcewfile.exe Config file = ew_FileOut.cfg Start Up = Auto Collaboration = col_FileOut
Multi-Mode e*Way	ew_JavaA2B	Executable = stceway.exe Config file = ew_JavaA2B.cfg Start Up = Auto Collaboration = col_JavaA2B
IQ	iq_1	Service = STC_Standard
	iq_2	Service = STC_Standard
Collaboration	col_FileIn	Collab Rule = cr_FileIn Sub = et_SysA from <EXTERNAL> Pub = et_SysA to iq_1
	col_JavaA2B	Collab Rule = cr_JavaA2B Sub = et_SysA from col_FileIn Pub = et_SysB to iq_2
	col_FileOut	Collab Rule = cr_FileOut Sub = et_SysB from col_JavaA2B Pub = et_SysB to <EXTERNAL>

Setting Up the Scenario

Use the following general steps to design and set up this scenario:

- Create the Event Types and Java ETDs.

- Create the Collaboration Rules and Java Collaboration Rules Classes.

- Add and configure the e*Ways, BOBs, and IQs.

- Add and define the Collaborations that route the data.

Note: *For complete information on how to create and configure e*Gate's Java-enabled components, see the e*Gate Integrator User's Guide.*

Creating Event Types and Java ETDs

This scenario uses two Event Types, each with its own ETD. The first Event Type, **et_SysA**, models the comma-delimited format of the data received from System A. The second Event Type, **et_SysB**, models the fixed-length data format required by System B.

In addition, you need to create and configure the following ETDs:

- **SysA.xsc**
- **SysB.xsc**

You need to create nodes in your **SysA.xsc** ETD (corresponding to System A) to track the following data:

- LastName
- FirstName
- JobTitle
- HoursWorked
- Rate

You also need to create nodes in your **SysB.xsc** ETD (corresponding to System B) to track the data shown in the following table:

Table 17   SysB ETD Fixed-Node Properties

Node Name	Structure	Length	Offset
FName	fixed	25	0
LName	fixed	25	25
EmpNo	fixed	10	50
Amt	fixed	10	60

Creating the Collaboration Rules and Java Collaboration Rules Classes

This scenario uses three Collaboration Rules: two Pass Through rules and one that uses a Java Collaboration Rule Class. The Pass Through rules, **cr_FileIn** and **cr_FileOut**, are used to route the Events through the e*Gate environment and the Java Collaboration Rule **cr_JavaA2B** is used to transform the Event from Event Type **et_SysA** to **et_SysB**.

Note: *The procedure for creating a Java Collaboration Rule is different from creating Monk Collaboration Rules. For complete information on creating and configuring all Collaboration Rules, see the **e*Gate Integrator User's Guide**.*

You must create the following components:

- **cr_FileIn** and **cr_FileOut** Collaboration Rules
- **JavaA2B.class** in the Java Collaboration Rules Editor

Adding the e*Ways

You must add and configure:

- **ew_FileIn** file e*Way
- **ew_FileOut** file e*Way
- **ew_JavaA2B** Multi-Mode e*Way

Adding the IQs

Add the following IQs:

- **iq_1**
- **iq_2**

Adding the Collaborations

Add and configure the following Collaborations:

- **col_FileIn**
- **col_JavaA2B**
- **col_FileOut**

After you have completed these steps, the HR Department's schema is finished.

Designing Hardware Topology

The hardware topology is the distribution of components across hardware systems. In this scenario, the desired operations must correlate and synchronize the appropriate data.

Hardware topology for this scenario would be similar to that given for the **"Designing Hardware Topology" on page 159**. See that section for additional details.

5.7.4 Case Study 4: Receiving and Purchasing Scenario

A company's Sales Department needs a way to track whether an entire purchase order for office supplies has been filled. The problem to solve with this deployment is how to correlate individual line items ordered by Purchasing with those taken in by Receiving. This scenario shows how e*Gate's ELS feature can be used to fulfill this requirement.

Background

This company's receiving-and-purchasing system operates as follows:

- Purchasing buys office supplies for the company. The orders generated by the purchasing system have multiple line items.

- Receiving processes the items as they are delivered.

- Typically, only individual line items are delivered, not entire purchase orders.

- Purchasing only pays purchase orders that are completely fulfilled.

Functional Requirements

The following figure shows a general diagram of how the Sales Department's office-supply receiving/purchasing process must operate to completely track the receiving and purchasing of these supplies for the Sales Department.

Figure 52 Receiving and Purchasing Business Requirements
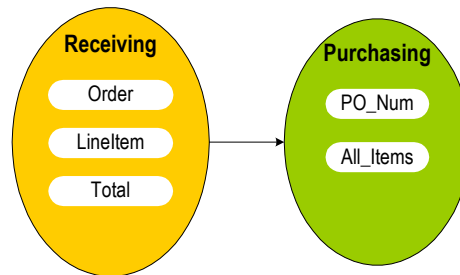
Solution Provided by ELS

The proposed e*Gate solution makes use of the e*Gate release 4.5 (and later) enhanced Java Collaboration Service and the ELS methods added to the Java Collaboration. These features allow you to combine the data from Events that have the same order number into a single Event. e*Gate is flexible about where the actual ELS processing can occur, as the data moves from the receiving system to the purchasing system.

Designing Communication Topology

As explained earlier (see **"Designing Communication Topology" on page 134**) communication topology is the relationship among external systems without regard for the connections provided by the eBI Suite. It is basically a representation of the communication requirements met by the system.

In the case of the Sales Department, when all the line items for a specific purchase order have all been received, a combined Event can be sent to Purchasing. Purchasing uses the receipt of this Event to trigger payment of the purchase order (see **Figure 53 on page 151**).
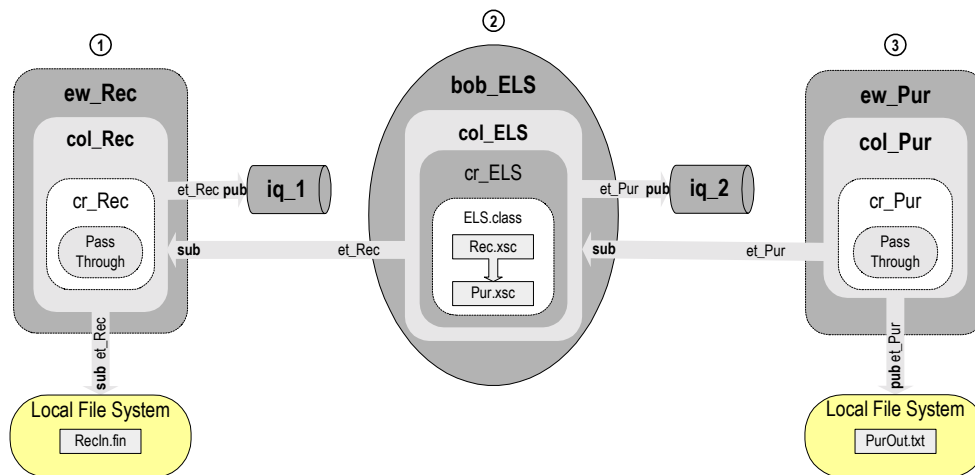
Figure 53 ELS Solution
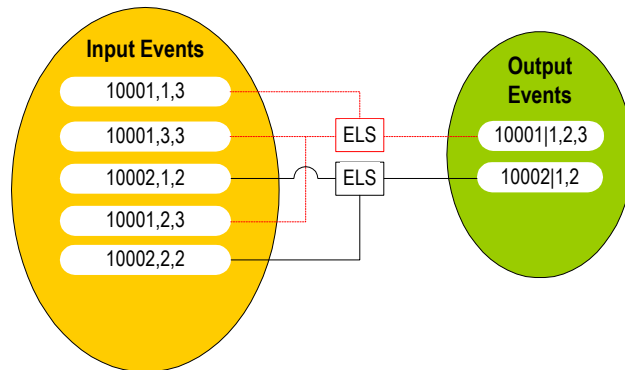
Designing Component Topology

The following figure shows all the components and their relationships to one another in the complete e*Gate schema for the Sales Department's receiving and purchasing scenario.

Figure 54 Receiving and Purchasing Scenario Overview

The solution in this deployment uses a BOB component as the main transformation component and two simple file e*Ways to bring data into and send data out from the e*Gate environment.

Notes on Receiving and Purchasing Scenario Overview

① **ew_Rec** brings data from the receiving system into e*Gate.

The **col_Rec** Collaboration in the **ew_Rec** e*Way subscribes to a location on the local file system. It polls this location for a text file with extension **".fin"** containing data from the receiving system, packages the data as an **et_Rec** Event then publishes the Event to the **iq_1** IQ.

② **bob_ELS** combines Events with matching order numbers into a single Event.

The **col_ELS** in the **bob_ELS** subscribes to **et_Rec** Events published by **col_Rec**. It uses an ELS-enabled Java Collaboration Rule **cr_ELS** to temporarily store **et_Rec** Events. Once all the Events associated with a specific order are received, **cr_ELS** publishes a combined Event, **et_Pur**, to the **iq_2** IQ. This Collaboration Rule uses the **ELS.class** file that implements the business logic.

③ **ew_Pur** writes the combined Event out to local file system.

The **col_Pur** Collaboration in the **ew_Pur** e*Way subscribes to **et_Pur** Events published by **col_ELS**. The **cr_Pur** Collaboration Rule uses the Pass Through service to move the data without modifying it. When a **et_Pur** Event is retrieved, the e*Way packages it as a text file and writes it to the specified location on the local file system, completing the end-to-end scenario.

Table 18 lists all the components for the schema. Substitute the name of the system running the schema for the *host-name* where applicable.

Table 18 Receiving and Purchasing Scenario Components

Component	Logical Name	Settings
Schema	ELS_E2E	
Control Broker	*host-name*_cb	
IQ Manager	*host-name*_iqmgr	Start Up = Manual
Event Type	et_Rec	
	et_Pur	
Java ETD	Rec.xsc	Package Name = RecPackage
	Pur.xsc	Package Name = PurPackage
Collaboration Rule	cr_Rec	Service = Pass Through Sub = et_Rec Pub = et_Rec
	cr_ELS	Service = Java Instance Name = Root_In ETD = Rec.xsc Mode = In Trigger = Yes Manual Publish = No Instance Name = Root_Out ETD = Pur.xsc Mode = Out Trigger = No Manual Publish = No
	cr_Pur	Service = Pass Through Sub = et_Pur Pub = et_Pur
Java Collaboration Rule Class	ELS.class	Source = Root_In (Rec) Destination = Root_Out (Pur)

Table 18 Receiving and Purchasing Scenario Components (Continued)

Component	Logical Name	Settings
Inbound e*Way	ew_Rec	Executable = stcewfile.exe Config file = ew_Rec.cfg Start Up = Manual Collaboration = col_Rec
Outbound e*Way	ew_Pur	Executable = stcewfile.exe Config file = ew_Pur.cfg Start Up = Manual Collaboration = col_Pur
BOB	bob_ELS	Start Up = Manual Collaboration = col_ELS
IQ	iq_1	Service = STC_Standard
	iq_2	Service = STC_Standard
Collaboration	col_Rec	Collab Rule = cr_Rec Sub = et_Rec from <EXTERNAL> Pub = et_Rec to iq_1
	col_ELS	Collab Rule = cr_ELS Sub = et_Rec from col_Rec Pub = et_Pur to iq_2
	col_Pur	Collab Rule = cr_Pur Sub = et_Pur from col_ELS Pub = et_Pur to <EXTERNAL>

Setting Up the Scenario

Use the following general steps to design and set up this receiving and purchasing scenario using Java and ELS:

- Create the Event Types and Java ETDs.
- Create the Collaboration Rules and Java Collaboration Rules Classes.
- Add and configure the e*Ways, BOBs, and IQs.
- Add and define the Collaborations that route the data.

The rest of this section explains how to design and set up the basic components of this Sales Department's business scenario. For complete information on how to create and configure e*Gate's Java-enabled components, see the *e*Gate Integrator User's Guide*.

Creating the Schema

This scenario would typically be part of a larger e*Gate environment and deployment. However, to help you keep track of this operation separately, you can set it up as a separate schema, as it is shown in this example. An e*Gate deployment can consist of one, two, or many schemas, depending on your system design.

Creating Event Types and Java ETDs

This scenario uses two Event Types, each with its own ETD. The first Event Type, **et_Rec**, models the format of the data sent by the receiving system. The second Event Type, **et_Pur**, models the format used by the purchasing system.

The following figure shows where these components fit into the collection of interrelated components that make up the finished schema.

Figure 55   Event Types and Java ETDs
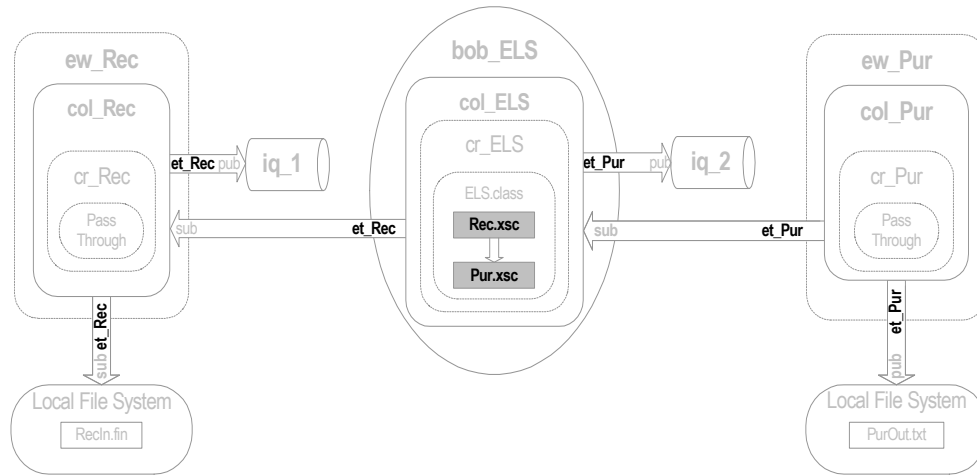
You must create the following ETDs:

- **Rec.xsc**
- **Pur.xsc**

Use the Enterprise Manager's Java ETD Editor to create these ETDs. When finished the completed **Rec** ETD looks like the one shown in the following figure.

Figure 56   Java ETD Editor—Completed Rec ETD
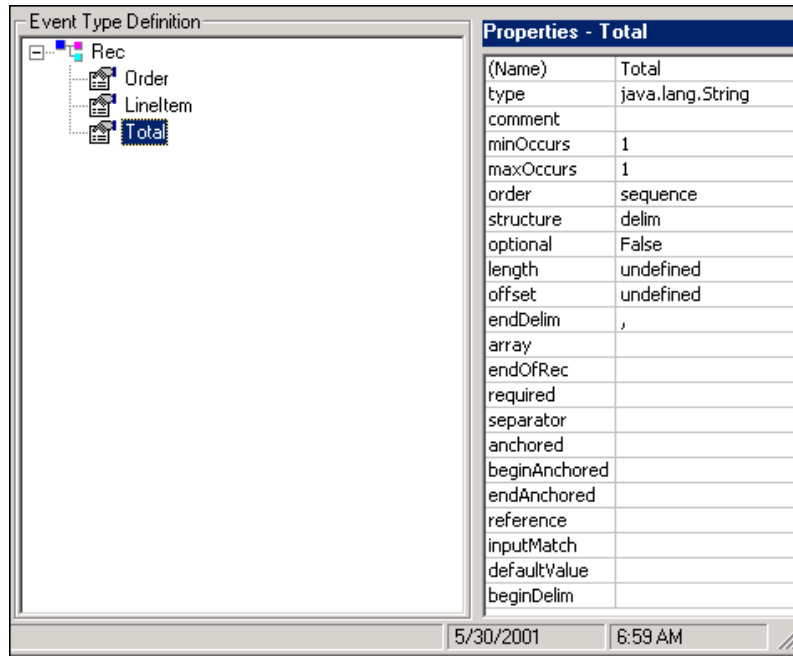
Figure 57 below shows the **Pur** ETD.

Figure 57   Java ETD Editor—Completed Pur ETD
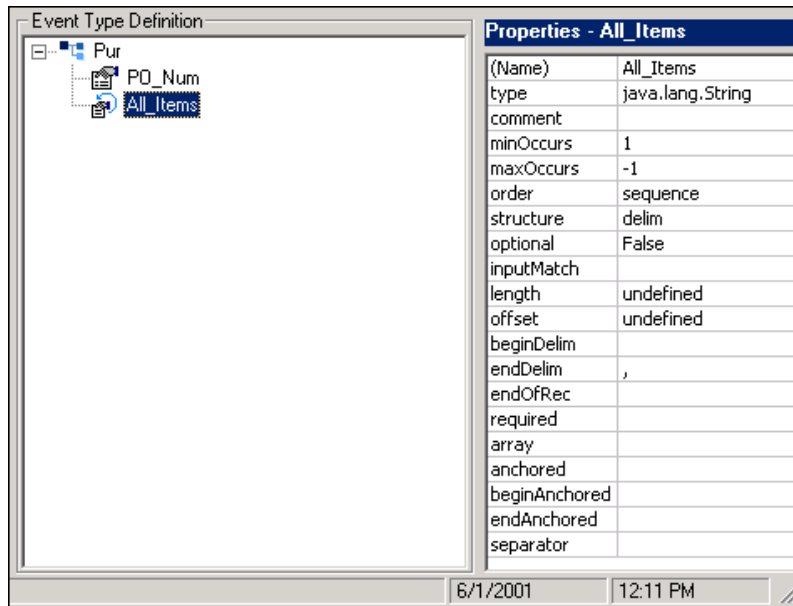
Creating the Collaboration Rules and Java Collaboration Rules Classes

This scenario uses the following Collaboration Rules:

- **cr_Rec**
- **cr_Pur**
- **cr_ELS**

The rules **cr_Rec** and **cr_Pur** are Pass Through rules and are used to route the Events through the e*Gate environment. The Java Collaboration Rule **cr_ELS**, which uses a Java Collaboration Rule Class (CRC), is used to combine the individual line item Events into a single complete purchase order Event.

The following figure shows where these parts fit into the collection of interrelated components that make up the finished schema.

Figure 58 Collaboration Rules and Java Collaboration Rules Class
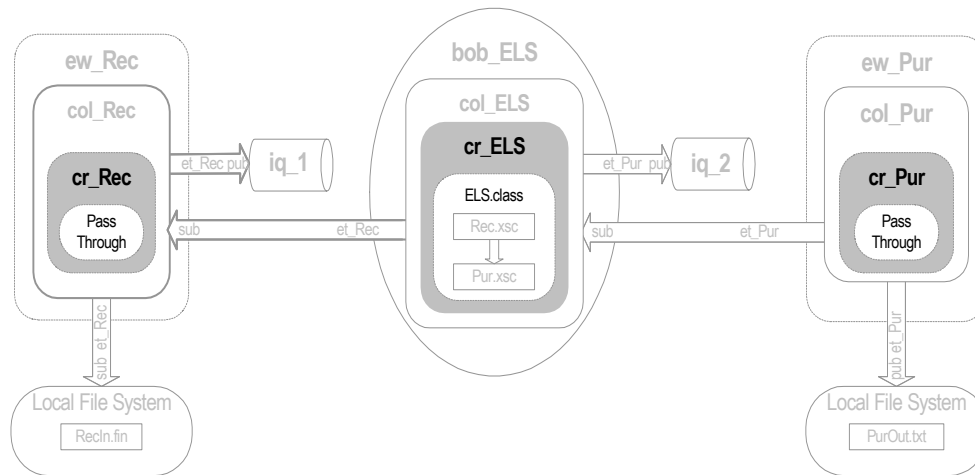
Pass Through Collaboration Rules are used to bring data into and take data away from the e*Gate environment. They do not change the data and therefore do not need the overhead of a sophisticated Collaboration environment such as Java or Monk.

Java Collaboration Rule: The procedure for creating a Collaboration Rule that uses the Java Collaboration Service is different from creating other e*Gate Collaboration Rules. See the *e*Gate Integrator User's Guide* for complete instructions on how to create this type of Collaboration Rule.

Use the following general steps to create the Java Collaboration Rules component for this scenario:

1 To create a Java Collaboration Rules component, you must also create a CRC. Use the Java Collaboration Rules Editor to create a new CRC with **Root_In (Rec)** as the source ETD and **Root_Out (Pur)** as the destination ETD.

2 You must now enter the logic used to combine the appropriate Events. You do this by adding the appropriate business rules (Java programming) using the Java Collaboration Rules Editor.

3 You must save and compile the Java source code. When the compiler is finished "Compile Completed" is displayed in the compile pane. The compile pane also

displays any errors generated by the compilation process in the Java Collaboration Rules Editor (see the following figure).

Figure 59   Java Collaboration Rules Editor—After Compiling
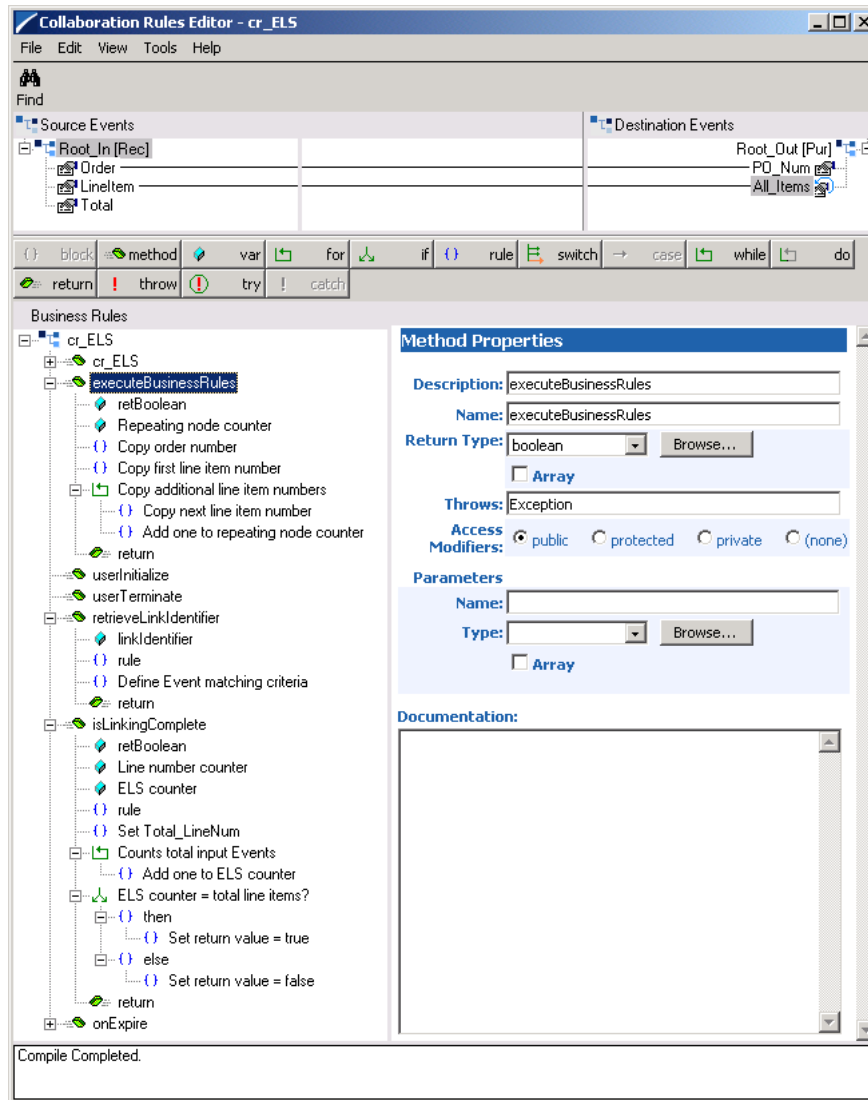
4   Finish this operation using the Enterprise Manager's Collaboration Rules Properties dialog box for **cr_ELS**. In the dialog box, **collaboration_rules\cr_ELS.class** is entered in the Collaboration Rules pane and **collaboration_rules\cr_ELS.ctl** is entered in the Initialization File pane.

Note: *For more information on how to create Collaboration Rules for ELS, see the **e*Gate Integrator Collaboration Rules Reference Guide**.*

Adding and Configuring e*Ways, BOBs, and IQs

Once you have created your ETDs and Collaborations, you are ready to add and configure the e*Gate components that use these parts.

The following figure highlights the components added in this step.

Figure 60   e*Ways, BOBs, and IQs
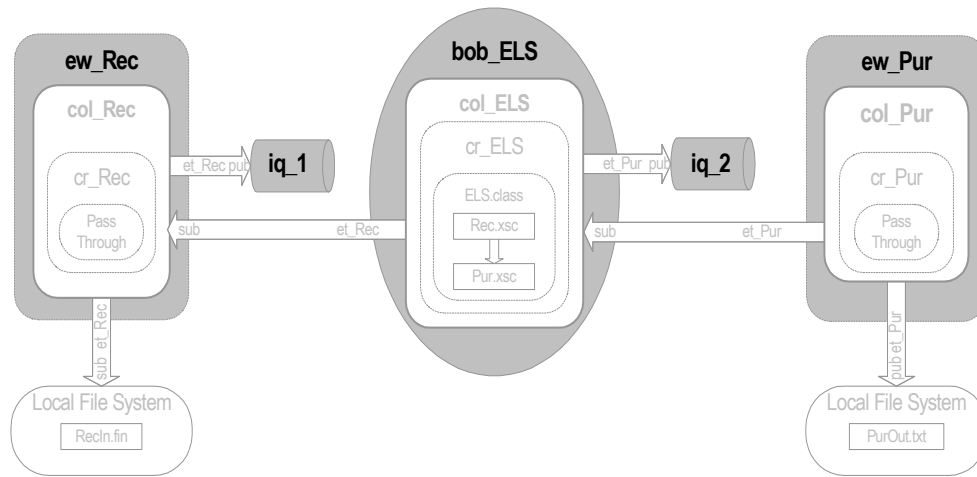
Create and configure the following file e*Ways:

- **ew_Rec file**
- **ew_Pur**

Create and configure the following BOB:

- **bob_ELS**

Create and configure the following IQs:

- **iq_1**
- **iq_2**

Add these IQs under the IQ Manager *hostname_**iqmgr.**

Note: *For details on how to create and configure these components, see the **e*Gate Integrator User's Guide**.*

Adding Collaborations That Route the Data

Collaborations are used by the e*Ways and BOBs to route the data through the e*Gate environment. Typically, the collaborations are configured in upstream-to-downstream order. Figure 61 shows the relationships of the Collaborations to the remainder of the parts that make up the complete schema.

Figure 61 Collaborations Showing Pub/Sub Relationships
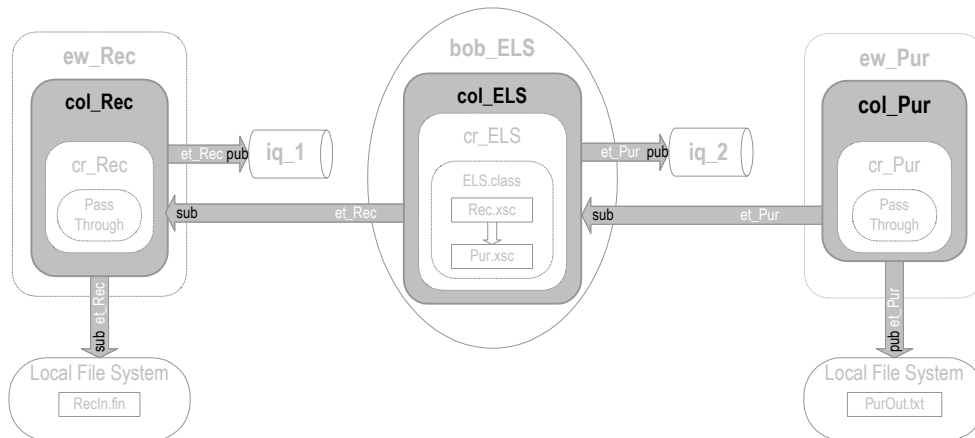
Create and configure the following Collaborations:

- **col_Rec** to bring the data into the e*Gate environment
- **col_ELS** to change the data from the **et_Rec** Event Type to the **et_Pur** Event Type
- **col_Pur** to send the transformed data out of the e*Gate environment

Note: *For details on how to create and configure Collaborations, see the **e*Gate Integrator User's Guide**.*

Designing Hardware Topology

The hardware topology is the distribution of components across hardware systems. In this scenario, the desired operations must correlate and synchronize the appropriate data. Keep in mind that this scenario can be just a small part of a much larger deployment.

The hardware distribution of your e*Gate components can dictate the need for ELS. For example, your system could have pre-sequenced data passing through components on two or more different hosts. Suppose you need to send all of that data to an external system in the same sequence (see **Figure 62 on page 160**).

Figure 62   Data in Multi-Host e*Gate Environment with ELS
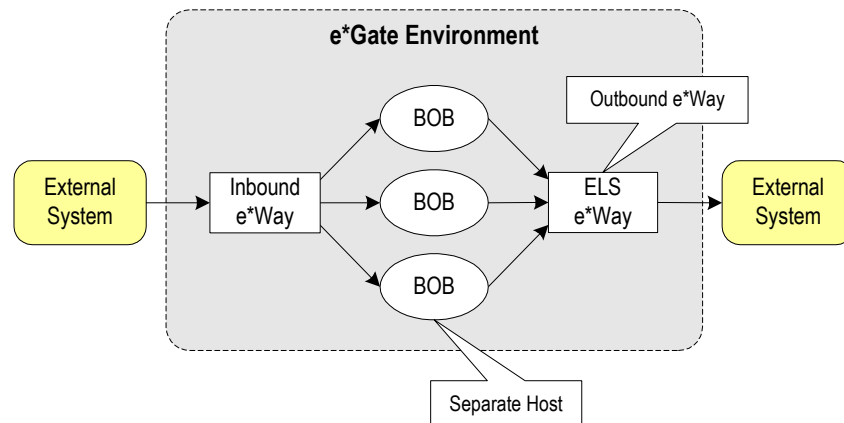
In the previous figure, BOB 3 resides on a separate host. You can configure the e*Way with an ELS Collaboration to regather the data and send it to external system B in a predetermined sequence. You could also create an additional BOB 4 (if desired) with an ELS Collaboration to gather the data in the same way. In this case, the outbound (Pass Through) e*Way could subscribe to BOB 4.

For best results, use hardware with maximum memory and speed capabilities (see **"System Requirements: Summary" on page 72**) to sustain the ELS data traffic.

Note: *You can mix ELS scenarios with non-ELS scenarios in a single deployment.*

Going Forward

After you have completed the total, detailed, design model of your environment and have finished creating that environment's architecture in e*Gate, e*Insight, and e*Xchange, you have successfully finished the design and development phase of your eBI Suite deployment. You are now ready to move into the final phases of your deployment project.

The rest of the chapters in this guide treat the eBI Suite deployment project under the following topics:

- **Testing and transition to production:** For a discussion of the testing, transition (go-live), and maintenance (fine-tuning) phases of your system deployment, see **Chapter 6**. These phases complete your deployment project.

- **Helpful tips: Chapter 7** "Frequently Asked Questions" gives you some helpful hints and tips for best practices.

Testing, Transition to Production, and Maintenance

This chapter explains the transition-to-production phase of eBI Suite deployment, including how to perform pre-transition testing, the transition operation, and post-transition maintenance procedures.
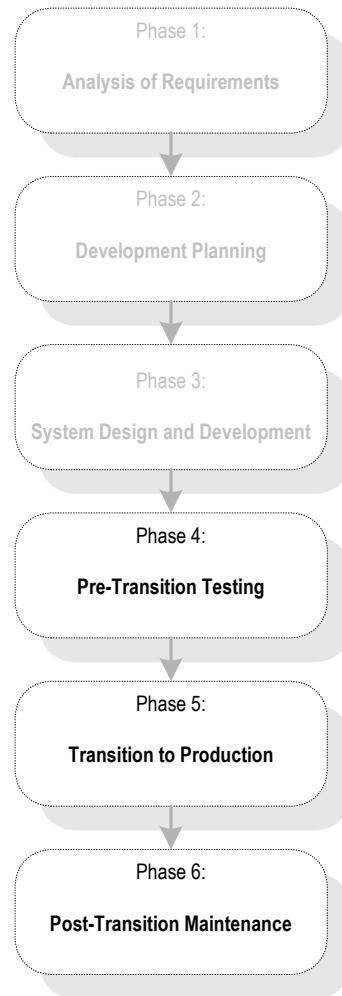
In This Chapter

6.1 Introduction: Transition to Production

After the analysis, planning, and design/development have been completed, there are three remaining deployment phases. **Figure 63 on page 162** shows a diagram of the six deployment phases. This chapter explains these remaining three phases:

- **Pre-transition testing:** It is crucial to the success of a project to fully test the system prior to transitioning from a lab to a production environment. This testing phase includes unit testing, system testing, and performance testing. This chapter explains the possible methods of testing an eBI Suite environment in the lab.

- **Transition to production:** After the system is fully tested, it must be transitioned, or migrated, from the lab to its ultimate production environment. This chapter covers the procedures and considerations for performing the transition to production. Another term for this phase is the "go-live" operation.

▪ **Post-transition maintenance:** Once the system has been migrated to its production environment, it must be monitored for correct performance, the need for changes, and possible errors. System monitoring is a critical step in the long-term success of the eBI Suite. Routine checks and fine-tuning help to establish long-term performance benchmarks and aid in identifying undesirable changes.

Figure 63   Testing, Transition, and Maintenance Phases

Phase 1:

Analysis of Requirements

Phase 2:

Development Planning

Phase 3:

System Design and Development

Phase 4:

Pre-Transition Testing

Phase 5:

Transition to Production

Phase 6:

Post-Transition Maintenance

Change Management

An important part of the entire deployment project is change management. In the Event that changes are required, they must be processed through the same cycle of planning, development and configuration, testing, transition to production, and maintenance monitoring as the rest of deployment. **Figure 64 on page 163** illustrates this cycle of change management.

Figure 64   Change Management Cycle
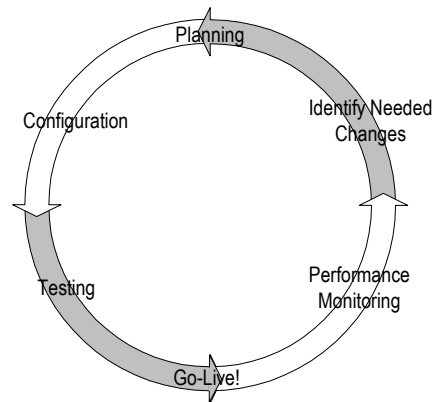
6.2   **Pre-Transition Testing**

An essential part of the implementation of any complicated system is thorough testing. You must do the following types of testing:

- **Unit testing,** testing of individual components and code in isolation

- **Integration testing,** testing of groups of components together, up to and including the entire system

- **Acceptance testing,** testing of a completed system (or portion thereof) to ensure that it meets the requirements established for it

For the most part, unit and integration testing are done in the development phase of the implementation, while acceptance testing is done as a final check before putting the system into production.

6.2.1 **Testing Methodology**

While how a system is tested varies, depending on the particulars of the specific system, certain methodologies apply to all system testing.

Parts to Whole

In general you must test the individual parts of the system before testing the entire system. Also along these lines, test individual components and blocks of code in isolation before testing them in a broader context.

6.2.2 **Test Plan**

Planning for system testing begins with a careful examination of the requirements of the system. A test plan is created in the analysis phase of the implementation. This test plan specifies how the system is tested and what requirements the system must meet before it is put into production (see **Chapter 3**). This test plan is further refined in the design phase of the implementation.

The functional and technical specifications outline the exact procedure used to conduct the tests, both at a component level and at a integrated system level. These specifications include:

- Type of data to use
- Expected output
- Who is responsible for the test

Type of Data To Use

The test plan specifies the type of data to use when testing they system. It is very important both at the component level and the integration level to work with data that is typical of the data that the system is designed to process. If possible, use real data from your pre-existing systems. Vary the data varied enough so that all possible types of processing implemented by the system are tested.

In addition to real-life typical data, use data designed to test the system's error handling. This data may have to be specially constructed.

Testing the Output

The test plan includes specifications for:

- Proper error handling
- Transaction processing speed
- Correct routing of information
- Correct transformation of data
- Any other special requirements

Responsibility for Testing

Who is responsible for a test depends on what type of test is done. In general, the responsibility for testing an individual component belongs to the developer who works on it. Whereas the responsibility for the testing of the entire system may fall to the project manager or the technical lead for the project. Acceptance testing is done by or in conjunction with people for whom the system is being created. Often this is the person or persons who are using the system when it is put in production.

6.2.3 Unit Testing

Unit testing checks the individual parts of a larger system for correct functioning prior to integration testing.

Each component and block of code used in the system must be unit-tested and its functionality verified before it can be used in the integrated system.

Unit testing is done as part of the development phase by the developer responsible for creating the component or code block in question. If the functional or technical specifications give a procedure for testing a particular component, this procedure must

be followed. If the specifications do not specify test procedures for a component, at a minimum the test must verify that the component or code block performs as outlined in the functional and technical specifications, testing individual parts of a system.

The tools used to test each part are different, depending on what the part is designed to do in the system. In addition, though all are designed to verify correct functioning, the methods employed vary, depending on the component being tested.
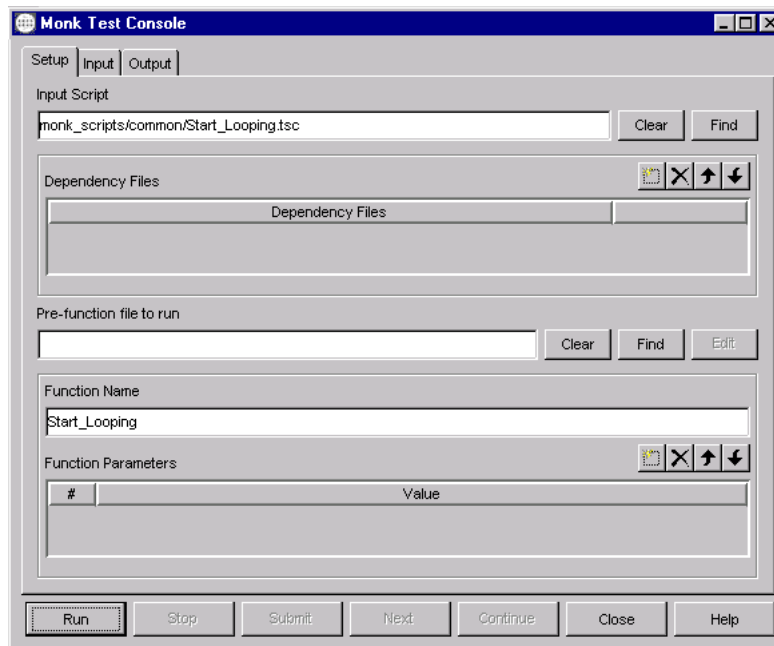
Monk Test Console

Use the Monk Test Console in the e*Gate Enterprise Manager to test each e*Way Intelligent Adapter or BOB component's operation for correct data transformation. The procedures that follow explain how to test the sample **Start_Looping.tsc** Monk Collaboration Rules (data transformation) script using the Monk Test Console.

To test a Collaboration Rule script using the Monk Test Console

1 From the Enterprise Manager, open the Monk Test Console window (see **Figure 65 on page 165**).

2 Enter the name of the transformation you want to test in the **Input Script** text box (see the following figure).

Figure 65   Monk Test Console—Setup Tab

3 Select **Input Data Files** and use the file selection controls to specify a text file to use as input, or select **User Data** and enter the input directly in the text box (see the following figure).

Figure 66 Monk Test Console—Input Tab
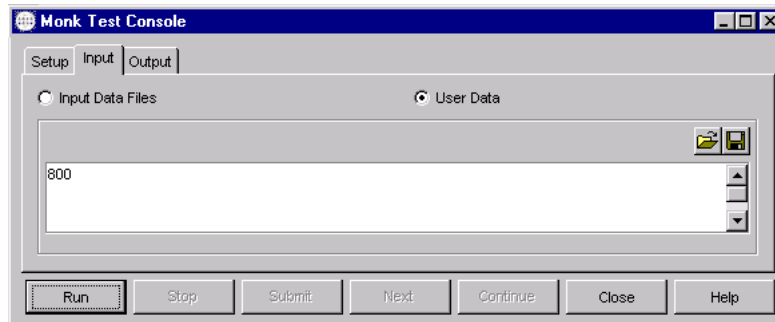
4 Select the **Output** check box then click **Run** to test the transformation.

The results of the transformation are displayed in the top text box (see the following figure).

Figure 67 Monk Test Console—Output Tab
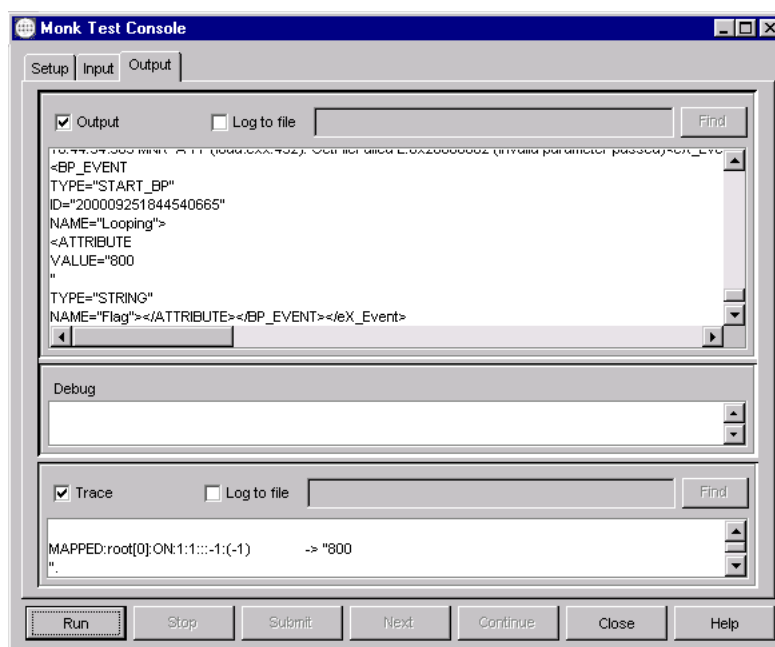
See the *e*Gate Integrator User's Guide* for more information on how to use the Monk Test Console.

Using stctrans

Monk functions can also be run from the command line using the **stctrans** command. This program functions in the same way as the Monk Test Console, without the GUI. See the *e*Gate Integrator System Administration and Operations Guide* for information on its usage.

Java Code Testing

The Java Collaboration Rules Editor automatically provides messages telling you the errors in any code, when you compile. You can also use the Java Debugger feature available in the e*Gate Monitor GUI to debug Java Collaborations. For details on how to use these features, see the *e*Gate Integrator User's Guide*.

Testing e*Way Configuration Files

Once the transformations used by an e*Way have been tested, you must run the e*Way to test whether it has been properly configured to communicate with the external system to which it connects. A BOB, because it does not connect to an external system, does not have a configuration (**.cfg**) file.

Note: *If the system is small and uncomplicated, you may wish to skip this test and move directly to integration testing, because of the additional time it takes to set up this test for each component.*

Testing an e*Way's configuration requires the use of an additional component, a file e*Way (**stcewfile.exe**), that is used to pass data to or take data away from the e*Way being tested.

For example, Figure 68 shows the components in a simple end-to-end scenario.

Figure 68 SAP-to-Accounting Bridge
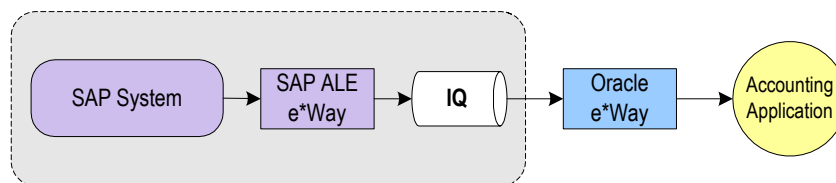
In this example the SAP ALE e*Way receives a purchase order in IDoc format from the SAP system. It converts the data to an X12 850 purchase order and publishes it to the Intelligent Queue (IQ). The Oracle e*Way retrieves the X12 data from the IQ and inserts it into the accounting application's database.

To test just the SAP ALE e*Way, we replace the Oracle e*Way with the simple file e*Way that writes a text file containing the X12 data published by the SAP ALE e*Way. You can then visually inspect the text file to see if it contains the correct data (see Figure 69).

Figure 69 SAP ALE e*Way Test Setup
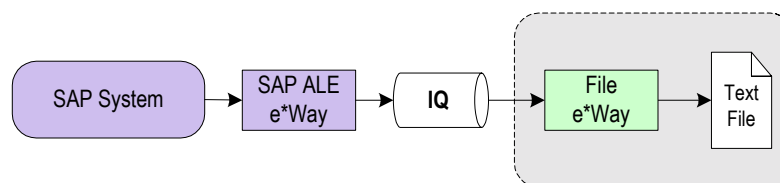
The advantage of this type of test is that, in addition to testing the transformation, the e*Way's configuration file that contains the information the e*Way uses to

communicate to the external system is also tested. In addition, this test also tests the data routing up to the point that the data is stored in the IQ.

To test an inbound e*Way

1   Set up the external system to which the e*Way connects.

2   Create or edit the e*Way's configuration file.

3   Add an outbound file e*Way to the schema containing the e*Way you wish to test. This file e*Way subscribes to the Event published by e*Way you are testing and writes the Event information to a text file without doing any transformation (Pass-through Service).

4   Set the file e*Way, the e*Way you are testing, and the IQ to automatically start when the Control Broker starts. The other components in the schema must be set *not* to start automatically.

5   Start the schema.

6   If possible, verify the connectivity between the external system and the inbound e*Way you are testing, before sending in data. There are various ways to do this depending on the particular e*Way. For example, some e*Ways write an entry into their log file if the connection to the external system is successful. In other cases you can use the external system to verify connectivity.

7   Use the external system to send data to the e*Way.

8   Verify that a text file with the correctly processed data is created by the file e*Way.

To test an outbound e*Way reverse the direction the data flows in the previous procedure.

To test an outbound e*Way

1   Set up the external system to which the e*Way connects.

2   Create or edit the e*Way's configuration file.

3   Create a test file that duplicates the information contained in an Event that the e*Way would normally pick up from an IQ.

4   Add an inbound file e*Way to the schema. This file e*Way picks up the test file and publishes it to the IQ without any transformation (Pass-through Service).

5   Make sure that only the components used in the test (the file e*Way, the e*Way being tested, and the IQ) are set to start automatically when the Control Broker is started.

6   Start the schema.

7   If possible, verify connectivity before sending data to the external system.

8   Make the input file available to the file e*Way. Do this by copying the file to the polling location or changing the name of a properly located file to a name to be picked up by the file e*Way.

9   Verify that the data is correctly translated and transferred to the external system.

6.2.4 **Integration Testing**

Integration testing verifies how well the new components work together and with the existing eBI Suite's infrastructure.

If the system is large and complex, you can break the integration testing into pieces designed to test a self-contained portion of the system.

Partial Integration Testing

Partial integration testing verifies correct data movement from one external system to another, that is, a complete data path. For example, you have a system that brings in data from a single external system then sends it to several other external systems. A partial test of this system would be to test whether the data can be sent to one of the external systems.

Complete System Testing

Complete system testing tests the entire system including the interaction with all the external systems. If the system is large and complex, this type of test requires a great deal of coordination. Set up this test to duplicate the actual production system. In fact, in many cases it is used as a dry run prior to doing the actual acceptance test.

Performance Testing

Closely related to integration testing is performance testing. Integration testing tests whether the system works, performance testing tests whether the system works fast enough. This type of testing must be done once a component or system is functioning correctly and transforming the data properly. It is important that you do this type of testing in the context of integration testing, because many factors in combination affect performance. Just speeding up one component may not speed up the performance of the entire system.

The exact requirement or goal in terms of the system's performance must be specified in the test plan. Whether you meet the goal determines whether you pass the test. An additional goal in performance testing is to find the slow spots in the system. Uncovering these slow spots allows additional system resources to be allocated intelligently in order to improve processing.

Speed Testing

This operation tests whether the system processes data fast enough. Make sure that the logging is set to normal levels before doing a speed test, because higher-than-normal levels of logging can seriously degrade system performance and slow processing speed.

Stress Testing

This operation tests whether the system can handle the expected load. Similar to speed testing, this type of testing attempts to overload the system with data to see whether or how it could fail. Many times, network bottlenecks can be uncovered this way. The test plan describes the methodology to employ for this type of test.

6.2.5 **Acceptance Testing**

This test is done before moving the system into production and is used as a final check to prove to its end-users that it performs according to plan.

Acceptance testing can be for a partial system or for a complete system. If the entire system is not being put into production at the same time, acceptance testing can be done on the portion of the system going into production.

The test plan specifies all conditions the system has to meet to be acceptable to put into production (see **Table 4 on page 57**). In addition, the test plan specifies the person or persons who must approve the system and must be involved in the test.

6.2.6 **Troubleshooting**

You have the following powerful tools at your disposal for finding and correcting errors in your individual components and in your integrated eBI Suite:

- e*Gate Monitor
- Log files created by individual components
- e*Gate Integrator Alert Agent
- e*Gate Integrator SNMP Agent

e*Gate Monitor GUI

The e*Gate Monitor GUI gives you control over the components in an e*Gate schema. The e*Gate Monitor alerts you to the status of these components, for example whether they are running, and it allows you to send commands to them, such as to start or shut down. See the *e*Gate Integrator Alert and Log File Reference Guide* for information on how to use the e*Gate Monitor's features.

When you use the e*Gate Monitor to troubleshoot running components, you can see at a glance whether any components are working. If a component has been set to auto-start and remains down after the Control Broker has been started, it is likely there is a configuration issue that is prEventing the component from starting properly. You can verify that this is the case by sending a "start" command to the component. If the component starts briefly then goes down, suspect a schema configuration problem.

For more information on using the e*Gate Monitor GUI, see **"Using the e*Gate Monitor" on page 180**.

Using Log Files

To gain further information you must use the log files created by the component that has failed. When a component or system is not working errors are written to the log file to help you diagnose the problem. In a normally functioning system, only the most serious errors are written to the log, such as a shut-down component, because each entry written to the log uses up system resources and slows processing. To learn more about a malfunctioning component, you can enable the system to log more information.

See the *e*Gate Integrator Alert and Log File Reference Guide* for information on troubleshooting e*Gate and how to use log files. This document also contains information on how to activate the component-logging and debug features.

In general, there are two categories of errors that you can find in the log files. Configuration errors (including communication errors) and transformation errors.

Configuration Errors

These errors can be traced to a faulty system setup or a bad e*Way configuration file. For example, an "unable to load module work slice" entry in the log file indicates that there is a problem with the publication-and-subscription information for that component. The figure below shows a sample log file with a configuration error.

Figure 70   Sample Error Log
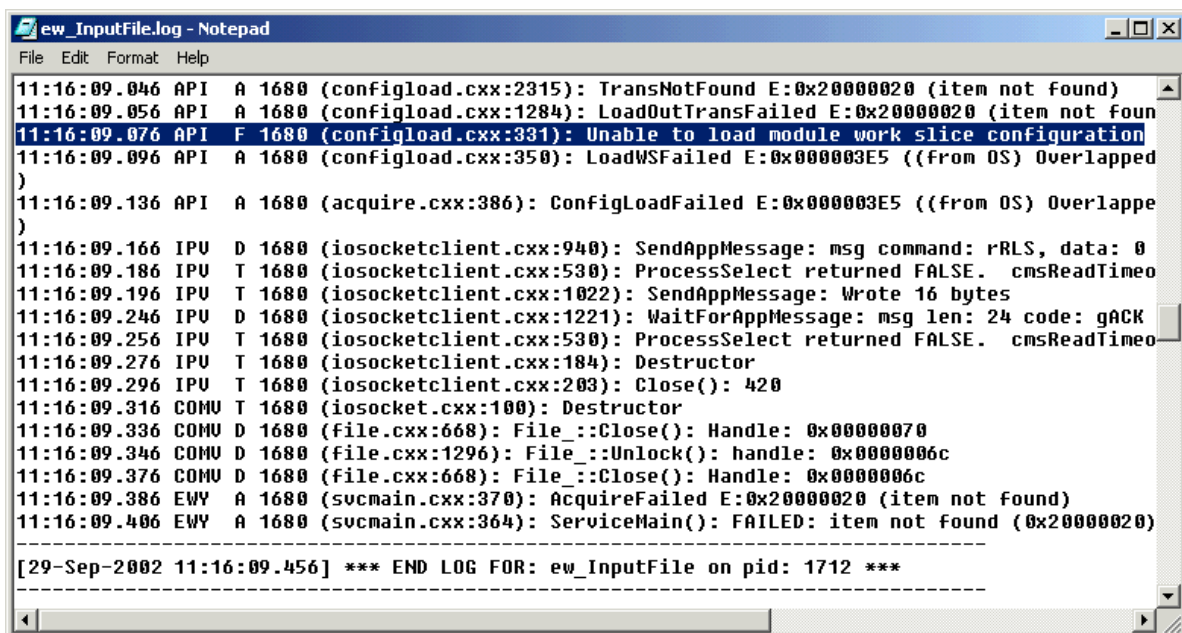
The "unable to load module work slice" error can also indicate that the configuration file has not been committed to the Registry. See the *e*Gate Integrator User's Guide* and *e*Gate Integrator System Administration and Operations Guide* for information on committing files to the Registry.

Transformation Errors

These are errors with the transformation associated with the component. These errors are documented in the component's log. The quantity and detail of the information in the log depends on the options selected. For transformation errors, having the Monk (MNK) and Monk verbose (MNKV) channels selected results in the most information.

You can use this information to make changes to your transformation then retry the scenario.

6.3   Transition to Production

After fully verifying the performance and reliability of the system, the next step is to transition it to the live production environment, the transition-to-production phase, also call the go-live phase. This process can be affected by various factors, such as the proximity of the lab to the production location and the amount of equipment and information to move.

Items to Migrate

The items that have to be moved from the lab to the production environment are:

- e*Insight business process files
- e*Xchange trading partner profile files
- e*Gate Registry files
- Third-party data sources files
- Related hardware (equipment)

Migration Tools

Use the following tools to assist you in your migration:

- **e*Insight** export/import features
- **e*Xchange** Repository Manager features
- **e*Gate Registry utility** (**stcregutil.exe**) or the e*Gate Enterprise Manager GUI export/import features

*Note: For more information about e*Insight and e*Xchange, refer to the **e*Insight Business Process Manager User's Guide** and the **e*Xchange Partner Manager User's Guide**.*

General Transition Steps

When transitioning your system from the lab to a production environment, follow these general steps:

1 Export the business processes, trading partner profiles, and schemas

2 Move the exported files from the lab to the production environment

3 Import the business processes, trading partner profiles, and schemas in the production hosts

These steps are explained in greater detail in the rest of this section.

6.3.1   Role of e*Insight

When e*Insight is combined with e*Gate, once you have modeled your business flow, you can use the e*Insight GUI to monitor the execution of the components of the model on a real-time basis.

Exporting Business Processes

e*Insight export/import features make it simple to move a business process from one host to another. Use these features to migrate the business processes from your lab to the production environment. See the *e*Insight Business Process Manager User's Guide* for details.

Note: *e*Xchange also allows you to export and import, using its Repository Manager feature (see **"Export/Import Using e*Xchange" on page 178**). For details on how to use the e*Xchange Repository Manager, see the **e*Xchange Partner Manager User's Guide**.*

Integrated Monitoring

The monitoring component of e*Insight provides a clear, step-by-step, color-coded graphical representation of each Business Process Instance (BPI). This feature allows you to identify processes that need intervention, repair, or authorization.

For example, e*Insight allows you to do the following operations:

- Analyze the contents of an operation that failed to be processed
- Modify (repair) that operation
- Restart the failed activity within a BPI, taking into account the modified operation

Figure 71 shows an example of the e*Insight Main window in the Monitoring mode of operation.

Figure 71   e*Insight Main Window (Monitoring Mode)
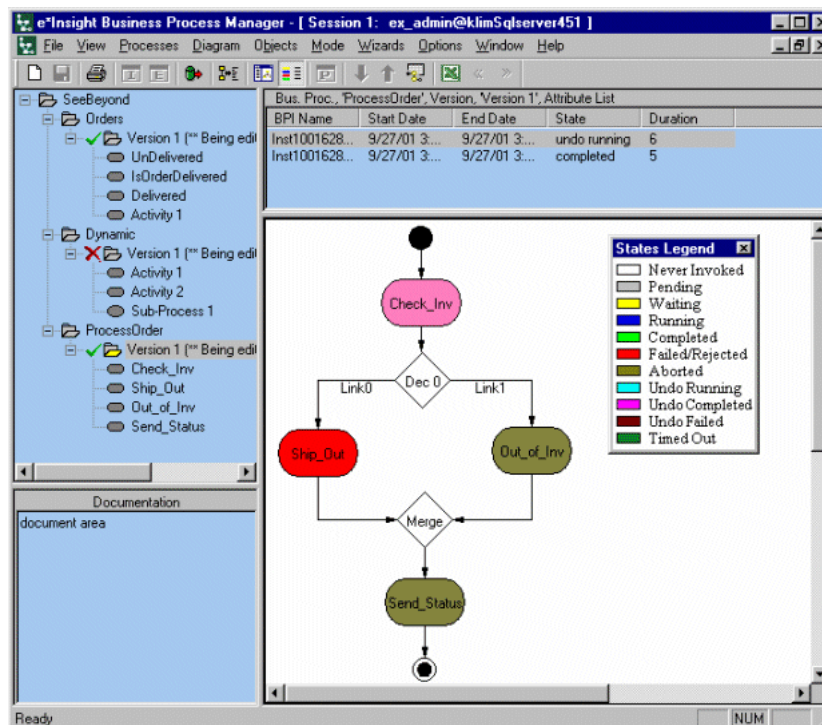
For more information on monitoring your system using e*Insight, see **"e*Insight Monitoring Mode" on page 181**.

6.3.2 Export Operations

This section explains the process of exporting:

- Business processes, using e*Insight
- Trading partner profiles
- Schemas

It is not necessary to perform these exports in the order shown in the previous list. These elements can be exported in any convenient sequence.

To export a business process version from e*Insight

1 In the configuration mode of e*Insight Tree view, select the business process version you want to export.

Note: *Take care when migrating a business process from a lab environment not to export the instance data. Otherwise, the lab's test data can be imported into your production host.*

2 Right-click the business process version and click the **Export Business Process Version** command on the pop-up menu. The following figure shows an example of exporting a business process version.

Figure 72 Export Business Process Version
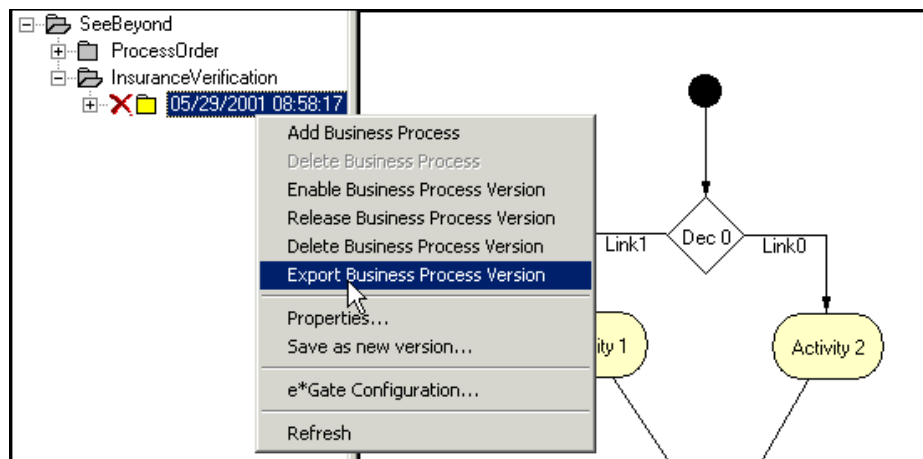
A dialog box appears (see **Figure 73 on page 175**), allowing you to export the business process instance data, as well as the business process definition.

To export the business process definition only, ensure that the **Export Business Process Instances** and **Archive** check boxes are not checked. The Export Business Process dialog box also allows you to define the file name and path for the export file.

Figure 73   Export Business Process Dialog Box
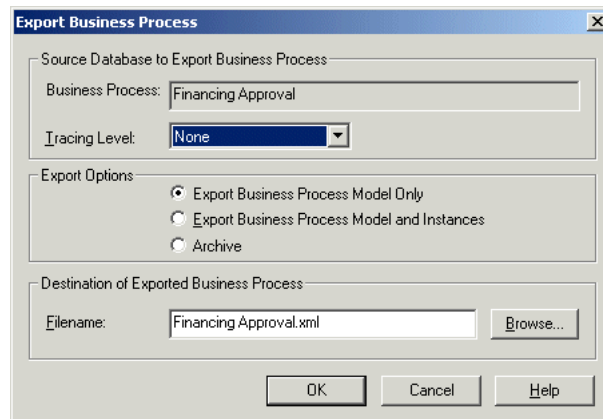
3 A command prompt window appears while export files are being created. When the export is complete, a message box appears showing the name of the export file (see the following figure).

Figure 74   Completed Export File
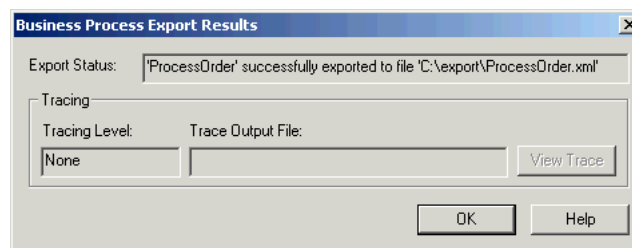
Exporting Trading Partner Profiles

To export the e*Xchange trading partner profiles, use the e*Xchange Repository Manager export feature. Create an **.exp** file to import into the production host.

Exporting e*Gate Schemas

To export an e*Gate schema, type the following at the command prompt:

```
stcregutil.exe -rh registry -rs schema-name -un user-name
    -up password -rt -usr -e file-name
```

Where:

registry is the name of the Registry Host

schema-name is the name of the schema you wish to export

user-name and *password* are valid a e*Gate username/password combination, and

file-name is the name of the ASCII text file to which the schema data is written.

The **-rt** and **-usr** parameters are used to export the resource table as well as the user names. For more information on using **stcregutil.exe**, see the *e*Gate Integrator System Administration and Operations Guide*.

Important: *This method for moving an e*Gate schema assumes that at least one of the following facts is true:*

- ◆ *The default schema for the source and target Registry Hosts are identical.*

- ◆ *The schema that is being moved relies upon no files that are stored in the default schema. In other words,* all *the files that the schema requires are stored in the schema directory rather than in the* **\default** *directory.*

A schema can also be exported from within the e*Gate Enterprise Manager using the Export Schema Definitions feature. For more information on the schema export feature, see the *e*Gate Integrator System Administration and Operations Guide*.

6.3.3 Moving Files

Depending on the size and amount of the files created during the export, they can be moved from the lab via floppy disk, tape, CD, or LAN connection.

Note: *The archiving tool you use must be able to archive and restore files that have long file names, that is, names longer than "8.3" restrictions allow. Archiving tools that store files in 8.3 format may not restore the schema files properly to the destination Registry Host.*

The files to be moved are:

- **e*Insight:** The **.xml** file created by the business process version export. By default, the file name is the same as the business process name. For example, when exporting a business process called "WesternAlliance," a file called **WesternAlliance.xml** is created, unless you defined a different name.

- **e*Xchange:** The **.exp** file created by the e*Xchange Repository Manager export feature.

- **e*Gate:** The **.zip** file created by the e*Gate full schema export and containing the schema definitions (.**exp** file) and associated Registry (and Sandbox) files. For example, when exporting a schema, you need the **.zip** file created by the schema export.

Note: *Where necessary, take care to restore any files to the appropriate locations on the production host computer.*

See the *e*Insight Business Process Manager User's Guide, e*Xchange Partner Manager User's Guide,* and *e*Gate Integrator System Administration and Operations Guide* for more information file export and import.

6.3.4 Import Operations

Before importing the exported files to the production host computers, you must first install all the application software. Install the e*Gate Registry Host, Participating Host, add-ons, e*Insight, and e*Xchange on the production hosts exactly the way they were installed on the lab computers.

Similarly, any third-party software (such as Oracle and the OBDC drivers) must be installed as well. You must install all these items before any files from the lab hosts are imported.

Importing Business Processes

To import a business process version to e*Insight

1 In the Configuration mode, on the File menu, click the **Import** command.

2 The Import Business Process dialog box appears (see the following figure).

Figure 75   Import Business Process Dialog Box
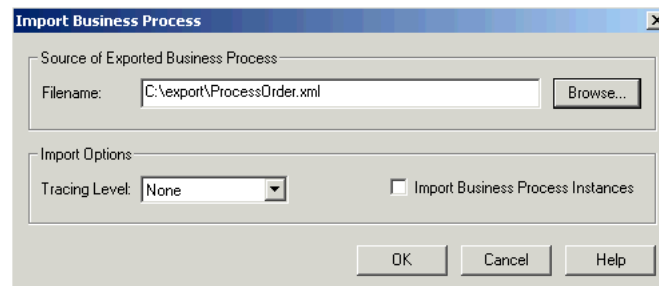
3 Click **Browse**.

   The Open dialog box displays.

4 Locate the file you want to import and click **OK**.

5 Verify that the **Import business process instances** check box is unselected (in case the instances were exported from the lab by accident).

6 Click **OK**.

   The business process is added to the Tree view of the e*Insight GUI.

Importing Trading Partner Profiles

To import e*Xchange trading partner profiles, use the e*Xchange Repository Manager import feature, and import the .**exp** file that was copied from the lab host. This is described in **"Export/Import Using e*Xchange" on page 178**.

Importing e*Gate Schemas

To import the e*Gate schema and all associated files

1 Log on to the e*Gate Enterprise Manager.

Note: *Using the Import Wizard GUI is not only the easiest way to import a schema, but it also allows you to change the host, Control Broker, or IQ Manager name, as well as change the port numbers.*

2 Create a new, empty schema then give it the desired name.

3 Select the **Components** tab.

4 On the File menu, click the **Import Definitions from File** command.

The Import Wizard Welcome page appears (see Figure 76).

Figure 76 Import Wizard Welcome page

5 Follow the prompts and complete the procedures the wizards tell you to do.

6 Verify that you restored all files from the lab into the newly created schema directory. Verify that the original directory structure from the lab environment has been maintained.

For details on how to use this import feature, see the *e*Gate Integrator System Administration and Operations Guide*.

6.3.5 Export/Import Using e*Xchange

You can use the Java-enabled e*Xchange Repository Manger feature to conveniently handle importing, exporting, archiving, and de-archiving in this application. For example, you can use it to export/import trading partner profiles. **Figure 77 on page 179** shows an example of this GUI feature.

Note: *See the **e*Xchange Partner Manager User's Guide** for details on how to use the Repository Manager.*

Figure 77   e*Xchange Repository Manager Window
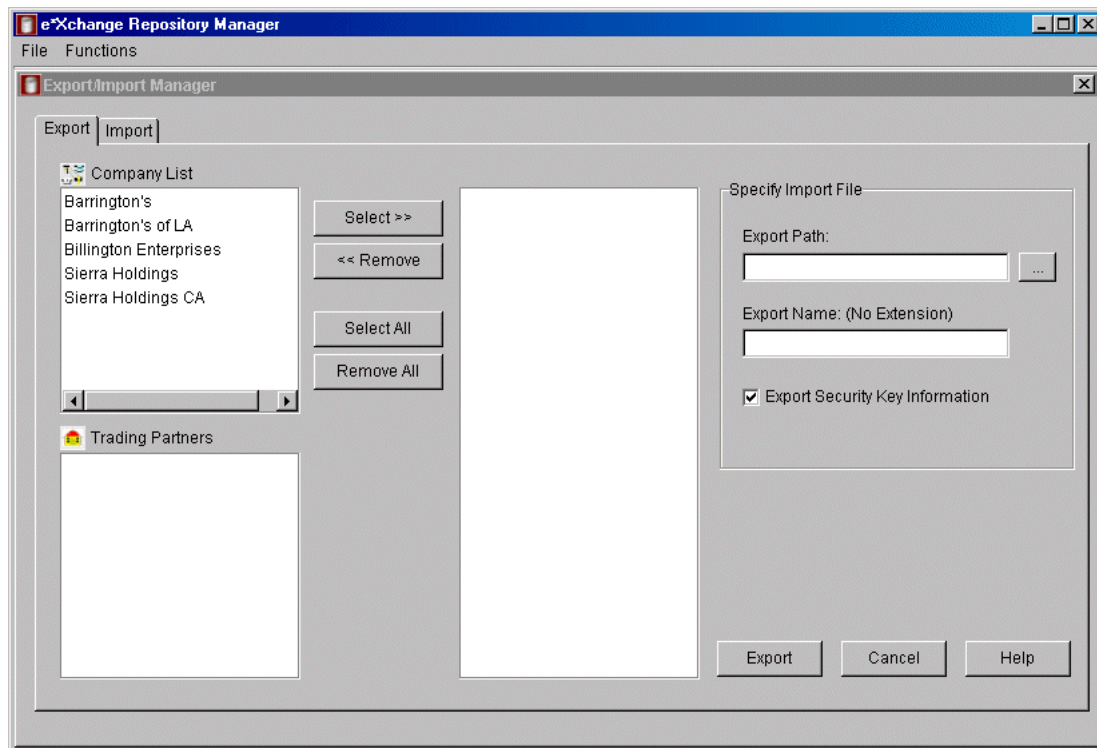
6.3.6 Running the Schema

Now the production hosts is ready to run. e*Insight, e*Xchange, and e*Gate have all been installed. Any business processes, trading partner profiles, and schemas have all been migrated from the lab hosts to the production environment. Now it is time to activate the production system.

Switch the Control Broker On

It is common to start an e*Gate schema's Control Broker from the command line using **stccb.exe** in a lab environment. However, in the production environment, start the Control Broker as a service (in Windows NT and Windows 2000). For detailed instructions on starting the Control Broker as a service, see the *e*Gate Integrator System Administration and Operations Guide*.

e*Insight/e*Xchange Oracle Database

Before running any schemas that use e*Insight and e*Xchange, ensure that your Oracle database is up and running. With the Oracle database running, you can start the e*Gate Control Broker. This action runs the schema that activates e*Insight and e*Xchange. See the appropriate e*Insight and e*Xchange implementation guide for details.

6.4   Post-Transition Maintenance

With your eBI Suite up and running, it is important to monitor the system's performance, check for errors, and make any needed changes. **Figure 64 on page 163** shows the overall steps to take when changes are required. Such changes must be put through the same high scrutiny as the original system design — from analysis to testing to the eventual transition to production.

6.4.1   Monitoring System Activity

Monitoring system performance is a critical step in the long-term success of the system. Routine checks help to establish long-term performance benchmarks. Such benchmarks are helpful in identifying undesirable changes. If you don't have a good feel for a healthy system's vital signs, you may have a harder time recognizing when your system could need some attention.

Using the e*Gate Monitor

You can use the e*Gate Monitor GUI feature to view the status of the components in an e*Gate schema. The **Figure 78 on page 180** shows a schema and all of its running components. Components that are running are displayed in their normal colors, but components that are stopped appear in red.

Figure 78   e*Gate Monitor Main Window
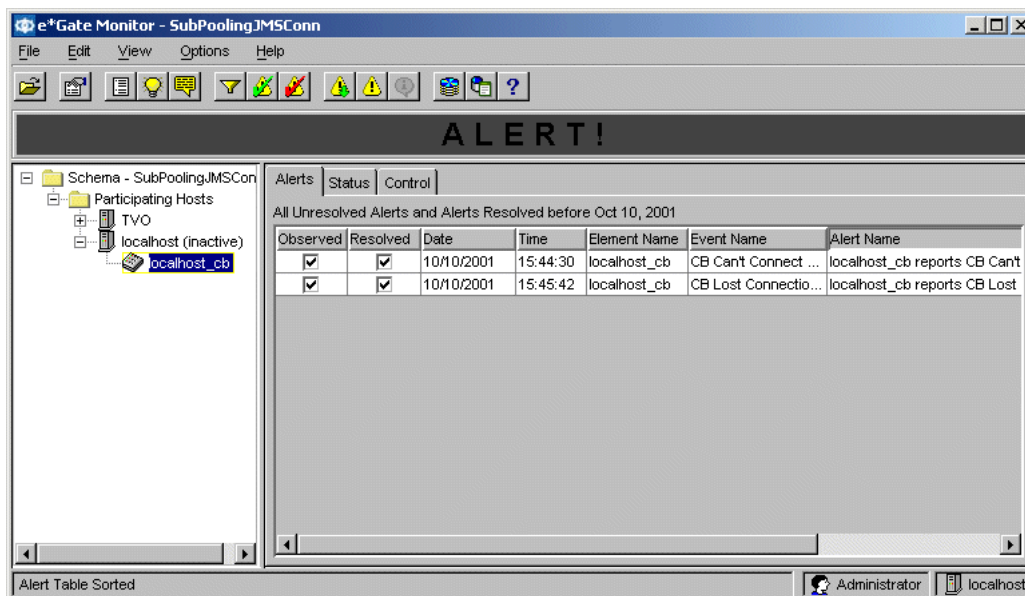
The e*Gate Monitor can also be used to start and stop e*Gate components. Any Alerts can be viewed and marked as resolved after the appropriate corrective action has been taken.

Note: *See the following guides for more information on how to use the e*Gate Monitor feature:*
e*Gate User's Guide for basic information, for examples, menu and toolbar operation
e*Gate Integrator Alert and Log File Reference Guide for how to interpret Alerts and troubleshoot e*Gate

e*Insight Monitoring Mode

The e*Insight Monitoring mode displays the business process instances. The instances can be viewed in the diagram, or list panes. The List pane (see **Figure 79 on page 181**) displays a list of all business process instances. A List wizard makes it easy to configure the view and layout in the List pane.

Figure 79 e*Insight Monitor—List View
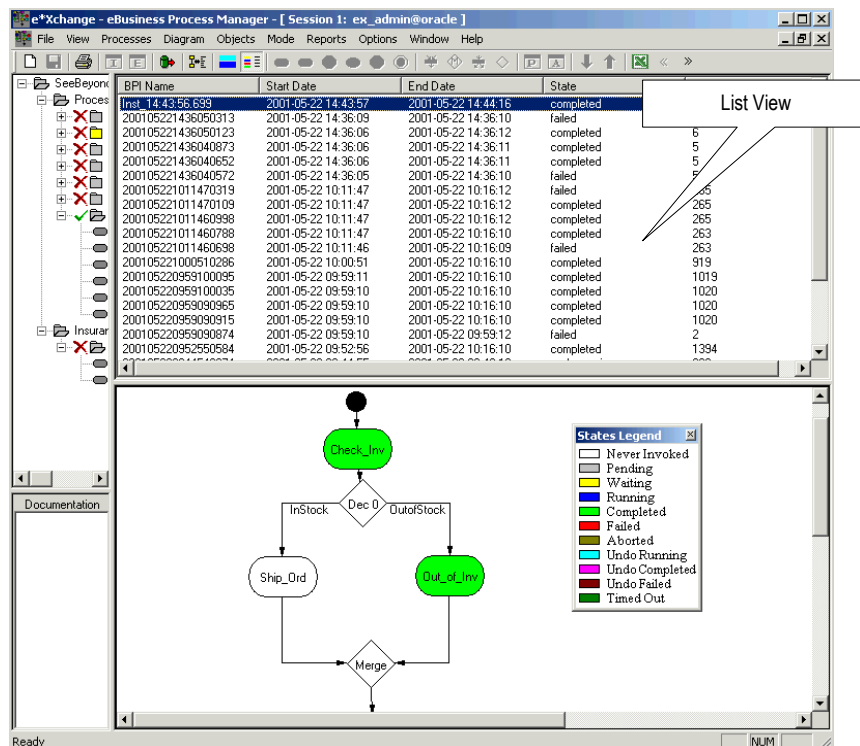
To view a graphic representation of a business process instance

1 Select a specific business process instance in the List pane of the e*Insight Monitor (see Figure 79).

A diagram of the business process instance appears (see Figure 80). The completed business process Activities are displayed in green. Failed Activities are displayed in red.

Figure 80   e*Insight Monitor—Diagram View
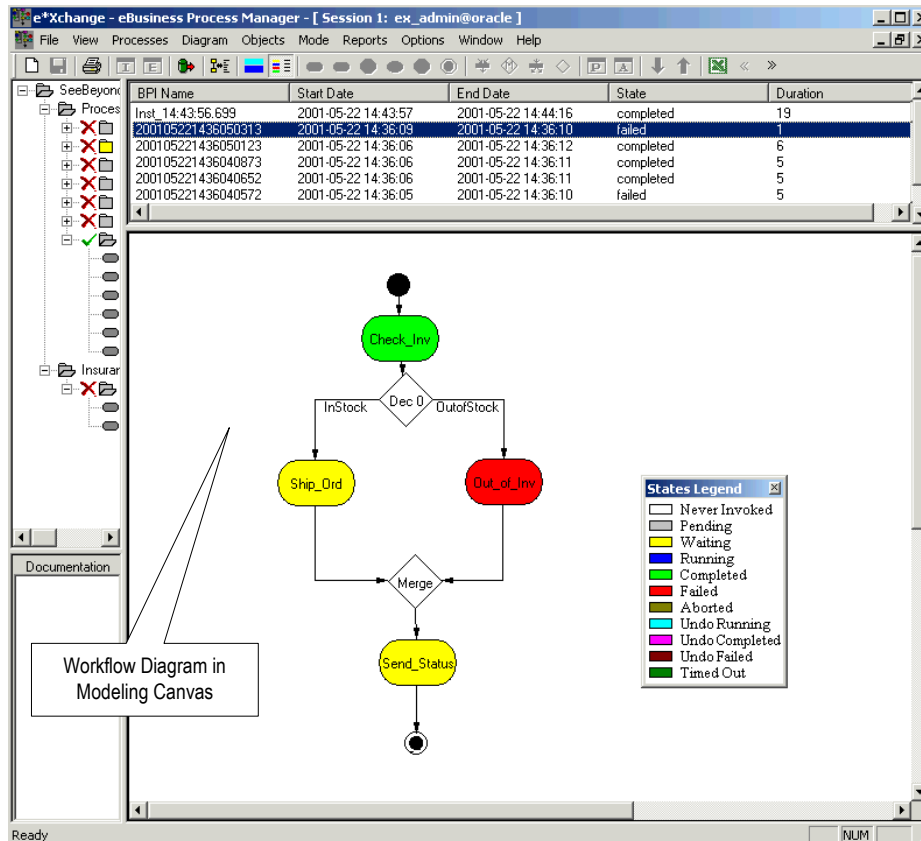
You can configure business processes to process failed instances manually or automatically as follows:

- **Automatic:** Failed instances in business processes configured as automatic are rolled back through a series of "undo" Events. The undo logic is performed for the failed activity and each of the upstream activities until the effects of the entire business process instance have been reversed (undone).

- **Manual:** Failed instances in business processes configured as manual are suspended. They then wait for user interaction. The properties for the failed instances can be viewed and the values edited (see the next figure). This feature

makes it possible to manually repair the failed instance so that it continues to be processed by e*Insight.

Figure 81   Business Process Properties
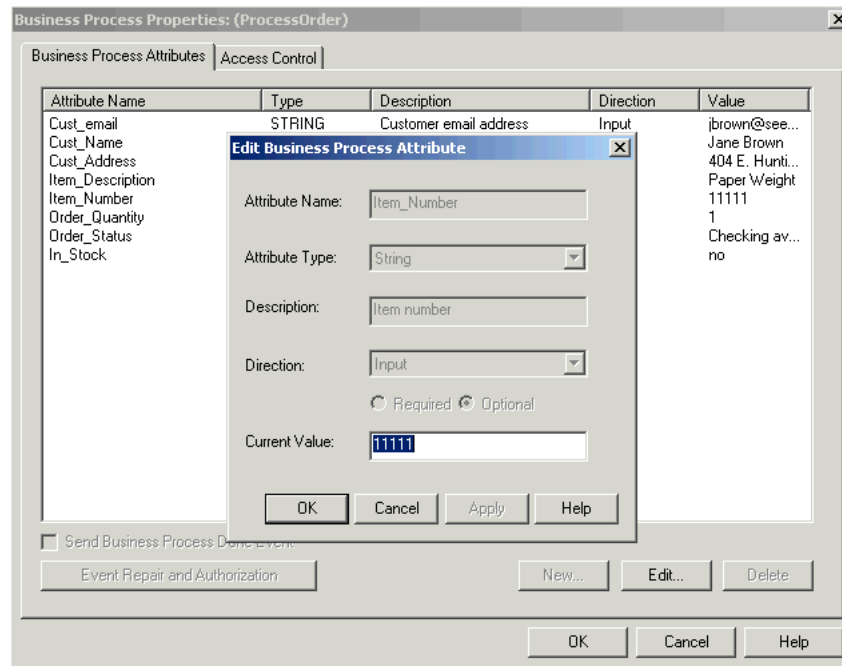
e*Insight Analysis and Reporting Features

e*Insight provides GUI wizards for analyzing historical business processes to enable process optimization over time. This tracking and measurement capability of business process performance is also useful for demonstrating returns on investments for eBusiness initiatives.

You can create a complete historical picture by tracking and storing all process instances and their associated attributes across all the integrated systems, often providing a reporting and analysis view that was previously unavailable to the enterprise because of the lack of integrated data.

e*Insight's reporting features allow you to view and track data values across business-process instances. You can also use these features to analyze historical execution trends and identify possible bottlenecks in the business process.

Report Wizard: e*Insight provides a Report Wizard that guides you through the report creation process to create both Chart and Tabular views of your eBusiness process information. You can also readily export information directly into a Microsoft Excel or other spreadsheet to utilize analysis features such as pivot tables.

Using e*Insight in the e*Gate Environment

The e*Insight components run within the e*Gate environment and are made up of the following major elements:

- **e*Gate Schemas:** Logical grouping of integration components that implement the configuration of your business processes.

- **e*Insight Database:** Oracle, SQL Server, or Sybase database of e*Insight's configuration, including the business-process templates and their run-time instances; this database also serves as a data warehouse for the business-process tracking and analysis.

- **e*Insight Engine:** Implements and manages the information-process flows and business rules; all business-process communications flow through the e*Insight Engine for monitoring, management, and analysis.

- **e*Ways:** Used to implement business-process Activities that perform actions requiring communications with external systems; e*Insight automatically generates the e*Ways needed to implement your business process.

For more information on implementing and using e*Insight with e*Gate, see the *e*Insight Business Process Manager Implementation Guide*.

6.4.2 e*Xchange Message Tracking

The Message Tracking features in e*Xchange are used to drill down to specific e*Xchange messages. These features are especially helpful in finding and resolving failed messages.

Using Message Tracking

e*Xchange can help you track the messages for your company and a trading partner, using the Microsoft Internet Explorer. Message Tracking helps you trace message trails and track down possible errors.

To use this feature, first select a trading partner profile (see Figure 82).

Figure 82 e*Xchange Trading Partner Profile Selection

Next select the desired message profile (see Figure 83).

Figure 83   e*Xchange Message Profile Selection

In the window shown previously, you can see that, for this company and the current trading partner, the following messages are tracked:

- Inbound Acknowledgments
- Inbound Orders

Order/Response Message Example: The rest of this section uses Orders and Order Responses as examples.

Using the Message Details window, you can see that if the company receives an order message, a response is required, which later becomes an Outbound message. Figure 84 shows this window.

Figure 84   e*Xchange Message Details Window

For example, if you need to learn more about an inbound message, you can access the following information:
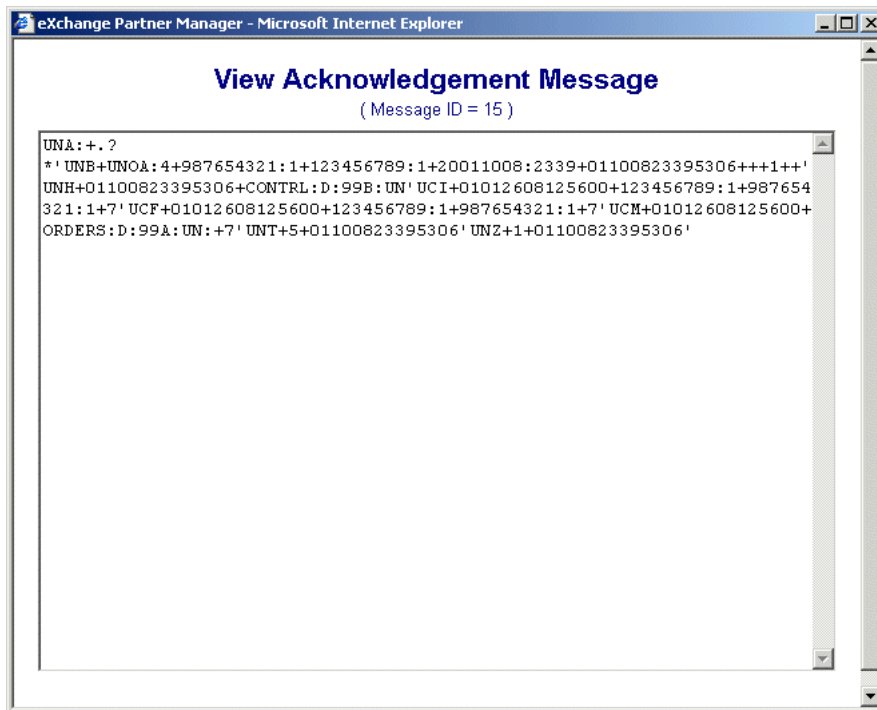
- Original message received by the company (see Figure 85) in the View Original Message window

▪ Acknowledgement generated (see **Figure 86 on page 186**) in the View Acknowledgement Message window

Figure 85 e*Xchange View Original Message Window, Inbound

Figure 86 e*Xchange Acknowledgement Message Window

The information shown in the previous figures makes it easy to see all the information relating to a given message.

If you want to look at any outbound messages for example, use the Message Profile Selection window to change from inbound to outbound messages (see **Figure 83 on page 185**).

Figure 87 below shows that the selected outbound message contains both acknowledgements and order responses sent back to order requests as inbound messages.
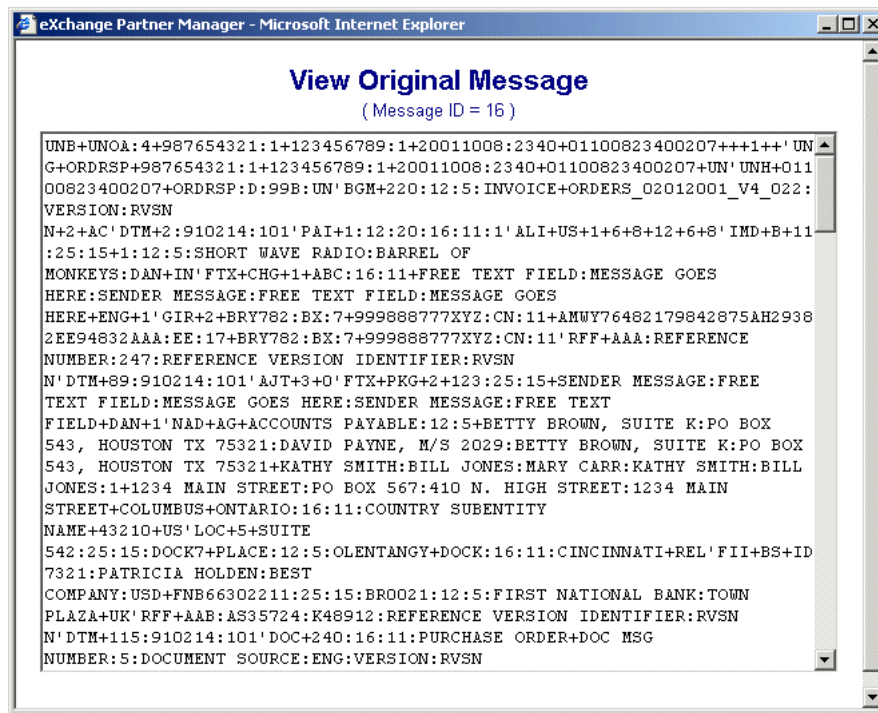
Figure 87 e*Xchange Message Details Window with Acknowledgements

Checking the View Original Message window again allows you to view the original message as shown below.

Figure 88 e*Xchange View Original Message Window, Outbound

If you want additional information on this outbound message, you can access the following information:

- Enveloped messages in the View Enveloped Message window (Figure 89)

- Acknowledgement messages in the View Acknowledgement Messages window (**Figure 90 on page 189**)

- Extended attributes in the View Extended Attributes window (**Figure 91 on page 189**)
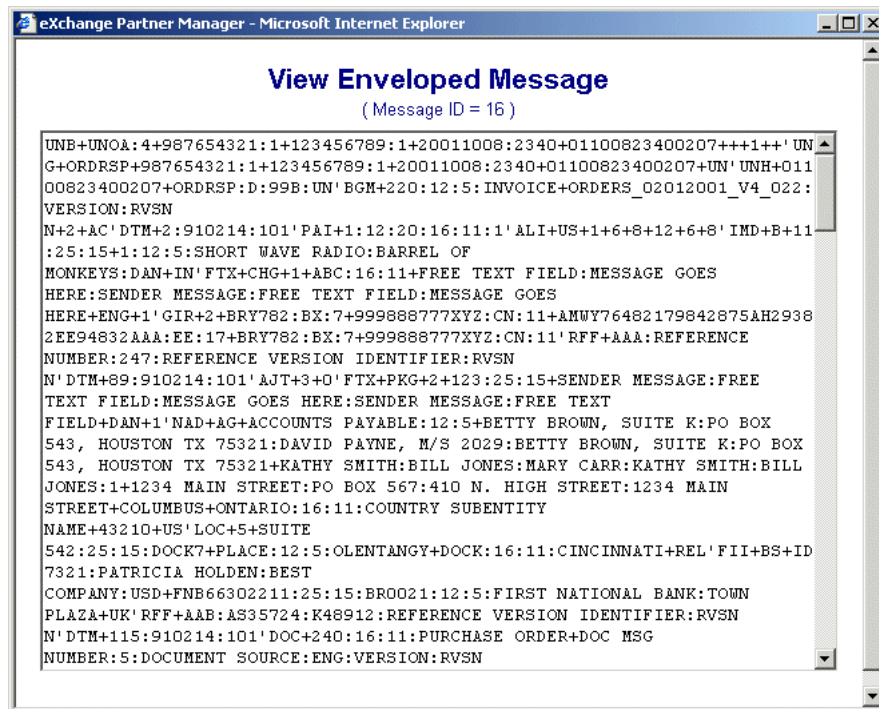
Figure 89   e*Xchange View Enveloped Message Window

Figure 90   e*Xchange View Acknowledgement Message Window

View Acknowledgement Message
(Message ID = 17)

```
UNA:+.?
*'UNB+UNOA:4+123456789:1+987654321:1+20010510:1945+01013119454531+++A+2+
'UNH+01013119454507+CONTRL:D:00A:UN++0'UCI+01100823400207+987654321+1234
56789+7'UCF+01100823400207+987654321:1+123456789:1+7'UCM+01100823400207+
ORDRSP:D:99B:UN:+7'UNT+5+01013119454507'UNZ+1+01013119454531
```
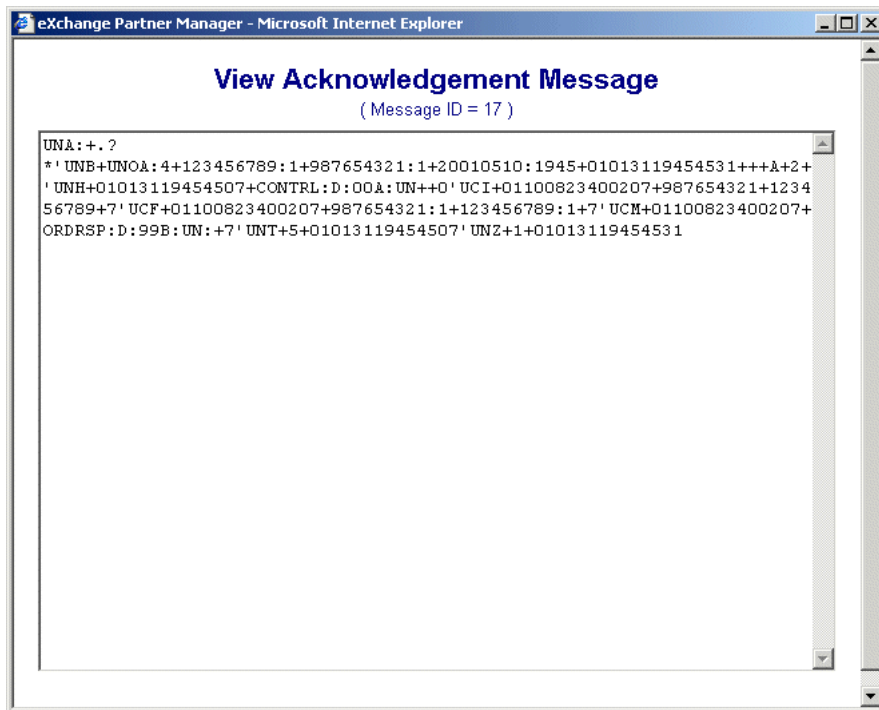
Figure 91   e*Xchange View Extended Attributes Window

View Extended Attributes

B2B Protocol: UN/EDIFACT-4B-Outbound
Message Profile: EDF_ORDRSPPurcOrdeRespMess_D99B
Original Message ID: 16

Attributes	Values
INTERCHANGE CONTROL REFERENCE	01100823400207
FUNCTIONAL CONTROL REFERENCE	01100823400207
MESSAGE REFERENCE NUMBER	01100823400207
Response TP Profile Msg ID	2
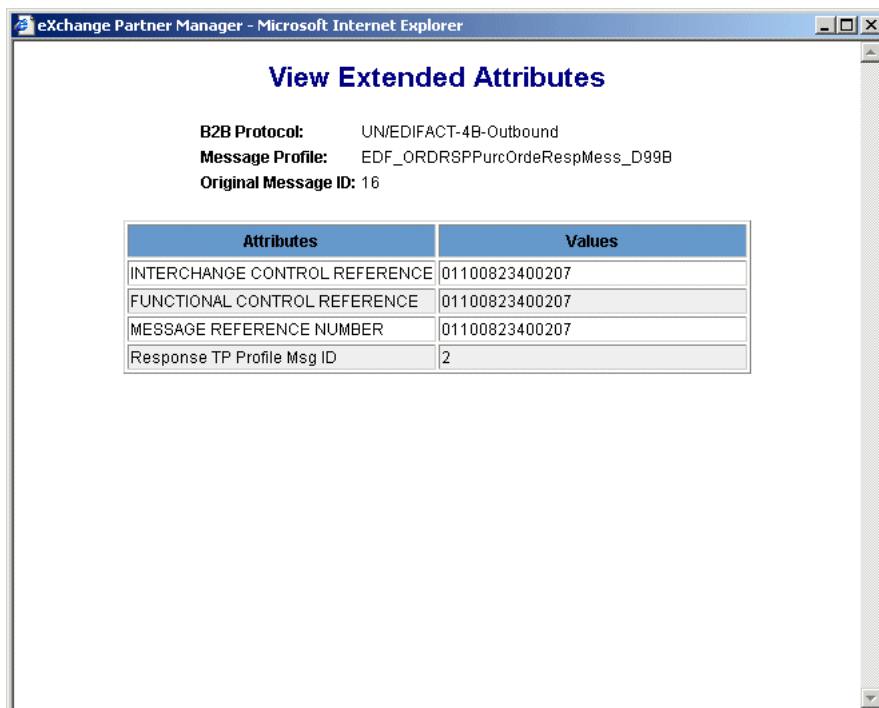
Error Tracking

You can use the e*Xchange View Error Data window as a troubleshooting tool, to view error details (see Figure 92) when any errors or problems occur with trading partner messages (click the red **Yes** in the Message Details window to access this GUI).

Figure 92   e*Xchange View Error Data Window
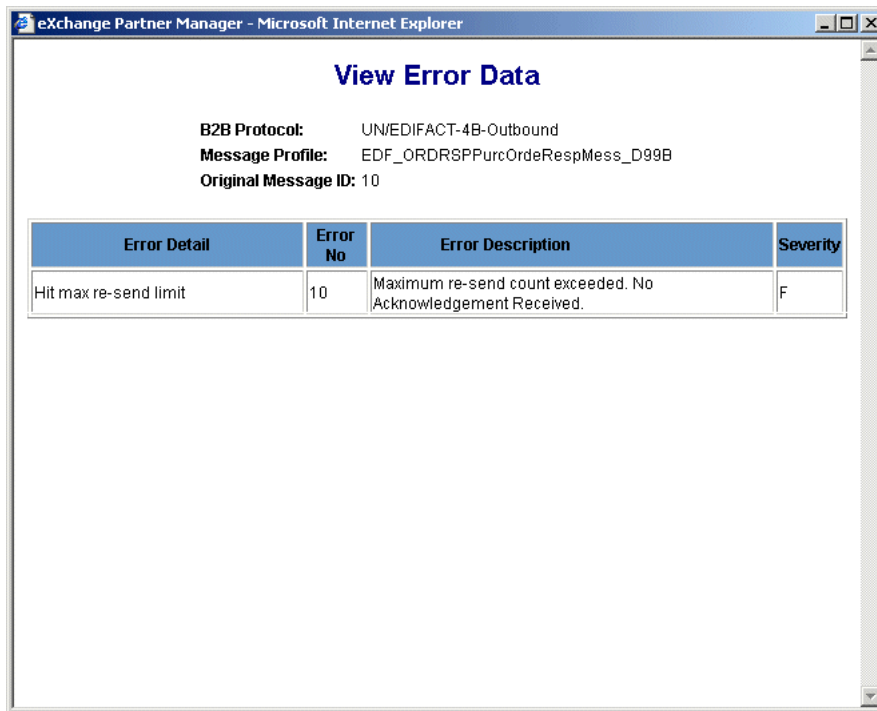
6.4.3 Implementing Changes

After a period of time, you may have to make changes to the eBI Suite. Changes are common as the needs of your end-users evolve and as additional external systems are added.

Do not make changes to the system hastily. Handle changes using the same process that was originally used to deploy your eBI Suite. Consider the change management process illustrated in **Figure 64 on page 163**. Applying this same process of planning, configuration, testing, migration, monitoring, and re-evaluation ensures a sensible deployment.

See **Appendix B** for a sample QA report that addresses continuing system maintenance and change issues.

6.5   Case Study Examples

The rest of this chapter provides the following examples of deploying the eBI Suite:

- **Scenario:** Uses e*Gate and e*Insight; see **"Case Study 1: Web Order Scenario" on page 191**.

- **Scenario:** Uses e*Gate, e*Insight, and e*Xchange; see **"Case Study 2: Expanded Web Order Scenario" on page 196**.

- **Scenarios:** Use e*Gate with Java-based features; see **"Case Study 3: Tracking Timecards and Payroll Scenario" on page 198** and **"Case Study 4: Receiving and Purchasing Scenario" on page 200**.

For the rest of this chapter, we provide examples of the Pre-transition Testing, Transition to Production, and Post-transition Maintenance phases of deployment. This section uses and continues the case study examples from **Chapter 5**.

6.5.1 Case Study 1: Web Order Scenario

In **Chapter 5**, we created an eBI Suite by using e*Insight and e*Gate. This system is used to process electronic orders placed via the company's Web site. At this point, the system has been designed, installed, and configured in the lab. The remaining steps are:

- Conduct unit, integration, and acceptance testing

- Export the entire lab system

- Migrate the export files to the production environment

- Import the configuration files in the production environment

- Monitor the performance of the live system

Pre-Transition Testing

Before transitioning the system to the production environment, the user performs all tests according to the test plan created during the design and development phase. The three types of testing to be performed are:

- Unit testing

- Integration testing

- Acceptance testing

Unit Testing

According to the test plan, the following components must be unit tested:

- **eX_Check_Inv** (e*Way)

- **eX_from_Check_Inv.tsc** (Collaboration Rules script)

- **eX_to_Check_Inv.tsc** (Collaboration Rules script)

- **eX_Ship_Ord** (e*Way)

- **eX_from_Ship_Ord.tsc** (Collaboration Rules script)

- **eX_to_Ship_Ord.tsc** (Collaboration Rules script)

- **eX_Out_of_Inv** (e*Way)

- **eX_from_Out_of_Inv.tsc** (Collaboration Rules script)

- **eX_to_Out_of_Inv.tsc** (Collaboration Rules script)

- **eX_Send_Status** (e*Way)

- **eX_from_Send_Status.tsc** (Collaboration Rules script)

- **eX_to_Send_Status.tsc** (Collaboration Rules script)

Testing the Collaboration Rules Scripts: The Collaboration Rules scripts are tested using the Monk Test Console. The best way to do this operation is to test each of the Collaboration Rules scripts in the order in which their Collaborations would execute. Start with the Collaboration that starts the business process—in this case the **START_BP** Collaboration in the **START_BP** e*Way.

The Monk Test Console requires properly formatted input. The input can be a file or it can be entered directly into the test console. By default, the output is displayed in the window. However, you can also redirect the output to a log file. This is a handy way to create the *input* file for the next Collaboration to be tested.
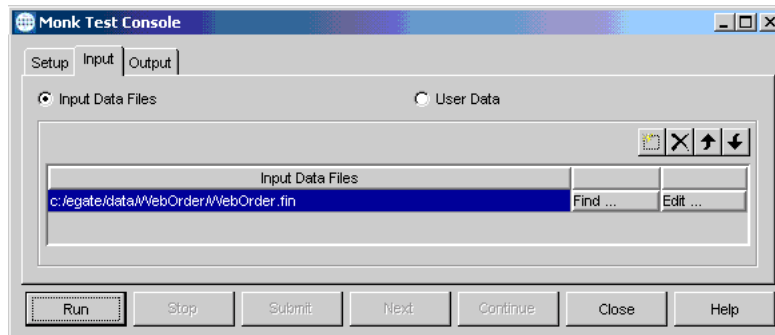
To test the script

1 Enter the name of the script you wish to test (**WebOrderIn.tsc**) in the **Input Script** box (see the following figure).

Figure 93   Monk Test Console—Setup Tab
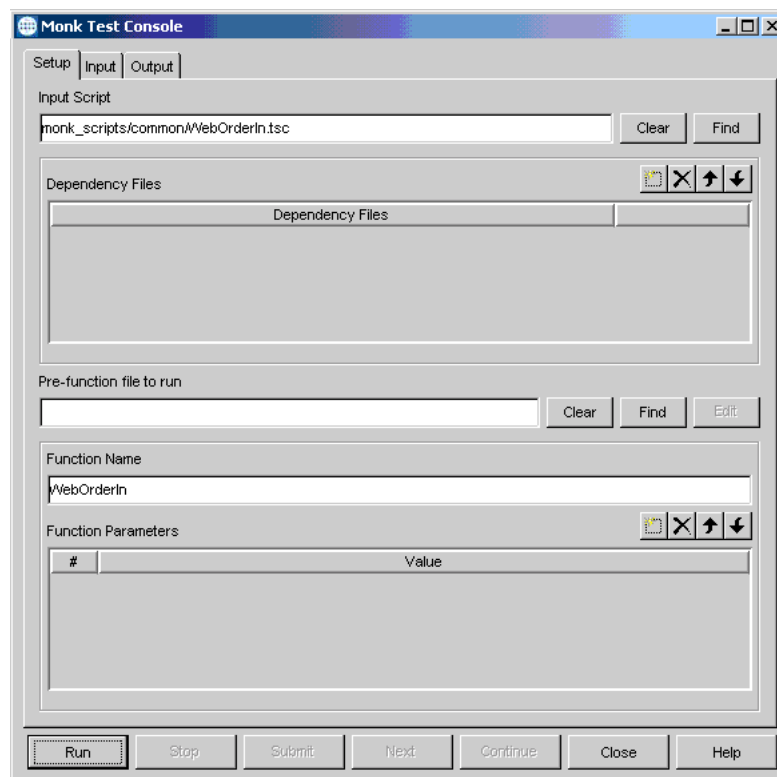
2 Select **Input Data Files** and use the file selection controls to specify a text file to use as input (see the following figure).
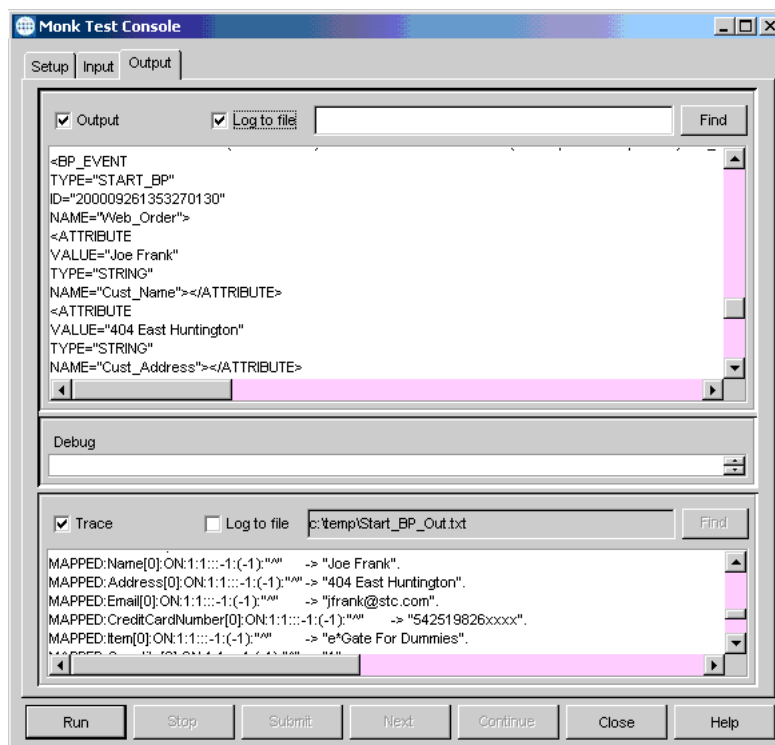
Figure 94   Monk Test Console — Input Tab

The input file mimics the data received from the Web server when an order is received.

3 Select the **Output** check box, and click **Run** to test the script (see **Figure 95 on page 193**).

Note: *For more information about the Monk Test Console, refer to the **e*Gate Integrator User's Guide**.*

Figure 95   Monk Test Console — Output Tab

The results of the transformation appear in the top text box. If the results are correct, select the **Log to file** check box to redirect the output to a text file. This text file is the input source for the next Collaboration you test.

Repeat this process for each of the Collaborations in the schema. Take care to test all conditional paths of the business process. For example, make sure you test both possible results of the decision gate: **Ship_Ord** and **Out_of_Inv**.

Testing the e*Ways: The previous process only tested the Collaboration Rules scripts for each of the Collaborations; it did not test the configuration files for the e*Ways. To test the e*Ways' configuration files, you must send actual Events through the e*Ways and examine the output.

This can be done by substituting file e*Ways for some of the other components in the schema. **Figure 69 on page 167** and **Figure 69 on page 167** illustrate how file e*Ways can be used to test an e*Way. The inbound file e*Ways can use the output files created by the Monk Test Console during the Collaboration Rules script testing.

Use the "Testing an inbound e*Way" **procedure on page 168** and "Testing an outbound e*Way" **procedure on page 168** to test each of the e*Ways in the schema.

Integration Testing

Now we have tested the Collaboration Rules scripts and the configuration files for all the components. However, we have not tested the logical routing between each of the components. The logical routing can be verified through integration testing.

Integration testing is the process of running the entire system through all of its paces in a lab environment. This can be challenging with complex systems. This emphasizes the importance of creating a thorough test plan during the planning phase. Use your test plan to test every possible condition that may arise in your system.

Acceptance Testing

After fully testing the system, the end-user who worked with you to create the test plan returns to the lab and conducts acceptance testing. The user must test the system according to the testing plan they helped to create. This may help avoid issues that can arise after the system is transitioned to the production environment.

Transition to Production

After fully completing the testing phase, the next step is to migrate the system to our production environment. In this scenario, the users must migrate the e*Insight Business Process and e*Gate schema to the final production location. The focus of this phase is to create a production system identical to the lab system.

To transition the system to production, use the following general steps:

- Export the business process and e*Gate schema
- Move the files to the production system
- Import the business process and schema into the production servers

Exporting the Lab System

Exporting the Web Order lab system requires the e*Insight business process as well as the e*Gate schema to be exported. These export files are copied over to the production environment before going live.

Exporting the Business Process: Use the e*Insight export feature to export the business process to an **.xml** file. In this case, we don't want to include the instance data; the instance data includes all the test instances from the lab testing.

The **.xml** file is, by default, named after your business process (**Web_Order.xml**) and is saved to the **\eInsight** directory.

Exporting the Schema: Click the **Export Schema Definitions to File** on the File menu and use the export GUI feature in the e*Gate Enterprise Manager. See the *e*Gate Integrator System Administration and Operations Guide* details.

Moving the Files

The business process export file (**Web_Order.xml**) and the e*Gate schema export file (**Web_Order.txt** or **Web_Order.exp**) are both relatively small. They can be archived and transferred using simple utilities such as Winzip and floppy diskettes. Take care to note the location of the files on the lab server. The files must be restored to the same locations on the production server.

The Registry files are larger — approximately 15 MB to 20 MB. Use the LAN connection to transfer these files.

Importing the Configuration Files

The business process and schema have been exported and copied from the lab server. The next step is to import these files into the production server.

Importing the Business Process: Use the e*Insight import feature to import the business process file (**\eInsight\Web_Order.xml**) into e*Insight. Use the same business process name as was used in the lab environment: **Web_Order**. Also, do not select the **Import business process instances** option.

Importing the Schema: Next, you must import the e*Gate schema and all of its associated files.

To import the schema

1 Click Import Definitions from File on the File menu.

   The Import Wizard GUI appears.

Note: *For information about importing e*Gate schemas, refer to the **e*Gate Integrator System Administration and Operations Guide**.*

2 Follow the easy steps provided by the wizard GUIs to import the schema and its associated files.

3 Create a new directory for the schema files. Name the directory

   \eGate\Server\registry\repository\Web_Order

4 Restore all the files from the lab into the newly-created directory. Take care to maintain the original directory structure from the lab environment.

Post-Transition Maintenance

With the system up and running, it is important to monitor the system's performance, check for errors, and make any needed changes.

Monitoring the Business Process

Use the e*Insight monitor mode to monitor the status of the business process instances. The Detail view can be used to view the specific progress of a particular business process instance (see **"e*Insight Monitoring Mode" on page 181** for details).

Monitoring the Schema

Use the e*Gate Monitor to view the status of the individual e*Gate components, such as e*Ways, BOBs, IQ Managers, and IQs. Problems with a specific component can be viewed in the **Status** tab (see **"e*Gate Monitor GUI" on page 170** and **"Using the e*Gate Monitor" on page 180** for details).

Assessing Future Needs

As the use of the system evolves, changes may eventually be needed. New external systems, new data requirements, and changes in policies can all lead to the need for changes in the business process. Use the change management process illustrated in **Figure 64 on page 163** to properly process any changes to the system.

6.5.2 Case Study 2: Expanded Web Order Scenario

The second scenario involves a system using e*Insight, e*Xchange, and e*Gate. This scenario takes the Web Order scenario from our previous case study and adds the element of communicating with trading partners. In **Chapter 5**, we designed, installed, and configured the system in the lab. The remaining steps are:

- Conduct performance testing and validation testing
- Export the entire lab system
- Migrate the export files to the production environment
- Import the configuration files in the production environment
- Monitor the performance of the live system

This case study highlights the differences brought about by the introduction of e*Xchange to the scenario.

Pre-Transition Testing

The pre-transition testing process for this scenario is similar to those in the previous one. The introduction of e*Xchange has little impact on the testing methods used. The Collaboration Rules scripts and e*Ways are put through the same unit testing procedures as those used in the previous scenario.

The process of integration testing is slightly different, because data movement must be tested as the e*Insight business process instance triggers an outgoing message to the trading partner via e*Xchange.

Once the unit and system tests are complete, the end-user can be brought in to perform the acceptance testing. The system is tested against the standards previously defined in the testing plan.

Transition to Production

The process of migrating the expanded Web Order system to the production environment is similar to the previous scenario. The main difference is that the trading partner profiles in e*Xchange are exported, copied, and imported to the production environment.

Exporting the Lab System

The process of exporting the configuration files in this scenario is nearly identical to the previous one. The additional steps for this scenario have to do with migrating the trading partner profiles.

Exporting the business process: Follow the same procedure to export the e*Insight business process to the .**xml** file as in the previous scenario (see **"Exporting the Lab System" on page 194**).

Exporting the trading partner profiles: Use the e*Xchange GUI to export the e*Xchange trading partner profiles to a .**exp** file. Take care to note the name and location of the file.

Exporting the schema: Follow the same procedure to export the business process to an export file as in the previous scenario (see **"Exporting the Lab System" on page 194**).

Moving the Files

With the addition of e*Xchange .**exp** file, the amount of data to be transferred from the lab environment to the production server has grown to the point where the data must be transferred via a LAN connection or with a mass storage device such as tape, CD-ROM, or Zip drive.

The files to be transferred are:

- e*Insight business process .**xml** file
- e*Xchange trading partner profile .**exp** file
- e*Gate full schema export .**zip** file

Verify the files are restored to the same path location as their original location on the lab server.

Importing the Configuration Files

The process of exporting the configuration files in this scenario is very similar to the previous one. Again, the main difference is in the importing of the e*Xchange trading partner profiles.

Importing the business process: Follow the same procedure to import the .**xml** file as in the previous scenario (see **"Importing the Configuration Files" on page 195**).

Importing the trading partner profiles: Use the e*Xchange GUI to import the .**exp** file into e*Xchange.

Importing the schema: Follow the same procedure to import the schema export file as in the previous scenario (see **"Importing the Configuration Files" on page 195**). Also copy the Registry files into the proper location on the production server.

Post-Transition Maintenance

With the system up and running, it is important to monitor the eBI Suite's performance, check for errors, and make any needed changes.

Monitoring the Business Process

Use the e*Insight monitor mode to monitor the status of the business process instances. The Detail view can be used to view the specific progress of a particular business process instance (see **"e*Insight Monitoring Mode" on page 181** for details).

Monitoring e*Xchange Messages

Use the e*Xchange Message Tracking feature to view the status of e*Xchange messages. Take note of any messages with errors and take the necessary corrective actions (see **"e*Xchange Message Tracking" on page 184** for details).

Monitoring the Schema

Use the e*Gate Monitor to view the status of the individual e*Gate components, such as e*Ways, BOBs, IQ Managers, and IQs. Problems with a specific component can be viewed in the **Status** tab (see **"e*Gate Monitor GUI" on page 170** and **"Using the e*Gate Monitor" on page 180** for details).

Assessing Future Needs

As the use of the system evolves, changes may eventually have to be made. New external systems, new data requirements, and changes in policies can all lead to modifications in the business process. Use the change management process illustrated in **Figure 64 on page 163** to correctly process any changes to the system.

6.5.3 Case Study 3: Tracking Timecards and Payroll Scenario

The third scenario involves a process using only e*Gate and is part of a larger system. This scenario sets up a communication example where requests for information are processed, and the results are forwarded to another external system for distribution. In **Chapter 5**, we designed, installed, and configured the system in the lab. The remaining steps are:

- Conduct performance testing and validation testing

- Export this modular operation/schema along with the entire lab system

- Migrate the export files to the production environment

- Import the configuration files in the production environment

- Monitor the performance of this operation as a part of the entire live system

Keep in mind that the transition-to-production steps for this system are similar to those in the previous case studies (1 and 2) except without e*Insight or e*Xchange plus with the addition of an extra components, that is, a BOB.

Pre-transition Testing

The pre-transition testing process for this scenario is similar to the previous case studies. In addition to the Collaborations and e*Ways, you must also set up the BOB and the Java Business Rules (in the Java Collaboration within the BOB) through the same unit testing procedures as those used in the previous scenario.

The process of integration testing is slightly different, because data movement is tested only in e*Gate (and not with e*Insight or e*Xchange).

Once the unit tests and system tests are complete, the end-user can be brought in to perform the acceptance testing. The system is tested against the standards that were previously defined in the testing plan.

Transition to Production

The process of migrating the Tracking Timecards and Payroll example to the production environment is similar to the previous two case studies. The main difference is that the additional components (the Java Business Rules/Collaborations and BOB) are exported, copied, and imported to the production environment.

Exporting the Lab System

The process of exporting the configuration files in this scenario is nearly identical to the previous one. The extra steps involve the additional components.

Exporting the Schema: Follow the same procedure to export the schema file as in the previous scenarios (see **"Exporting the Lab System" on page 194**).

Moving the Files

Again, the process of moving the files in this scenario is virtually identical to the previous two. The amount of data to be transferred from the lab environment to the production server may entail using a LAN connection or with a mass storage device such as tape, CD-ROM, or Zip drive.

The files that to be transferred are:

- e*Gate schema export file
- e*Gate Registry directory and files:
  \eGate\Server\Registry\Repository\Expanded_Web_Order

Take care to restore the files to the same path location as their original location on the lab server. Keep in mind that if this operation is part of a larger deployment, the Registry directory and files encompass the entire e*Gate environment.

Importing the Configuration Files

The process of exporting the configuration files in this scenario is very similar to the previous two. Again, the main difference is in the importing of the extra components.

Importing the Schema: Follow the same procedure to import the schema export file as in the previous scenarios (see **"Importing the Configuration Files" on page 195**). Also copy the Registry files into the proper location on the production server.

Post-Transition Maintenance

With the system up and running, it is important to monitor the e*Gate environment's performance, check for errors, and make any needed changes.

Monitoring the Schema

Use the e*Gate Monitor to view the status of the individual e*Gate components such as e*Ways, BOB, IQ Manager, and IQs. Problems with a specific component can be viewed in the **Status** tab.

Assessing Future Needs

As the use of the system evolves, you may eventually have to make changes. New external systems, new data requirements, and changes in policies can all lead to modifications in the business process. Use the change management process illustrated in **Figure 64 on page 163** to correctly process any changes to the system.

6.5.4 Case Study 4: Receiving and Purchasing Scenario

Steps involved in this scenario are similar to the previous one, so little additional commentary is necessary.

As the use of the system evolves, you may eventually have to make changes. New external systems, new data requirements, and changes in policies can all lead to modifications in the business process. Use the change management process illustrated in **Figure 64 on page 163** to correctly process any changes to the system.

6.6 Transition to Production: Summary

The proper use of a lab environment provides an excellent opportunity to verify and refine the system configuration that was implemented earlier in the deployment process. Focusing on the deployment plan ensures the smoothest possible eBI Suite deployment.

By thoroughly unit testing and system testing the system in the lab, costly errors are avoided and the end users are much more satisfied with the final results.

Going Forward

Once all the members of your management, your Deployment Project Team, and the end-users agree that the system is up and running according to plan, you have successfully finished the deployment of your eBI Suite.

If you have any questions about further system operation and maintenance, see the appropriate documents listed in **"Supporting Documents" on page 20** or contact SeeBeyond.

Note: **Chapter 7** *"Frequently Asked Questions" gives you some helpful hints and tips for best practices.* **Appendix B** *"Sample QA Report" also provides an example of the Quality Assurance report completed following an eBI Suite deployment.*

Frequently Asked Questions

This chapter lists some common questions that may be encountered during an eBI Suite deployment, and the answers to those questions. Here you can find a reference with good tips, helpful hints, and best practices.

Note: *It is recommended that you read this chapter before beginning your deployment.*

In This Chapter

7.1 Introduction: Using These FAQs

The purpose of this chapter is to make you aware of some of the questions to ask yourself prior to deploying the eBI Suite. They are a combination of hints, tips, and ways to obtain optimum performance from your system. This information can greatly help you deploy your system in the most efficient manner, and, at the same time, aid you in spotting problems to avoid.

7.2   Deployment FAQs

This section answers commonly asked questions about eBI Suite deployment procedures.

7.2.1   Setting Up eBI Suite FAQs

1   **Do we need a deployment road map? If so, how detailed does the map need to be?**

The answer is an emphatic yes! It may seem like overkill to plot your eBI Suite out to the smallest detail, but careful planning will result in a successful deployment of your system. Include the following in your deployment road map:

- **Analysis of Requirements Phase**

  The analysis of requirements phase initiates the project, defines the integrated system to be developed, creates top-level design documents, and produces a formal project plan. During this phase:

  - Examine your business needs.

  - Define the properties for your system to meet those needs, such as your performance requirements and system constraints.

- **Deployment Planning Phase**

  The deployment planning phase is the process of planning how the system will be built; that is, determining the procedural and data components needed and how these components will be assembled to form your integrated systems solution. It includes:

  - The development of design documents and diagrams that describe:

    - What each interface component will do.

    - How it will be done.

    - How each interface component works within the design architecture.

- **System Design and Development Phase**

  The system design and development phase begins after your system design plan has been worked out and management approved. It includes:

  - Filling in the details of the general system designs created during planning.

  - Coding and testing of interfaces and supporting architecture.

  - Integrating of the various components into your system and the testing of these components.

  - Documenting of all interfaces and components.

  - Building and testing of your environment.

  - Creation of a test plan to validate your system.

◆ **Pre-Transition Testing Phase**

The pre-transition testing phase (also called the customer testing phase) uses the requirements that were developed during the deployment planning phase, along with the test plans that were developed during the system design and development phase, to test the design and code against your requirements to ensure compliance. Your staff will be involved in this testing.

◆ **Transition-to-Production Phase**

The transition-to-production phase is when you move your interfaces into production.

◆ **Post-Transition Maintenance Phase**

The post-transition maintenance phase provides for fine-tuning your system after the transition to production.

2 **Should we chart how our eBI Suite flows?**

Yes. When creating the diagrams of your eBI Suite, make sure you chart all your incoming and outgoing processes as this gives you an idea of where your traffic is heaviest.

Include the following charts:

◆ **External systems**

- Include the type of data these systems will generate or receive.

- Include a communications diagram of everything that is outside of your eBI Suite.

◆ **eBI Suite**

- Include the Collaborations that will transform the data in your external systems chart.

- Include a components diagram with e*Way Intelligent Adapters and shows the direction in which data will flow within e*Gate and your eBI Suite.

As your system grows, the information you chart can be invaluable when deciding how to divide it up over additional computers (Participating Hosts). For more information on what to look for and make special note of in charting data flow, see **Chapter 3**.

3 **If we underestimate our total hardware needs for our eBI Suite, will this cause us trouble later?**

Hardware underestimation generally does not cause any problems. Because the eBI Suite is easily expandable, it can offer a performance boost when you add new processors and distribute your processes across your processors. In addition, you can add one or more computers (Participating Hosts) at any time and distribute your processes across them.

When planning hardware distribution, keep in mind that it is best to keep the network traffic between computers to a minimum, because this precaution lowers the chances of data-bottleneck problems between computers.

4   **If we do not sign up for e*Insight and e*Xchange, is it difficult to add them later?**

The e*Insight and e*Xchange systems are layered on top of e*Gate, which means that e*Gate must be installed first. There are not any problems with adding e*Insight or e*Xchange to your system at a later date. Depending on your hardware capacity and configuration, you may need additional computers and capacity to make these additions.

5   **What do I do if, later on, my total amount of data traffic grows? What if I want to add additional e*Gate components?**

Either case is not a problem. If, at any time you need to add additional e*Gate components to your system, such as e*Ways or BOBs, you can do so easily, using the Enterprise Manager. Keep in mind that any later redesign of your overall system capacity and configuration could entail the addition of extra hardware capacity. However, if you planned for later expansion during your deployment, such additions probably will not be necessary.

6   **How do I plan for redundancy and fail-over capabilities in e*Gate?**

The e*Gate environment has the following fail-over features:

◆ **Distributed operation:** This "normal" feature of e*Gate allows you to plan built-in fail-over features as a part of your overall system design. See **"Distributed Architecture Considerations" on page 76** for details.

◆ **Registry Replication:** See the *e*Gate Integrator System Administration and Operations Guide* for details.

◆ **Use of the Alert Agent:** See the *e*Gate Integrator Alert Agent User's Guide* for details.

◆ **General fail-over features:** See **Chapter 8** for details.

You can also run e*Gate release 4.5 and later at multiple sites in cooperation with a third-party high availability software. Specifically, e*Gate is completely compatible with the Microsoft clustering software. See **Appendix D** for details.

7.2.2 Performance Tuning FAQs

1   **What is a JMS IQ? Can it help with my system's performance?**

The Java Message Service (JMS) IQ is a new feature with e*Gate release 4.5 and later. Yes, they can help improve overall system performance. They are designed to store data better in memory and on the disks. These IQs generally run faster than SeeBeyond Standard IQs. You can view and administer them using the JMS Administrator GUI.

*Note: For more information on the JMS IQ, see the **SeeBeyond JMS Intelligent Queue User's Guide**. For more information on the IQ Viewer, see the **e*Gate Integrator Intelligent Queue Services Reference Guide**.*

2   **How many IQs can my IQ Manager handle?**

We do not recommend putting more than three IQs under the same IQ Manager. If the IQs are high-volume in nature, two is the maximum.

3 **How would you recommend setting polling frequency?**

Set the polling frequency of a route as close to real time as you want it. For example: 1 per sec or 1 per 15 min.

4 **What is subscriber pooling?**

Subscriber pooling is a method of distributing Events (packages of data) among multiple e*Ways for efficient processing (see **"IQ Subscriber Pooling" on page 92**).

5 **Is it possible to distribute components across multiple hosts?**

Yes. If your Registry Host slows down because your system is running out of memory or CPU cycles, you can distribute e*Gate components (e*Ways, BOBs, IQ Managers, and Monitors) to a Participating Host or Hosts. Distribute the components by moving them to the desired Participating Host in the Enterprise Manager.

6 **When setting up Event Type Definitions (ETDs), what are my parsing considerations?**

Parsing considerations can be memory intensive. Because of this, when you set up your ETDs, you must decide what is more important: performance or maintainability.

If performance is more desirable, define the simplest ETD possible for your Collaboration to work. This design allows your Collaboration to perform quickly. At the same time, the lack of detail will make your Collaboration hard to maintain.

7 **Can I configure e*Gate to notify me when my disk is full?**

Yes. The disk-usage thresholds mechanism in e*Gate sends out an alert notification when disk usage on a selected drive or partition exceeds or drops below the set maximum and minimum limits that are set in the **Disk Threshold Settings** dialog box in the Enterprise Manager.

After the disk or partition threshold has been set, the setting can be reset. To do this, on the **Disk Threshold Settings** dialog box, select a disk or partition to monitor and click **Change**. When the **Threshold Properties** dialog box opens, reset the threshold limits.

8 **How can I configure e*Gate to notify me when an e*Way receives too many or too few Events?**

When the number of Events exceeds or drops below the set threshold limits and a notification has been sent, e*Gate creates a script that either creates an additional BOB to handle the overload or shuts down the BOB that is currently not needed.

9 **Is there a way to configure e*Ways and BOBs to improve performance?**

Yes. If there are certain times during the day when e*Ways and BOBs are not used, you can configure them to shut down.

The e*Gate interactive monitors have three commands (**reload**, **suspend**, and **activate**) that enable you to control when e*Ways and BOBs are active. The **suspend** command takes a component "off-line" but does not end the executing process until the component finishes processing any in-process Events, at which point it goes into a "wait" state. The **activate** command brings the component back on-line. The

reload command only reloads e*Ways for which you must explicitly reload configuration changes.

7.2.3 Hardware FAQs

1 Are there set hardware size limitations?

You must ask yourself, "Is my system adequate for my eBusiness requirements?" To answer this, you must carefully plan your system prior to deploying it.

2 How much CPU speed, RAM, and disk space are needed to run the eBI Suite?

For Windows NT

- ◆ 1 GB of RAM
- ◆ 20 GB disk space
- ◆ Pentium III-class 866 MHz CPU

For UNIX

- ◆ 1 GB of RAM
- ◆ 20 to 30 GB disk space
- ◆ 400 to 450 MHz CPU

3 What is the smallest size monitor I can use?

17-inch color monitor

Note: *For complete details on determining your eBI Suite hardware requirements, see* **Chapter 4***.*

7.3 General FAQs

1 Is it possible for me to generate an Event or an Alert?

Yes. An **iq-put** (Monk function) generates an Event by placing it on the output IQ, but does not commit it to the IQ until the Monk transformation or identification function returns successfully.

If the Monk function is operating under the Monk Collaboration service and the transformation only generates a single Event, it does not have to make an explicit call to **iq-put** to forward the Event to the queuing system.

Include this call if a Monk Collaboration generates more than one output Event.

The Monk Collaboration Service enqueues the returned string to the default Event Type vector. The output Event Type and input Event Type must be from the list of configured Event Types that the component is able to receive and produce. The input Event Type is included to help maintain the history of the Event as it passes through the system.

All Events of lower priority level are dequeued before any Events of a higher priority level. Priority zero Events are dequeued first. In typical usage, all calls to this function will be made with the same priority level.

An **Event-send** issues a Monitoring Event from any Monk script, which in turn generates an Alert.

Events can use the standard SeeBeyond Event codes, or a "user Event" code you can use to communicate status conditions of user-created applications. For more information on Monk functions, see the *Monk Developer's Reference*. For more information on e*Gate monitoring, see the *e*Gate Integrator Alert and Log File Reference Guide*.

2 **Is it possible to execute Java using e*Gate?**

Yes. The Java Collaboration Service (JCS) enables you to develop external Collaboration/Business Rules that will execute e*Gate business logic using Java. The JMS IQ Manager provides a point of contact for one or more external applications. In general, JCS is appropriate for internal (to e*Gate) Java processing or synchronous processing with external applications.

e*Gate has GUI support for Java programming using the Java Collaboration Rules Editor in the Enterprise Manager. You can also create Java-based ETDs using the Java ETD Editor. For details, see the *e*Gate Integrator User's Guide*.

Note: *For detailed information on the operation of e*Gate's Java-enabled features, see the* **e*Gate Integrator Collaboration Services Reference Guide**. *The* **Java Generic e*Way Extension Kit Developer's Guide** *is also included on the installation CD-ROM.*

3 **Can I call a Java application on a different computer? For example, if e*Gate is running on NT and the Java application is running on a UNIX computer.**

The Java entry points must run in an e*Gate component. However, they may invoke a remote Java service, such as an EJB.

4 **Is it possible to execute C language using e*Gate?**

Yes. The C Collaboration Service (CCS) enables you to develop external Collaboration/Business Rules that will execute e*Gate business logic using C language. See the *e*Gate Integrator Collaboration Services Reference Guide* for details.

5 **What is the advantage or disadvantage of using non-standard IQs, such as Oracle IQ or MQSeries IQ?**

There may be some compelling reasons to use the database IQs. For example, if your organization has invested in monitoring, optimization, or recovery tools. In general, however, SeeBeyond Standard IQs are the simplest while JMS IQs are the fastest. See the *e*Gate Integrator Intelligent Queue Services Reference Guide* for details.

6 **Why do some Java ETDs appear to be read-only?**

When you create a new ETD using the ETD Builder Wizards, the ETD you create is read-only (with rare exceptions). You can use read-only ETDs as source and destination Event Types in Collaboration Rules. You can also use read-only ETDs as external templates for a custom ETD, but you cannot make or save any changes to

the structure or elements of a read-only ETD. For more information on creating an ETD with the ETD Builder wizards, see the *e*Gate Integrator User's Guide*.

Note: *In some cases, **.xsc** files in the default schema can only be opened in read-only mode.*

7 **How can I get data from an Access database to my application running on UNIX?**

In order to access any database from a different platform using ODBC, you can use an ODBC driver.

Because Access does not provide any communications layer, you cannot access it remotely. You must run the ODBC e*Way on the server that runs Windows NT or Windows 2000. If it runs Windows 95, you can share the drive with the NT computer where your e*Way is running and access it from Windows NT.

For more information on determining hardware requirements, see **Chapter 4**.

8 **Can I integrate different versions of e*Gate, e*Xchange, and e*Insight with each other?**

Some versions of e*Xchange and e*Insight are compatible with different e*Gate releases. For example, both e*Insight 4.5.2 and e*Xchange 4.5.2 are compatible with e*Gate 4.5.1. For more information on these compatibilities, see the System Requirements sections of the *e*Insight Business Process Manager Installation Guide* and the *e*Xchange Partner Manager Installation Guide*.

Important: *If you plan to install different versions with each other, you must install e*Gate 4.5.1 **before** installing e*Xchange 4.5.2 or e*Insight 4.5.2.*

7.4 Service FAQs

1 **Does SeeBeyond offer any services that could help me with my deployment?**

Yes. SeeBeyond offers the following services:

- **Architecture Review Service**
  SeeBeyond's Architecture Review service provides you with a configuration and deployment road map to help you achieve your company's goals. SeeBeyond's experienced consultants examine your architecture and network design to assess suitability and determine the best use of SeeBeyond products within this architecture.

- **Implementation Service**
  As a follow-up to the Architecture Review, SeeBeyond can provide consultants to contribute to the successful deployment of your new or modified architecture. SeeBeyond consultants participate as members of your Deployment Project Team to guide the development process, assist in SeeBeyond software configuration, and transfer SeeBeyond product implementation skills to your staff. SeeBeyond project managers, system analysts, and application developers work closely with your staff to accelerate the project schedule and ensure success.

2   **Can you tell me a little about your service team?**

SeeBeyond's Professional Services division is comprised of experienced, technically savvy integration professionals. Our consultants excel in developing and deploying eBusiness integration solutions, as well as supporting the strategic and tactical goals of your organization.

3   **How can I upgrade from DataGate to e*Gate?**

Upgrading from e*Gate release 3.6 to 4.5 or later is facilitated by a set of user-friendly upgrade tools contained within the system. For detailed information, see the *e*Gate Integrator Upgrade Guide*.

Deploying for High Availability

This chapter describes how you can design eBI Suite for high availability in case of failure.

In This Chapter

- **"High Availability in e*Gate: Overview" on page 210**
- **"Product Features, e*Gate, and High Availability" on page 210**
- **"Sample Scenarios" on page 213**

8.1 High Availability in e*Gate: Overview

Implementing high availability in an e*Gate deployment involves eliminating single points of failure in the network configuration. e*Gate's architecture lends itself to a distributed and redundant configuration that is fault tolerant, which addresses certain high availability issues. However, high availability requires a complex network solution that goes beyond the e*Gate application itself.

Note: *For details about configuring the basic e*Gate environment, see the **e*Gate Integrator User's Guide**.*

High availability solutions are a combination of hardware, network configuration, and software to manage the high availability, as well as the applications that must be failed over. This chapter clarifies the role e*Gate plays in a high availability configuration. It also includes examples of e*Gate environments that take advantage of e*Gate's built-in high availability features.

For information about implementing Microsoft Windows 2000 clustering software, refer to **Appendix D**.

8.2 Product Features, e*Gate, and High Availability

This section describes the e*Gate features that a high availability design must address.

8.2.1 The e*Gate Registry

The e*Gate Registry holds all the necessary information for each e*Gate component to run. Each Participating Host (also called a node) authenticates with the Registry through the schema name and logical Control Broker name. For these reasons, all e*Gate Participating Hosts have the potential to run the e*Gate Control Broker and any schema.

The key design criterion is to ensure that no configuration is tied to the files on any given physical Participating Host. If you lose all systems except the Registry, you can install new systems as Participating Hosts, name them the same as the original systems (or change the host names configured in the e*Gate schema), and take on all the original configurations by connecting to the Registry through the same logical component names.

8.2.2 Registry Replication

The e*Gate Registry can be installed in a replicated configuration. A primary Registry sends all schema changes to one or more secondary Registries. e*Gate Control Brokers and other components can be assigned to more than one Registry. The components attempt connection to the first Registry in their Registry list. If this attempt fails, the component attempts the next Registry in the list. Assuming that the failing Registry process is not on the same computer as the Participating Host, this feature eliminates the Registry connection as a point of failure.

You can use the Registry Replication option independently or in conjunction with the scenarios described in this chapter. The main purpose of Registry Replication is to duplicate the Registry contents in additional locations and allow the other e*Gate components — for example, Control Brokers, e*Way Intelligent Adapters, e*Way Connections, Multi-Mode e*Ways, and IQ Managers — when necessary, to attempt to connect to a list of Registry Hosts instead of a single Registry Host.

*Note: Registry Replication does not allow Participating Hosts to implement high availability, nor does it cause the replication (backup) of transaction data in the IQs. For more information on Registry Replication, see the **e*Gate Integrator System Administration and Operations Guide**.*

8.2.3 Multiple Participating Hosts

Management of e*Gate components in a distributed network environment is dependent upon the number of Participating Hosts and their configured resources. All distributable components, including Control Brokers, have logical names that are independent of physical location. This feature results in a configuration where publish-and-subscribe information is dependent on logical names and independent of host names.

Participating Hosts have one property that sets the host name, and it is this name that e*Gate components use to find one another, along with the port number. Because of this property, it is easy to reassign host names and ports, or move e*Gate components to other computers without changing the basic e*Gate configuration.

8.2.4 IQ Subscriber Pooling

A typical load-balancing scenario involves an Event (unit of work or data package) being published as a shared Event Type. Then, multiple components can subscribe to the same Event. If the first component is busy processing the Event when the next Event is published, a peer component in the IQ subscriber pool can read the new Event and process it in parallel.

On a single computer, replications can be scaled up, as necessary, until you reach the memory resource limits and the CPU is fully utilized. In addition, the distributed architecture of e*Gate allows subscriber pooling to continue across additional computers, effectively removing such resource limitations.

Note: *For more information on IQ subscriber pooling, see* **"IQ Subscriber Pooling" on page 92**. *For additional details, see the* ***e*Gate Integrator Intelligent Queue Services Reference Guide****.*

8.2.5 System High Availability Methodology

In the event of a failure, unaffected e*Gate components continue to run independently of the Control Broker of the host (node) suffering the failure. Therefore, consider the following general maximum-availability design methodologies:

- If an e*Gate host suffers a failure causing it to be inaccessible, enable all of its critical processes to provide high availability to other hosts. If any non-critical processes are involved, do not cause them to fail over. This design lessens the additional load on remaining active hosts.

- The system attempts to restart the e*Gate Registry (service or daemon) several times after it goes down. After several unsuccessful restarts, you can then allow the Registry to fail over to the Secondary Host.

- The system attempts to restart an e*Gate Control Broker several times after it goes down. If the Control Broker cannot be restarted, do *not* allow it to independently fail over to a Secondary Host.

Note: *If a down Control Broker cannot be restarted, the* **NotificationQueue** *directory in the client directory of the Control Broker may need to be deleted.*

Control Broker High Availability

If a down Control Broker is failed over, the child processes, such as e*Way Connections, Multi-Mode e*Ways, and BOBs may be left running on the failed host. When the failed host's Control Broker is restarted, these processes are likely to be non-responsive. The high availability policy for a Control Broker must include additional scripting to ensure any child processes on the failed host are stopped before the Control Broker is restarted.

Note: *High availability software can define different policies for which conditions may activate the feature, retry behavior on the services/daemons, and dependencies between the services/daemons.*

8.3 Sample Scenarios

These scenarios discuss e*Gate high availability configurations and their related characteristics, as examples only. They are meant to show possible high availability design methodologies and do not constitute recommendations or preferred practices.

8.3.1 e*Gate with Standby Host

The following figure shows an example of how to configure e*Gate hosts in a clustered environment that provides a standby host.

Figure 96 e*Gate in Clustered Environment with Standby Host
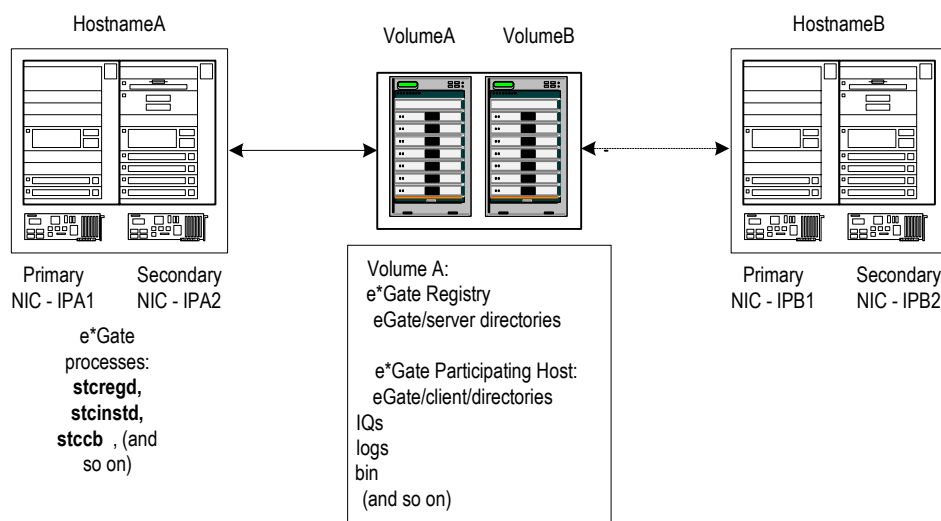
Example Characteristics

This example has the following hardware and software configuration:

- Two servers (Primary and Secondary Host)
- An external storage array in a redundant (RAID 1-5) configuration to allow the array to become available independently of the servers
- Each server (5 GB of RAM) with the equivalent amount of swap
- Each server with a local disk where the operating system (OS) is installed
- Each server with two network interface cards (NICs) in a redundant configuration
- Third-party high availability software (for example, HACMP, Verta, or Microsoft) installed on the servers' disks with the OS

The following sample software is installed on the external array:

- e*Gate Integrator
- e*Gate command scripts

This e*Gate environment has the following high availability characteristics:

- The e*Gate Registry and e*Gate Participating Host processes are running on the Primary Host.

- All the e*Gate files, such as binaries, log files, and IQs, are located on the storage array.

- Two TCP/IP addresses and host names are assigned to the Primary Host.

- Two TCP/IP addresses and host names are assigned to the Secondary (standby) Host.

Figure 97 on page 214 shows the clustered environment with e*Gate failed over to the Secondary Host.

Figure 97   e*Gate in Clustered Environment Failed-Over State
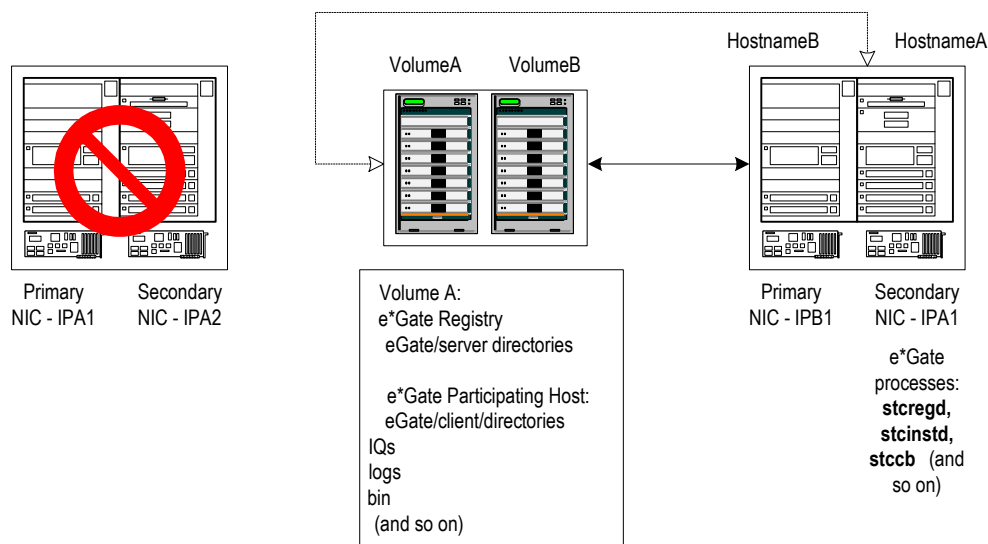
High Availability Processes

If the Primary Host goes down, the following high availability process takes place:

- The Secondary (standby) Host computer takes over the primary IP address and Primary Host name.

- With two NICs, it is possible for both host names and both IP addresses to resolve to the Secondary Host.

- High Availability software calls scripts to start up e*Gate processes on the Secondary Host. The e*Gate Registry starts up as the same Registry name. Because the Primary Host name has been taken over, the Participating Host processes start up with the same host-name parameters they had on the Primary Host.

- Pre-existing processes on the Secondary Host can continue to run after maximum-availability processes have been implemented. However, keep in mind that these processes must share the memory and CPU with the failed-over processes. To avoid this problem, you can fail over to a continually running (reserve) standby host or to

a test computer where the former processes can be cleared to make room for the failed-over processes.

▪ Pre-existing processes on the Secondary Host can continue to run after maximum-availability processes are implemented. Therefore, it is important to remember that these processes share the memory and CPU with the failed-over processes. To avoid this situation you may want to consider the alternative of failing-over to a continually running (reserve) standby host or to a test computer where the former processes can be cleared to make room for the failed-over processes. Another possibility is to run both the primary and secondary as peers that share the load at all times, but are sized so that either one can handle the entire load. If one fails, the other can just work harder.

▪ If a test instance of e*Gate already exists on the Secondary Host, it may need to be shut down and cleared before starting up the failed-over e*Gate processes. You can ensure this precaution using high availability scripts.

8.3.2 Subscriber Pooling Without Server High Availability

The primary purpose of IQ subscriber pooling is to foster load balancing. Operations that require processing transactions in sequence may have limitations that warn against parallel processing with subscriber pooling. However, SeeBeyond provides techniques and functions to ensure that the proper sequence in processing is achieved.

In some cases, these sequencing techniques and functions may negate the advantages of subscriber pooling. For this reason, *do not consider IQ subscriber pooling as a general-purpose alternative to high availability.*

The following figure shows a system where IQ subscriber pooling is used for general parallel processing

Figure 98 Subscriber-Pooled Configuration for Parallel Processing
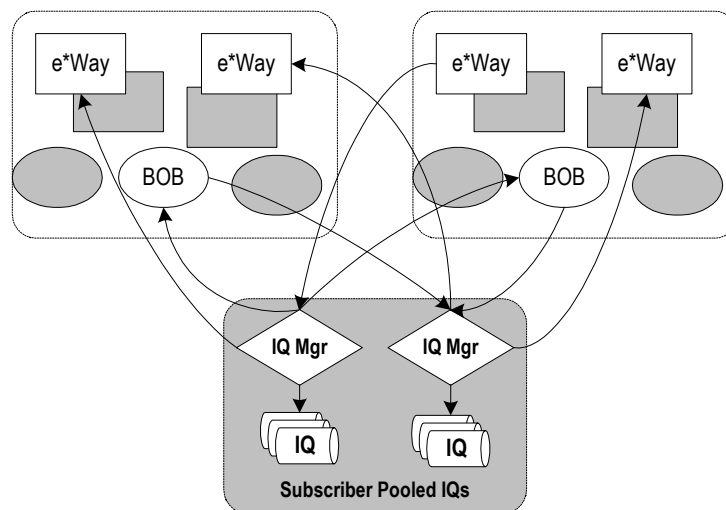
Example Characteristics

Hardware: This example has three host computers, and two of them are processors. The BOBs and e*Ways process data on theses computers. One computer is for IQs only.

Configuration: The subscriber components are redundant and run from both the host computers.

The IQs are configured as follows:

- They are subscriber-pooled so their operation is load-balanced across software processes and also across host computers.

- The writes to the IQs must travel via the network, so there is some processing overhead used in writing to IQs across two computers versus writing only to a local hard disk.

High Availability Processes

Figure 99 shows how the IQ subscriber pooling configuration operates if one non-IQ host computer fails.

Figure 99   Subscriber-Pooled Configuration with Non-IQ Failure
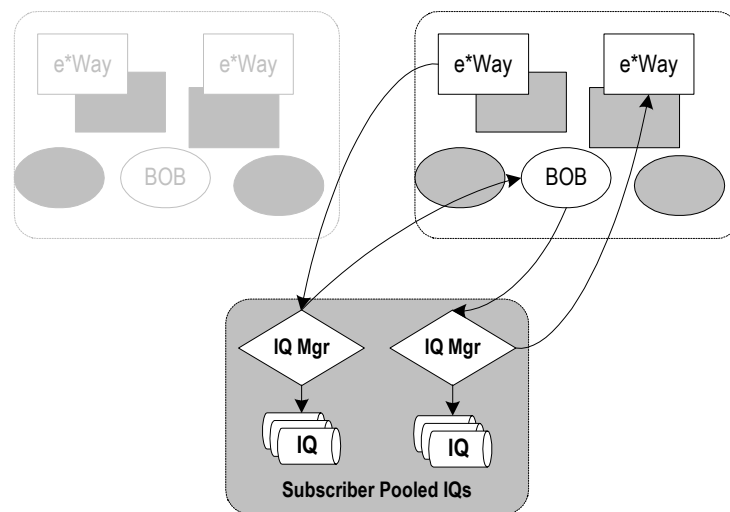
In this type of setup, the distributed configuration allows the remaining host computer to continue processing, but now it handles the entire load. The throughput of Events may decrease by a small amount, but none of the functionality is lost.

However, the following figure shows what happens if the host computer *with* the IQs fails.

Figure 100   Subscriber-Pooled Configuration with IQ Failure
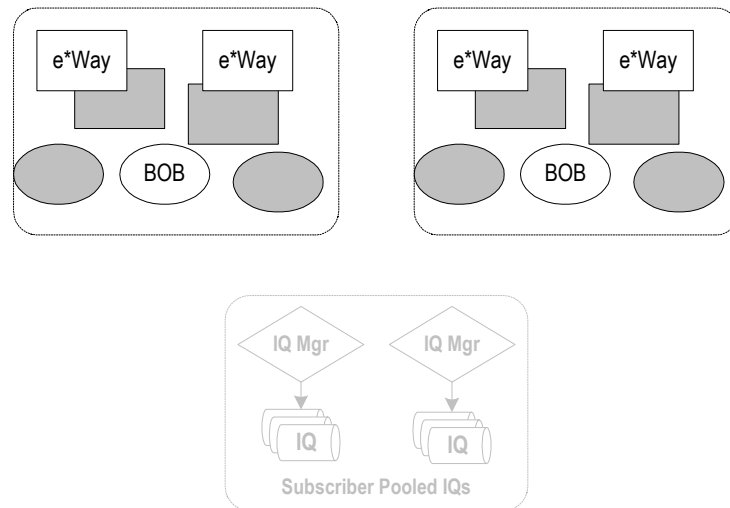
In this case, if the IQs are writing to a local hard disk, they become unavailable.

8.3.3 Subscriber Pooling With Partial High Availability

To avoid the problem shown in **Figure 100 on page 217**, you must combine some type of additional high availability process with IQ subscriber pooling.

The following figure shows the same subscriber-pooling configuration as the one explained in the previous example, except that additional maximum-availability characteristics have been designed in the e*Gate environment.

Figure 101 IQ Subscriber Pooling with High Availability
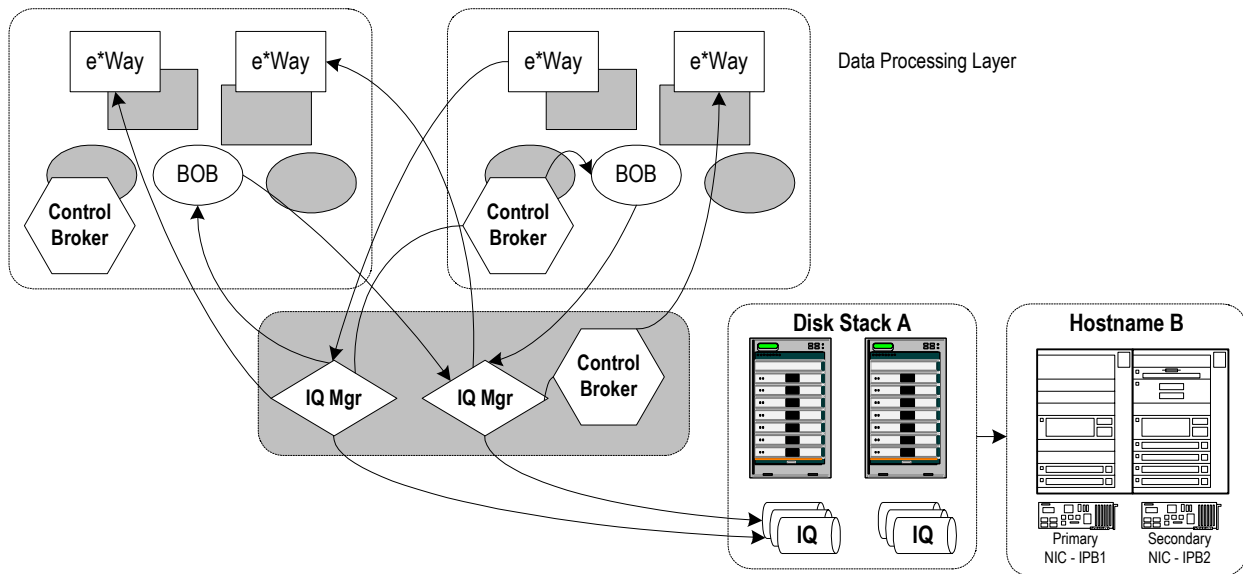
As shown in the previous figure, you can use IQ subscriber pooling for application-level high availability. External storage and additional setup is required to remove all single points of failure. In this scenario, subscriber pooling is used in conjunction with an external hard disk array and high availability software. The additional software ensures a smooth, immediate transition of processes to the external disks, in case high availability is needed.

Deployment Surveys

The following pages contain a sample survey. Surveys are excellent tools to gather information from a variety of people and departments in your organization. Use this information to help you take their needs and requirements into account during the analysis and planning phases of the eBI Suite deployment (see **Chapter 3**).

You can copy these pages and use them, or you can use this survey as a guide to help you make up questions of your own.

In This Chapter

A.1 System-Specific Information

What existing systems/applications do we need to connect?

_____

_____

_____

_____

_____

_____

How do we want to connect to these systems/applications?

_____

_____

_____

_____

_____

_____

What person (or department) is in charge of these existing systems/applications?

_____

_____

_____

_____

_____

_____

What are our data requirements in terms of the types of data we need to process?

_____

_____

_____

_____

_____

_____

What are our data requirements in terms of the processing speeds we need and the volume of data we need to handle?

_____

_____

_____

_____

_____

_____

What are our current system/hardware limitations and constraints?

_____

_____

_____

_____

_____

_____

_____

Is our existing equipment capable of scaling up to meet future demands?

A.2 Operation and Performance

What are our system performance requirements?

_____

_____

_____

_____

_____

_____

What are our internal security requirements?

_____

_____

_____

_____

_____

_____

Do we need to trade large amounts of data with other businesses?

_____

_____

_____

_____

_____

_____

What are our data security needs in transactions with other businesses?

_____

_____

_____

_____

_____

_____

Do we have all the necessary information regarding these trading partners' data formats?

_____

_____

_____

_____

_____

_____

What are our error-handling and data validation requirements?

_____

_____

_____

_____

_____

_____

Can we bring all of our external systems online with eBI Suite at once, or do we need to roll out in phases? Have these phases been identified?

_____

_____

_____

_____

_____

_____

_____

A.3 Personnel and Training

Do we have personnel trained and able to deploy the eBI Suite and what are their names?

_____

_____

_____

_____

_____

_____

Do we have personnel trained and able to maintain the system after deployment and what are their names?

_____

_____

_____

_____

_____

_____

Have we scheduled any necessary training classes with outside vendors?

_____

_____

_____

_____

_____

_____

A.4 Business Planning

What business processes do we want the eBI Suite to help us enable? Do we have any documents that describe our business processes?

_____

_____

_____

_____

_____

_____

Do we currently have the facilities in place to house the equipment needed for a Production environment? Have we allowed space for a Sandbox/lab environment?

_____

_____

_____

_____

_____

_____

Are these facilities scalable enough to grow as our systems grow?

_____

_____

_____

_____

_____

_____

_____

What are our record-keeping and documentation needs for the deployment project?

_____

_____

_____

_____

_____

_____

_____

What will our record-keeping and documentation needs be after deployment?

_____

_____

_____

_____

_____

_____

_____

What important time/schedule and resource allocation issues do we need to take into

account in creating our Deployment Project Plan?

_____

_____

_____

_____

_____

_____

_____

Are there additional issues we need to know and take into account not addressed earlier in this questionnaire? Can you give us information on these issues?

_____

_____

_____

_____

_____

_____

_____

Sample QA Report

The following pages contain a sample Quality Assurance report. This type of QA report would typically be created following an eBI Suite deployment. QA reports provide a useful record of the deployment process and any remaining punch-list items for all parties involved. Such a report can also be useful in the event of personnel changes where details of the deployment need to be retrieved in the future.

This section is based on an actual QA report created following an eBI Suite implementation. The company name has been changed.

Although this QA report documents an e*Gate deployment, the same principles would apply to a deployment using the e*Insight Business Process Manager and the e*Xchange Partner Manager.

Note: *The actual company's name in the report has been replaced by the fictional "Diamond Financial."*

In This Chapter

- **"Introduction" on page 231**
- **"Schema Components" on page 232**
- **"Overall Design Objectives" on page 236**
- **"Environments and Source Control" on page 239**
- **"Run-Time Management" on page 239**

B.1 Introduction

B.1.1 Background

The e*Gate environment provides the Event transformation and bridging services used to integrate Diamond Financial's "Diamond Sales Force" (DSF) client/server and mainframe applications which utilize MSMQ and MQSeries messaging technologies. The Application Architecture has identified many messaging transactions that will employ e*Gate, spanning the Intuitive MSMQ client/server domain, Diamond client/server systems, and Diamond's on-line and batch mainframe applications.

B.1.2 Objectives

This QA report was commissioned by Diamond Financial to examine the current implementations and usage of e*Gate with the following objectives:

- Recommend improvements for the ongoing development
- Identify good features/practices to be promoted in future developments

B.1.3 Approach

Information on the current development project was collated from various sources for analysis:

- Two day on-site visit participating in a single development cycle from development to Model Office test environments
- One day on-site visit for detailed examination of the e*Gate environments, configuration, structures and rules
- Existing project documentation (detailed in **"Document Inputs" on page 232**)

B.1.4 Document Inputs

Document	Version	Date
e*Gate Detailed Configuration Design	1.1	05.25.2000
Detailed e*Gate Schema Design for Diamond Financial	1.0	06.06.2000

B.2 Schema Components

B.2.1 General

Diamond's schema design employs a strong naming convention that identifies:

- Type of component (e*Way, Event Type, Collaboration, and so on)

- Source and target application system (DSF or LOB)

- Host name/number (DSFGWST01 or DSFGWST02; 01 or 02)

- Direction of data flow relative to e*Gate (IN or OUT)

- Application function name or transaction ID (ALEADSALE, CLOASGTPOL, and so on)

- Other functional areas (ERROR, GENERIC, LOADBAL, and so on)

A well-defined naming convention has the following advantages:

- Easy identification of an existing component's purpose within the schema

- "Automatic" allocation of names to new components

- Guaranteed uniqueness for component names

- Clarification of relationships between components

- Self-documentation at the outline level

- Ease of maintenance

The naming convention has been applied consistently throughout the schema at the component level. A few naming irregularities were noted at the sub-component level (for example, the root node of the **etdDSF_GENERIC_IN** definition is named **LOBResponse_XML**); however, these are not considered significant.

B.2.2 Event Types

Event Types fulfill a fundamental role within the e*Gate schema, driving the publish-and-subscribe mechanism used to route data between components. An Event Type name is effectively a label, which is attached to an Event when it is published to an IQ. A subscribing component declares which Event Types it wants to receive, with the option of taking Events from any source (anonymous subscription) or from specified

sources (defined as source Collaborations and providing pre-defined, point-to-point routing).

The Diamond schema utilizes the following algorithm for generating Event Type names directly based on Event content. Inbound Event Types are named:

etLOB_<function name>**_IN**

etDSF_<transaction id>**_IN**

where the "function name" or "transaction id" is extracted from the meta-data fields at the start of the Event. Such a formulaic approach to applying Event Type names has two significant advantages:

- Event Type identification scripts (**crDSF_GENERIC_DSF.isc** and **crLOB_GENERIC_LOB.isc**) require few lines of code and are therefore highly efficient.

- The addition of new transactions/functions in future developments will not require any amendment of the identification scripts.

B.2.3 **ETDs**

ETDs are classes of Event structures. They fall into two categories in the Diamond schema:

- Fixed-format Events for the back-end LOB applications
- Delimited XML Events for the front-end DSF applications

Both categories include a set of meta-data fields that are used to pass identification, routing, and error information.

LOB Structures

The fixed-format LOB structures have been manually built in the e*Gate ETD Editor based on documented transaction specifications produced by the DSF project. Abstracting common or repeating sub-structures into templates can optimize manually built ETDs. The use of templates reduces future maintenance work as template modifications are automatically promoted to parent structures.

Sub-structures that are common across different ETDs must be abstracted as global templates (saved as separate **.ssc** files). Sub-structures that are repeated within a single ETD must be abstracted as local templates (saved within the parent **.ssc** file).

The LOB structures utilize one global template, **tLOB_META_DATA.ssc**, which defines the meta-data fields common across all LOB transactions. There is scope for further abstraction of templates as several ETDs share common fields outside the set of meta-data fields.

For example, the following ETDs share at least nine common fields:

- etdLOB_CNONPENS01_IN.ssc
- etdLOB_CNONPENS02_IN.ssc
- etdLOB_CNONPENS03_IN.ssc

      ◆ etdLOB_CNONPENS04_IN.ssc

XML Structures

The XML structures have been automatically built with the e*Gate XML Converter tool based on document type definitions (DTDs) produced by the DSF project. The optimization of XML structures is achieved mainly through the generic features of the XML definition. A common data format can be defined based on the optional elements and element attributes of XML. Diamond's XML structures already exhibit strong optimization, for example, the **etdDSF_POLREQ_IN.ssc** structure is associated with 15 Event Types.

B.2.4 Collaborations

The Diamond schema uses point-to-point subscriptions (Event Type and source Collaboration defined) for all Collaborations. This mode provides a precise statement of the Event flows that will be handled by the running schema.

For future developments it would be worth considering the alternative of anonymous subscriptions (Event Type only defined). This mode is chiefly intended to provide an independent relationship between publishers and subscribers that has the following advantages:

▪ Only one subscription needs to be defined for each Event Type (using source <ANY>).

▪ New publishers of an Event Type can be added to the schema without any requirement to modify subscriber details.

The multiple subscriptions defined for the "PARSE_ERROR" Collaborations are a close candidate for anonymous subscription. However, as the same Event Types are used on both Participating Hosts, such a change would cause PARSE_ERROR Events to flow between the hosts and duplicate the delivery of Events to the cloned e*Ways. This point illustrates that careful consideration is required at an early stage in the schema design before opting for anonymous subscription.

B.2.5 Collaboration Rules

The Collaboration Rules scripts for the Diamond schema exhibit good abstraction of common functionality. This has produced transformation and identification scripts that are simple to comprehend and easy to maintain, consisting mainly of COPY statements and calls to abstracted functions.

Opportunities for further abstraction still exist. For example, the copying of meta-data fields is repeated across all transformation scripts (19 copying LOB to DSF, 3 copying DSF to LOB). A change in the meta-data field specification could potentially require changes to 22 Collaboration Rules scripts. Abstracting these copying requirements to two functions would greatly reduce the potential maintenance overhead.

B.2.6 Monk Library

As already noted in the previous section, the Diamond schema shows good abstraction of functions, which promotes code reusability and ease of maintenance. Functions have been developed to convert currency amounts, date formats and MSMQ Event IDs.

Some tidying up of the function library is desirable. Monk functions are automatically loaded from the **monk_library** directory and therefore duplicate copies of the function files (for example, in **monk_scripts\common**) must be removed. Also functions and function files prefixed with the developer's name must be removed before final testing and going live.

B.2.7 e*Way Configurations

The Diamond schema consists of two cloned sets of five e*Ways, each set running on one of the clone servers. Each set consists of:

- Inbound MSMQ e*Way receiving DSF transaction requests
- Outbound MSMQ e*Way sending DSF transaction replies and LOB application errors
- Outbound MSMQ e*Way sending DSF Event parse errors
- Outbound MQSeries e*Way sending LOB transaction requests and LOB Event parse errors
- Inbound MQSeries e*Way receiving LOB transaction replies and LOB application errors

The Event parse errors are generated within the confines of the running e*Gate schema. For example, a DSF Event parse error can be generated by the inbound MSMQ e*Way or the outbound MQSeries e*Way when processing a DSF transaction request.

This design is relatively easy to comprehend and certainly keeps both development and run-time maintenance to a minimum by reducing the number of active e*Ways. The design could be made slightly more transparent and symmetrical by adding a separate outbound MQSeries e*Way to handle LOB Event parse errors.

Both types of e*Way, MSMQ and MQSeries, are installed with a set of standard version functions that handle the requirements of interacting with the external system (connection establishment and verification, getting and putting of Events, acknowledgements and shutdown). Modifications to the standard functions have been implemented for both types of e*Way in the Diamond schema.

The outbound MQSeries e*Way utilizes a modification to the **Process Outgoing Message Function** which sets the MQ Event priority when putting an Event on the IQ (function name: **MQ-stdver-proc-outgoing-set-priority**). Modifications were previously recommended to implement the **MQ WaitInterval** option on MQGET and the use of transactional Units of Work on both MQGET and MQPUT. These recommendations were reported as implemented but not observed on the "diam015054" workstation.

The MSMQ e*Ways utilize a modification to the **External Connection Establishment Function** which extracts the IQ global unique identifier (GUID) from a flat file based on

a logical IQ name (function name: **MSMQ-stdver-conn-estab-get-guid-from-file**; file name: **MSMQ-Queue-Name.init**). As IQ GUIDs are reassigned whenever the IQs are recreated, this enhancement allows the e*Ways to pick up the new GUIDs without invoking the e*Way configuration tool. Note, however, that the logical IQ names no longer reflect the names displayed in the MSMQ Explorer, which provides potential for future confusion when updating the **MSMQ-Queue-Name.init** file.

The inbound MSMQ e*Way also utilizes the following modified functions:

MSMQ-stdver-data-exchg-get-id
Pre-pends the Event ID to the incoming Event.

MSMQ-stdver-neg-ack-errq
Puts an Event to the MSMQ error IQ when the incoming Event has failed to be committed to the e*Gate IQ.

MSMQ-stdver-pos-ack-pers
Removes the MSMQ journal file after successfully committing to the e*Gate IQ.

The outbound MSMQ e*Way for error messages utilizes the following modified function:

MSMQ-stdver-proc-outgoing-with-label
Puts the Event into the IQ using the prefixed error message as the Event label.

B.3 Overall Design Objectives

B.3.1 Performance

The Diamond schema employs a two-stage process flow aimed at achieving performance efficiencies. The first stage, handled by the inbound e*Way, involves identification of the incoming Event Type through parsing of the meta-data fields. The second stage, handled by the outbound e*Way, involves function-specific transformations based on the Event Type assigned in the preceding stage. This design provides the following efficiencies:

- The Event parsing load during the first stage is restricted to the meta-data portion of the structure through the use of generic ETDs.

- The allocation of each Event Type to a separate Collaboration in the second stage gives multi-threading of the transformation processes (each outbound Collaboration runs in a separate thread).

These performance efficiencies could be further improved by converting the inbound and outbound e*Ways to Pass-Through services and introducing a third stage to handle all parsing and transformations. The third stage would consist of a subscriber pool of BOBs, subscribing to Events from the inbound e*Way and publishing Events for the outbound e*Way. Each BOB in the subscriber pool would use the same Collaboration Rules, which would do the following:

- Extract the function name or transaction id using a string search or very restricted Event parse

- Use the extracted string to generate the name of a translation function

- Invoke the translation function to do a full Event parse and transformation (input = inbound Event string; output = transformed Event string)

- Publish the transformed Event as the appropriate Event Type

The translation functions would be the Collaboration Rules scripts that have already been created with the Collaboration Rules Editor for the outbound e*Ways.

These changes would provide the following efficiencies:

- Each Event restricted to a single parse during its lifetime

- Minimum loading on inbound and outbound e*Ways as the Pass-Through service has no parsing overhead

- Multi-threading of parsing and transformation processing through the use of a subscriber pool

- The opportunity to dynamically respond to variable Event loads through the use of Event Thresholds and Alert scripts to change the size of the subscriber pool

B.3.2 Error Handling

The Diamond schema makes good use of the exception-handling facility to trap and process error conditions. User-specified exceptions are currently defined in the file, **define-exceptions.monk**, for the following conditions:

amount-format
Thrown by currency format conversion function.

date-format
Thrown by date format conversion functions.

invalid-func-name
Thrown by get-Root-Node function (currently unused).

msgid-length
Thrown by MSMQ Event ID conversion function.

Every Collaboration Rules script utilizes a function-level try-catch statement to trap the above exceptions and additionally the system-defined exception-mapping condition (Event parsing failure). Trapped errors are handled by generating and publishing an error message (Event Types: **etDSF_PARSE_ERROR** and **etLOB_PARSE_ERROR**).

Note:

- *The date format conversion functions are currently throwing the **amount-format** exception. Change this to the **date-format** exception.*

- *The **invalid-func-name** exception could be utilized by the identification scripts.*

B.3.3 Component Failure and System Fail-Safe

The current design for component failure is based on the launch of BOBs from an Alert script for the purpose of rerouting Event loads between the two clone servers. If an outbound e*Way fails, a BOB on the local server is launched to publish incoming Events to the clone IQ on the remote server. If an inbound e*Way fails, a BOB on the remote server is launched to subscribe to Events coming from the clone inbound e*Way and publish them to the local IQ.

This design has the following problems:

- It assumes that the external system which is served by the failing e*Way can also reroute its Event load through the clone e*Way. Each e*Way connects with a unique IQ-manager/IQ, for example:

  - **ewLOB_IN_01** connects with **PAST01T0/EGATE.PAST01T0.FROMLOB**

  - **ewLOB_IN_02** connects with **PAST02T0/EGATE.PAST02T0.FROMLOB**

It is not clear whether the back-end LOB application could swap between these IQs when an e*Way fails:

- Rerouting Events through a clone e*Way will increase the Event load on the e*Way and degrade overall system performance.

- Parse error Events are not currently included in the design.

A more resilient design would be to have both sets of cloned e*Ways defined for each Participating Host with only one set active on each host under normal processing conditions. For example:

- **cbDSFGWST01** has **ewLOB_IN_01** which is auto-started and **ewLOB_IN_02_B** which requires a manual start

- **cbDSFGWST02** has **ewLOB_IN_02** which is auto-started and **ewLOB_IN_01_B** which requires a manual start

When **ewLOB_IN_01** fails on **DSFGWST01**, **ewLOB_IN_01_B** would be launched on **DSFGWST02** from an Alert script.

Effectively all components, e*Ways, BOBs, and IQ Managers, would be duplicated on each server. Only one set is auto-started—components in the other set are started from scripts when a failure condition is realized.

The current design makes no allowance for a system fail-safe; however, the previously recommended design is compatible with a fail-safe strategy. In the case of a complete server failure, all duplicate components on the clone server could be started to take over processing. This design would require an additional component of a heartbeat mechanism so that each server could detect the failure of the other clone server.

Finally, give some consideration to resilience of the Registry Host. From e*Gate release 4.1.0 onward, a facility for distributing and replicating the Registry is provided. This allows mirror copies of the master Registry to be created on multiple hosts with automatic propagation of changes from the primary Registry Host to secondary Registry Hosts. Participating Hosts can then load their configuration from any available Registry Host.

B.4 Environments and Source Control

Diamond's working procedures exhibit carefully planned division into multiple development and test (Model Office) environments. There are the following types of development environments:

- One using file-based e*Ways

- One using IQ-based e*Ways

This setup provides an effective mechanism for separately testing Event identification/ transformation/routing and the interactions with external systems.

There are well-documented procedures for the migration of development phases from the IQ-based development environment to the test environment. These procedures cover the export and import of the schema data and file repository, and the registration of Control Broker and Installer services. The particular procedures required for migrations between the file-based and IQ-based environments are not documented. Caution is necessary for this migration; transfer only a subset of the components. Do not include file or IQ-specific components.

All development and testing work is conducted under the Administrator user, with changes promoted to run time before testing. Use the Team Registry feature for future development work, especially when more than one developer is concurrently making changes. Each developer would use their own user Sandbox area for making and testing changes before promoting the changes into the run time (shared) area. When commencing work on an existing component, a developer would be advised if the component was currently locked for update by another developer.

Source control of development phases is currently based on duplication of schemas in the Model Office test environment. Whenever a phase is migrated to the Model Office, it is imported as a new schema and repository. Schema names contain number suffixes to indicate the progression of phases. This is a secure (although coarse-grained) method of controlling source changes; however, there is a requirement to document the steps required to downgrade a development environment to a previous phase saved in the Model Office.

B.5 Run-Time Management

Specific monitoring and alerting requirements have as yet to be defined and implemented. The following issues need to be considered:

- Which error conditions require alert notification

- Which notifications must be resolved automatically through the script channel and which require manual intervention

- Which notification channels are to be utilized (interactive monitors, email, pager, fax, and so on)

- What escalation levels are required

- What levels of logging or audit trail information are required

The Detailed Configuration Design document notes a possible requirement for reporting status and error notifications to an external system management tool (the DSF infrastructure). Handle this via the SNMP Agent notification channel, which is configured in the following places:

- During installation of the e*Gate SNMP Agent

- By adding an SNMP Agent component in the schema at the host level

- By modifying the SNMP Agent channel section in the Notification Routing script

Other run-time issues to be considered are:

- What disk utilization thresholds are required on each Participating Host

- What notification expiration and cleanup schedules are required

- What start/stop schedules and restart intervals are required for e*Ways, IQ Managers and BOBs

- What Event thresholds are required for e*Ways and BOBs

- What active and journaled Event Type expiration intervals are required

- What IQ cleanup schedules are required

Upgrading from DataGate to e*Gate

This appendix lists the major tasks you must keep in mind when upgrading from e*Gate release 3.6 (also called DataGate) to e*Gate 4.5.3.

In This Chapter

C.1 Upgrading from DataGate: Overview

This section provides a general summary of how to upgrade from e*Gate 3.6 to e*Gate 4.5.3.

C.1.1 e*Gate Compatibility Components

The e*Gate Compatibility Components automate the upgrade process from e*Gate 3.6 to e*Gate 4.5.3. This tool set consists of the following components:

- **Schema Configuration Utility (stcdgschema)** allows you to create an e*Gate 4.5.3 schema from e*Gate 3.6 Route Table files. This command also creates:

  - **Communication Client Proxy e*Way,** which enables both standard and custom e*Gate 3.6 e*Ways to exchange data with e*Gate 4.5.3 IQ components

  - **Route Table Collaboration (RTC) Service**, configured in the inbound Communication Client Proxy e*Way of the schema created by the **stcdgschema** command to transport and manipulate data in accordance with the e*Gate 3.6 Route Table

- **e*Way Configuration File Conversion Utility** uses the original e*Gate 3.6 e*Way configuration file to help generate an equivalent e*Gate 4.5.3 e*Way's configuration files. Use this tool when substituting e*Gate 3.6/Proxy e*Way combinations with the functional equivalent of e*Gate 4.5.3 e*Ways.

C.1.2 Basic Upgrade Procedure

The following steps provide a procedural overview of the upgrade operation:

1 Verify that you have completed all the necessary prerequisite procedures.

2   Run **stcdgschema** for each e*Gate 3.6 environment being upgraded and commit any new schemas to the e*Gate 4.5.3 Registry.

3   Confirm that the schema(s) and Proxy e*Ways were created correctly.

4   Test all Proxy e*Ways with data. This includes Proxy e*Ways that were created for e*Gate 3.6 e*Ways that will convert to e*Gate 4.5.3 e*Ways.

5   Convert any desired e*Gate 3.6 e*Way configurations to e*Gate 4.5.3 e*Ways using **dgw2ew.cmd** procedure.

6   Test the new e*Gate 4.5.3 e*Ways with data.

7   Move the test environment to production.

C.2   Upgrade Guide

SeeBeyond provides a document that explains complete details on how to perform this upgrade. For more information, see the *e*Gate Integrator Upgrade Guide*.

Installing e*Gate on Windows 2000 Clusters

This appendix explains how to implement Microsoft Windows 2000 network clustering software.

In This Chapter

- **"e*Gate with Microsoft Clustering" on page 243**
- **"Implementation Procedures" on page 244**

D.1 e*Gate with Microsoft Clustering

As discussed in **Chapter 8**, you can design an e*Gate environment to take advantage of a third-party clustering or high availability applications. The examples in that chapter show how this software can enhance IQ-related fail over. This appendix provides additional detail about designing a simple failover environment with Microsoft clustering software.

Note: *e*Gate has not been fully tested for UNIX high availability products.*

The following figure shows a sample e*Gate environment in a Windows 2000 cluster.

Figure 102   e*Gate Environment with Windows 2000 Clustering Software
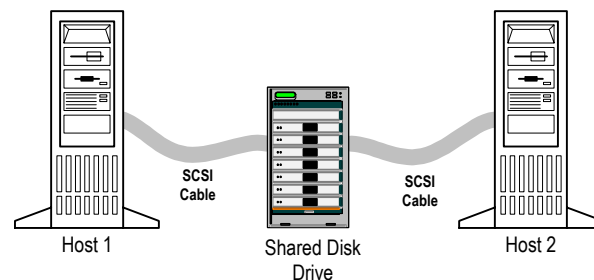
Host 1 and host 2 are sharing a single secure hard disk. Use the following general steps to set e*Gate up in this environment:

1   Set up the desired cluster of computers and verify that they are communicating and working properly.

2   Install e*Gate on host 2, but installing to the shared hard disk. Verify that the hard disk has a secure redundancy capability.

3 Install e*Gate on host 1 to the shared hard disk. Use the same directory locations.

4 Ensure that the OS registry entries are the same for both hosts.

5 Run e*Gate on host 1, using host 2 as a backup in case host 1 fails.

The rest of this appendix explains these steps in greater detail.

Note: *Microsoft has a certification program for hardware vendors that supply a "cluster system." When you go to a vendor to buy computers for a cluster, the vendor has to sell you two computers, set up in the specific way dictated by Microsoft. See the appropriate Microsoft documentation or Web page for details.*

D.2 Implementation Procedures

This section explains specific procedures necessary to implement Microsoft clustering.

D.2.1 General Considerations

Before implementing Microsoft clustering software in your e*Gate environment, verify that:

- The shared hard disk is used for installation, instead of local drives (see **Figure 102 on page 243**)

- You edit the Windows registry entries (the rest of this section gives details)

- When setting up the Participating Host inside the schema, you give it the cluster name used in the Microsoft software, instead of the individual computer name

- You set up the e*Gate services as resources for each cluster, using the Microsoft Cluster Administrator feature

- You begin the e*Gate installation on host 2 and use the Cluster Administrator to activate the default group the "Cluster Group" on host 2

D.2.2 Procedure

This section provides step-by-step instructions for making the e*Gate environment Microsoft clustering compliant

Note: *This procedure explains e*Gate installation and configuration with Microsoft clustering. For instructions on how to install and set up the Microsoft software, see the appropriate Microsoft documentation.*

To set up Microsoft Windows clustering

1 Begin the e*Gate installation on host 2 with installation disk No. 1 as shown in **Figure 102 on page 243**.

Use the same host names in the Microsoft software as you do in e*Gate. Using the same host naming convention in both systems avoids confusion later on. In this example, the naming convention is:

- Host 1 = WIN2KCLUSTER1
- Host 2 = WIN2KCLUSTER2

Note: *You must start with host 2 (WIN2KCLUSTER2) to proceed with the e*Gate installation on an active-passive cluster configuration because only one computer at a time has access to the shared SCSI drive. This installation order gives host 2 access to the shared hard disk. This hard disk is a resource in "Cluster Group."*

2   For the Registry Host, use the following options:

- No Registry Replication
- Destination folder: **S:\eGate\Server**

where drive **S:** stands for shared hard disk. This drive can have any letter designation, as long as it is the shared drive.

- Platforms: Windows NT/2000

3   For the Participating Host, use the following options:

- Destination folder: **S:\eGate\client**
- Host name: **WIN2KCLUSTER2** (for this example); the Control Broker name then becomes **win2kcluster2_cb**
- Do not specify secondary Registry Hosts
- Leave password path as default (**S:\eGate\client**)

Note: *When installing, enter the user name "Administrator" and password "STC."*

4   In the Windows Control Panel Services, set all e*Gate services (**stcregd, stcreginst, stccb**) to manual start instead of automatic (see **Figure 103 on page 246**).

For complete instructions on setting Windows Services properties, see the appropriate Microsoft Windows 2000 documentation.

Figure 103 Windows Control Panel Services
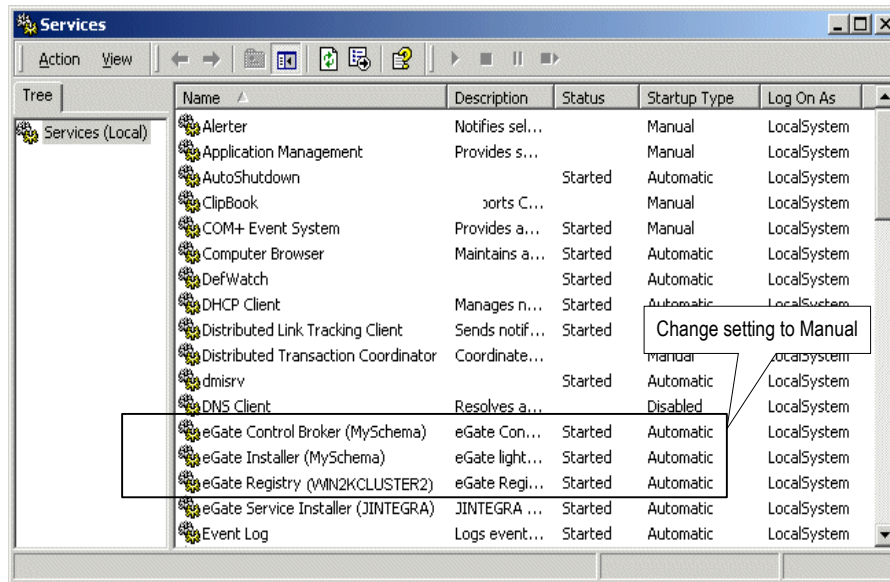
Under Windows 2000, stcregd, stcreginst, and stccb run as services

5 Reboot host 2.

Note: *These settings cause the default cluster group to fail over to host 1. Once host 2 is rebooted, you can use the Cluster Administrator to make "Cluster Group" active on host 2 again.*

6 Insert the e*Gate installation disk with the GUI.

Note: *To ensure the correct disk order, for example, which disk contains the GUI, see the* **e*Gate Integrator Installation Guide** *(Windows installation chapter).*

7 Choose to install e*Gate Enterprise Manager and e*Gate Monitor.

8 When prompted, answer "Yes" to install Exceed and use the destination **C:\EXCEED**.

  ◆ e*Gate GUI destination **S:\eGate\client**

  ◆ e*Gate Registry Host name: **WIN2KCLUSTER2** (for this example)

  ◆ Install the SeeBeyond Editors and use the destination **S:\eGate\client\bin** (the default)

9 Reboot host 2 again.

10 After this restart, when you log back on to host 2, the message "Preparing to install" appears in a dialog box with the title "SeeBeyond Editors." To continue, cancel this and any additional related dialog boxes.

Note: *These messages mean that host 2 has become passive and no longer has access to the S drive after restarting. Windows does not like this condition and tries to install the*

SeeBeyond Editors again. Canceling these dialog boxes and continuing does not interfere with your installation.

11 Use the Cluster Administrator to move "Cluster Group" to host 1 (named WIN2KCLUSTER1 in e*Gate and the Microsoft software).

12 Proceed with an identical e*Gate installation on host 1, following the previous steps in this procedure, with the following exceptions:

  ◆ In this example, use **WIN2KCLUSTER1** for the host name (**win2kcluster1_cb** for the Control Broker).

  ◆ During the Registry Host installation, answer "Yes" to the **.rdb** file-related questions.

13 Ensure that all **.egate.store** files point to the **S:** drive on the shared hard disk instead of to the C: drive.

Once you have installed e*Gate on both systems, modify the Windows registry, also on both systems.

To modify the Windows registry

1 Find the services key for "stcregd(cluster2)" on host 2 and "stcregd(cluster1)" on host 1, at the following location:

  **HKEY_LOCAL_COMPUTER\SYSTEM\CurrentControlSet\Services
  \stcregd(win2kcluster2)**

2 Change both key names to "stcregd(clusterX)." This action allows the Cluster Administrator to find the same service (now named identically) on both computers.

3 Change **–ln** parameter of the **ImagePath** value from "cluster1" to "WIN2KCLUSTER" on both computers. The point of giving the hosts similar names is to make the names simple and easy to change without confusion.

Next, you must make a few changes in the e*Gate Enterprise Manager GUI.

To modify the Enterprise Manager

1 Activate host 1 (WIN2KCLUSTER1 in this example), using the Cluster Administrator.

2 Rename the WIN2KCLUSTER1 Participating Host to WIN2KCLUSTER. Also rename the host 1 Control Broker to **WIN2KCLUSTER_cb** and the IQ Manager to **WIN2KCLUSTER_iqmgr**.

3 Change the host 1 Participating Host *network host name* to WIN2KCLUSTER (see **Figure 104 on page 248**).

Figure 104   e*Gate Enterprise Manager with Modifications
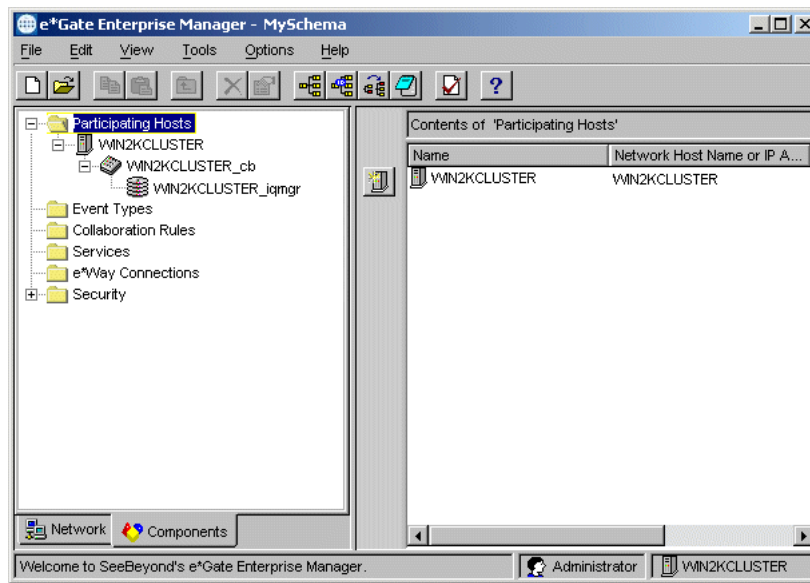
Next, run the following two commands at the command line.

To run the commands

1   On both computers, run the following command:

```
stcinstd -rh WIN2KCLUSTER -rs schema-name -un Administrator
    -up STC -ss
```

2   On both computers, run the following command:

```
stccb -rh WIN2KCLUSTER -rs schema-name -un Administrator -up STC
    -ln WIN2KCLUSTER_cb -sm
```

Finally, you need to make some additional changes in the Microsoft clustering software.

To modify the Cluster Administrator

1   Using the Cluster Administrator, add a new resource in the default group "Cluster Group" called "e*Gate Registry Service."

2   Specify all other resources in the group as dependencies.

3   Specify the name of the service for this group as "stcregd(ClusterX)".

4   Add the resource in the Cluster Administrator for the **WIN2KCLUSTER_cb** Control Broker.

5   Specify all other resources in the group as dependencies, including the e*Gate Registry resource.

Note: *Make sure that you set up the resources cluster IP address and cluster name as dependencies, so these resources always start **before** e*Gate.*

Glossary

access control list (ACL)
In e*Gate, a security list that determines user access to schemas, components, features, and operations in the system; in the e*Xchange Partner Manager, a list of information associated with a trading partner profile component (company, trading partner, outer envelope, or inner envelope) that specifies which users and user groups have permission to access the components and what specific access rights they have (add, edit, full control, or read).

action item
A task request that you can save in the e*Xchange Partner Manager database for subsequent retrieval by you or another user. You can track the status of action items and use them to create electronic reminder lists for yourself.

activity
An activity is an organizational unit for performing a specific function.

activity direction
Defines the Activity as Input, Output, or Input/Output.

Administrator
In e*Gate, a user with full access rights to the system; in e*Insight Business Process Manager, a user with full access rights who sets up users in the e*Xchange Administrator interface and has full rights within e*Insight. Administrators within e*Xchange Partner Manager are also able to set up users and perform application setup functions.

agent (Alert, SNMP)
A stand-alone application that monitors processes and resources and sends Notifications to e*Gate users, informing them of system status (for example, when a preset disk space level is exceeded).

application programming interface (API)
A set of protocols, routines, and tools for building software applications. The e*Xchange Partner Manager API consists of a set of Monk functions that can be called from custom validation Collaborations to interface with the database.

attribute
Attributes pass user-defined control information (programming arguments) to and from the e*Insight Business Process Manager and its activities.

B2B Protocol

The trading partner in e*Xchange profile component that you use to enter technical information about the exchange of messages between you and your trading partner. The type of eBusiness protocol you agree to use, such as ANSI X12, UN/EDIFACT, RosettaNet, or BizTalk, is an example of a B2B Protocol characteristic.

Business Object Broker (BOB)

A BOB component is similar to an e*Way in the sense that it establishes connectivity and is capable of data transformation. BOBs use Collaborations to route and transform data within the e*Gate environment. They have the following properties:

- They only communicate with IQs within e*Gate. They do not communicate with external applications as e*Ways do.

- They are optional by design. You can add them to an environment to remove some load from your e*Ways, either to set up easily maintainable data processing or to enable multiple internal processes.

business process instance (BPI)

A single instance of an executed Business Process Version. See **business process version**.

business process version

A form or variant of the original business process model.

Business Rules pane

Use the **Business Rules** pane in the Java Collaboration Rules Editor to navigate and edit the Java code of a Collaboration.

Business Rules Toolbar

Use the buttons on the Business Rules Toolbar in the Java Collaboration Rules Editor to add corresponding Java statements to a Collaboration.

byte length

Length in bytes of the string or regular expression to be matched within an ETD. e*Gate measures fixed-length data from byte 1.

byte offset

The beginning byte location of the string or regular expression to be verified within an ETD, beginning at byte 0.

child node

Node that is below a given node within the same branch of the ETD tree. Child nodes can inherit certain properties, such as delimiters, from their parent nodes.

code table

The mechanism used to customize values that appear in e*Xchange Partner Manager drop-down lists.

Collaboration

The component within an e*Way or BOB that performs data transformation and routing. It is the business logic that is applied to an Event in the course of delivery from a publisher to a subscriber. Collaboration components do the following functions: Subscriber components receive Events of a known type while publisher components distribute the transformed Events to a specified recipient. See also **Collaboration Rule**.

Collaboration Rules Editor

The GUI used to work with Collaboration Rules scripts in the Java and Monk programming languages. See also **Collaboration Rules script.**

Collaboration-ID Rules Editor

The GUI used to create Collaboration Rules scripts in the Monk programming language for *e*Gate Version 3.6 only*. See also **Collaboration Rules script.**

Collaboration Rule

The program logic that instructs a Collaboration how to execute the business logic required to support e*Gate's data transformation and routing. See also **Collaboration** and **Collaboration script**.

Collaboration Rules script

A Collaboration script (program) written using the Collaboration Rules Editor feature.

Collaboration script

The data flow and transformation logic contained in and configured by an e*Gate Collaboration and written as a program in any of the following programming languages: Monk, Java, or C.

Collaboration Service

Libraries that provide the low-level facilities by which Collaborations execute Collaboration Rules, for example, issuing system-specific terminate calls.

command line

A tool for monitoring and controlling e*Gate by entering application programming interface (API) commands at a DOS or DOS-type prompt.

committing files

Takes them out of the run-time schema and places them in the Sandbox. See also, **Sandbox** and **run time**.

Control Broker

An automatically generated e*Gate component that starts and monitors e*Ways and BOBs. At least one Control Broker must be running on each host within a schema.

command line

A tool for monitoring and controlling e*Gate by entering application program interface (API) commands at a DOS or DOS-type prompt.

Company

An organization with which you conduct electronic business (eBusiness). A company can consist of one or more trading partners. See also **Trading partner**.

delimiter

A special character assigned to mark the boundary of an Event node.

delimiter declaration field

In the HL7 standard, the location within an Event where a character is to be used as a delimiter. Also refers to the ETD node boundary it marks.

destination

Pertaining to the primary output ETD within a Collaboration Rules component or Collaboration Rules script.

e*Gate Monitor

A standard e*Gate component that provides graphical access to e*Gate environments and e*Gate status information, state control, and troubleshooting log files and journals.

e*Way Connection

An e*Way Connection is the encoding of the access information for one particular external connection or SeeBeyond JMS IQ Manager. In terms of content, it is similar to an e*Way configuration file, in defining enough information to be able to "login" or connect to the particular system. However, unlike e*Way configuration files, there is no schedule information. The idea is that the e*Way Connection will be information shared across multiple interfaces.

e*Way Editor

The GUI used to configure e*Ways.

e*Way Intelligent Adapter

A component that provides a noninvasive point of contact between an e*Gate environment and an external business application (often abbreviated as e*Way). e*Ways establish connectivity with applications, using whatever communication protocol is appropriate. e*Ways perform the following main functions: (1) receiving unprocessed data from external components, transforming it into Events, and forwarding it to other components within e*Gate via IQs; and (2) sending processed data to external components (can also include data transformation).

e*Insight Business Process Manager (e*Insight)

An application within the eBI Suite that facilitates the automation of the business process flow of eBusiness activities.

eBusiness protocol

An eBusiness protocol is a generally accepted standard for formatting and exchanging electronic messages between trading partners. ANSI X12, UN/EDIFACT, RosettaNet, and BizTalk are examples of eBusiness protocols.

e*Xchange Partner Manager (e*Xchange)
An application within the eBI Suite, that you use to set up and maintain trading partner profiles and view processed messages. e*Xchange also processes inbound and outbound messages according to certain eBusiness protocols and your validation Collaborations.

Error table
The mechanism used to define error messages that you can use with custom validation Collaborations.

eSecurity Manager (eSM)
An add-on to e*Xchange that secures transmission of business-to-business exchanges over public domains such as the Internet.

Enterprise Manager
The e*Gate GUI that allows you to create, configure, and modify all components of an e*Gate environment.

Event
A unit package of data processed by the e*Gate environment. This data has a defined structure, for example, a known number of fields with known characteristics and delimiters. Events are classified by Event Type and exchanged within e*Gate as ETDs.

Event, delimited
A variable-length Event made up of nodes whose boundaries are marked by delimiters.

Event, fixed
An Event of prescribed length. Each node within a fixed ETD is identified by its length and location within that ETD.

Event Linking and Sequencing (ELS)
Event Linking and Sequencing is a feature that allows for Events that arrive from independent input streams to be delivered to subscribers as related units. Complex Linking and Sequencing can be configured using the e*Gate 4.5 Java Collaboration Rules Editor, so that **n** different input streams can be linked and sequenced according to rules based on any combination of content or time-out rules.

Event, monitoring
An Event sent from one e*Gate component to another that describes an internal e*Gate condition, such as "component up" or "component down."

Event Type
A class of Events with common characteristics. An Event Type is also a logical name entry in e*Gate that points to a single ETD.

Event Type Definition (ETD)
A programmatic representation of an Event Type that Collaboration Rules can use when parsing, transforming, or routing data.

EDT Editor

The GUI used to configure ETDs in the Java and Monk programming languages; abbreviated as ETD Editor. See also **Event Type Definition.**

ETD node

A segment of an ETD that is represented graphically as a node in an ETD tree in the ETD Editor window, and represents a portion of an Event.

EDT tree

The graphical or logical representation of the ETD and its hierarchy.

extended attribute

Information you can store at the company, trading partner, B2B protocol, and Message Profile levels, as needed for your business. For companies and trading partners, you can create extended attributes to store specific information about the company or trading partner. For B2B protocol and Message Profile, the extended attributes are specific to a particular eBusiness protocol. Characteristics of ANSI X12 Interchange, Functional Group, and Transaction Set envelopes are examples of extended attributes you need to enter if you exchange X12 messages with a trading partner. Contrast with *General attributes*.

external system

A system that sends or receives data and is outside of the e*Gate environment.

general attributes

Basic information in e*Xchange that identifies companies and trading partners. For B2B protocol and Message Profile, this includes the information you enter for a trading partner profile that is necessary for the exchange of messages but is not specific to a particular eBusiness protocol. The direction of a transmission or the password needed to send messages to an FTP site are examples of general attributes. Contrast with **Extended attributes**.

Guaranteed Exactly Once Delivery (GEOD)

Using XA, GEOD guarantees once and only once delivery. Guaranteed Exactly Once Delivery refers to the usage of XA-compliant e*Gate and external components to ensure the delivery occurs once regardless of failures.

GUI

Graphical User Interface. A type of computer interface that enables the user to perform actions via the use of symbols, visual metaphors and pointing devices.

ignore

When a file from the run-time schema, which already carries an advisory lock, is checked out. The advisory lock stays with the original user who checked out the file, and does not transfer to the new user.

hash

Hashing is the transformation of a string of characters into a usually shorter, fixed-length value that represents the original string. The hash is a mathematical summary of the original message and is created by a hash function.

A cryptographically strong hash function has a number of requirements: It is easy to compute, one-way, and collision-free. This means that it is computationally infeasible to find a message that corresponds to a known hash, or to compose two messages whose hash values are the same.

The fixed-length hash value makes message authentication through the use of digital signatures possible, because only a small number of bytes must be used in a computationally expensive public key operation, rather than the entire message.

The most common cryptographic hash functions in use today are SHA-1 (the Secure Hash Algorithm Standard) and MD5 (Message Digest #5).

implementation guide

A document, published for a particular electronic message standard by an industry subcommittee, that describes the structure and content of a specific message type. You can use the Validation Rules Builder to convert electronic versions of ANSI X12 implementation guides to validation Collaborations used by e*Xchange.

instance

A specific node within a series of repeating nodes.

Intelligent Queue (IQ)

A standard e*Gate component that manages the exchange of information between components within the e*Gate environment, providing nonvolatile storage for data as it passes from one component to another.

IQ Manager

A standard e*Gate component that reorganizes IQs, archives queue information upon request to save disk space, and locks the queues when maintenance is performed.

IQ Service

A utility that provides the transport of components within IQs, handling the low-level implementation of data exchange, such as system calls to initialize or reorganize a database.

Java Message Service (JMS)

See **SeeBeyond JMS** for the e*Gate implementation of JMS.

log file

A text file that contains a record of all actions taken by an e*Way. Use log files to troubleshoot any problems in the system and discover how to solve them.

message log

A record of inbound and outbound electronic transactions processed by the e*Xchange Partner Manager. This is implemented as the message tracking facility e*Xchange.

Message Protocol
An Message Protocol definition is a set of parameters and other information you enter about each electronic inner envelope you process with e*Xchange Partner Manager. This definition associates the validation Collaborations that are needed to validate each kind of message.

The version number of the eBusiness protocol that applies to the message and whether the message will be transmitted interactively or in batch are examples of Message Protocol characteristics.

message tracking attribute
An attribute you can define to identify messages stored in the e*Xchange Partner Manager database. Special message tracking extended attributes can be set up and associated with a specific message type (protocol, version, and direction). Examples of attributes that are set up at the message tracking attribute level are Process Instance ID and Activity Instance ID for RosettaNet and FG and TS control numbers for X12.

Monitor
An executable e*Gate component that enables users to view messages that describe the state of e*Gate internal components. Interactive monitors also enable users to send commands to e*Gate components; non-interactive monitors only enable users to view notifications.

monitoring Event
An Event, sent by one e*Gate component to another (usually to the Control Broker) that describes occurrences within the e*Gate environment. Monitoring Events include error messages, such as "component down" or "component lost;" status messages such as "component up" or "contact re-established;" system performance messages, such as "Event processing below preset threshold" or "disk space low;" and miscellaneous messages such as scheduled timers, configuration changes, or "Event content of interest."

Monk
SeeBeyond's Event-processing language.

Monk Test Console
A GUI test feature for testing Monk functions and Collaboration scripts before introducing them into the run-time environment.

Navigator Tree
The tree-like graphical display in the Navigator/Components pane of the Enterprise Manager window. This display shows the components of the e*Gate environment and how they relate to each other in pictorial form using an icon to represent each component.

node
See **ETD node**.

node set
A group of associated nodes that are order-independent, or that repeat.

non-repudiation
   The inability of a sender to refute a message—that is, to claim at a later date that the sender was not the originator of the message. This is implemented through the use of a digital signature attached to the message. The signature can be used by the recipient to prove that the sender positively wrote the message, and that its contents were not tampered with after it was signed.

   The sender of a message can also obtain irrefutable proof of receipt of the original message. Non-repudiation of receipt is implemented using an acknowledgment to the sender. This acknowledgment contains the digital signature of the message, and is also digitally signed by the receiver of the original message.

notification
   A notification sent to the user by the e*Gate environment.

notification routing
   The Collaboration Rules script that specifies how monitoring Events are translated into notifications.

operator
   An operator controls the logical flow of data-based decisions in the business process model. An operator outputs specific information when specified input conditions are met.

parent node
   Node that is above a given node within the same branch of the ETD tree.

Participating Host
   A client computer that supports an e*Gate environment, as opposed to the Registry Host, which acts as a server to the Participating Host. See also, **Registry Host**.

Partner Manager Envelope Profile
   A partner manager envelope profile is a set of default extended attribute values that you define for a trading partner profile component (company, trading partner, outer envelope, or inner envelope).

PKCS
   An acronym for Public-Key Cryptography System. PKCS is a set of informal intervendor standard protocols developed by RSA Security, the licensers of the RSA public key cryptosystem, for making secure information exchange on the Internet possible. The standards include RSA encryption, password-based encryption, extended certificate syntax, and cryptographic message syntax for S/MIME, RSA's proposed standard for secure email.

PKI
   A PKI (public key infrastructure) enables users of a basically unsecured public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority.

It is a networked system of certification authorities (CAs), registration authorities (RAs), certificate management systems (CMSs), and X.500 directories (specialized distributed databases). It enables two parties unknown to each other to exchange sensitive information and money over an unsecured network.

promoting files

Update the run-time schema to use the new file or files. If the file already exists in the run-time schema, that file is replaced with the file from the Sandbox. Promoting a file automatically removes it from the user's Sandbox and, if the user has locked the file, releases the lock. When you delete a file from the Sandbox without promoting it to the run-time schema, you *remove* the file. If the file was locked, the lock is released.

public key encryption

Encryption using PKCS. See **PKCS**.

publish

See **publish/subscribe**.

publish/subscribe

Abbreviated as pub/sub; subscriber components retrieve Events. Publisher components make Events available to other e*Gate components. See also **Collaboration**.

Registry

The storage place (in a directory) for all e*Gate configuration details, including file containment.

Registry Host

A computer that is running the e*Gate Registry daemon/service (**stcregd.exe**) and acts as the e*Gate environment server; a system that provides Registry services to other systems running e*Gate applications. See also, **Participating Host**.

Registry Service

The service that handles all requests for updates to the e*Gate registry and forwards updated files to Participating Hosts (clients) as necessary.

regular expression

A pattern representing a set of strings to be matched.

removing files

Delete a file from the Sandbox without promoting it to the run-time schema. If the user carried the advisory lock for the file, the lock is released.

Report List

A list of reports that can be generated by an e*Xchange Partner Manager user.

run time
The environment in the Registry shared by all users of that Registry. The run time contains parameters that run for each instance of e*Gate unless the controlling user has a parameter in his or her own Sandbox, in which case the Sandbox is overridden. The run time is the production environment of a schema. See also, **Team Registry**.

Sandbox
A user's local development area. Each user has his own Sandbox. Files in a user's Sandbox are available for testing the functions in the file themselves, but they are unavailable to the run-time schema. A user can test some parameters in the Sandbox while insulating other users from these changes. See also, **Team Registry**.

schema
Includes files and associated stores created by e*Gate, which contain the parameters of all the components that control, route, and transform data as it moves through e*Gate in a predefined system configuration.

SeeBeyond JMS
e*Gate implementation of the Java Message Service (JMS) using IQ Managers, IQs, and a special e*Way Connection.

sibling node
Node that is a child of the same parent node.

SEF
See **Standard Exchange Format (SEF)**.

signature key
The key used to encode a message signature. The signature key might be the same as the encryption key; but when two different keys are used for different purposes, this is known as a dual-key system. See also **key**.

S/MIME
An acronym for Secure/Multipurpose Internet Mail Extensions; it is an Internet email security standard that makes use of public key encryption.

source
Pertaining to the primary input Event or ETD within a Collaboration or Collaboration script.

.ssc file
See Event Type Definition (ETD).

Standard Exchange Format (SEF)
The Standard Exchange Format (SEF) is a flat file representation of an EDI implementation guideline. It is a standard that defines how data segments and data elements are structured so that the message can be understood between trading partners. It also includes validation rules, for example what are the valid values for a data element, or conditions such as if Field A is present then Field B is required.

The purpose of SEF is to put the EDI implementation guidelines in a file in computer readable format so that translators can directly import the file and use the implementation guidelines to translate or map the EDI file. The file can also be used as a means to exchange the implementation guidelines between trading partners, and can be posted on a public bulletin board or on the company's Web site in the Internet to convey to the public the implementation guidelines used by the company.

The SEF format was developed by Foresight Corporation and is now in the public domain. Programs that can directly import SEF files can save users considerable time in developing new translations or maps.

subnode
A node that is connected through parent-child relationships to another node that is higher in the ETD tree.

subnode set
A set of order-independent or repeating ETD nodes one level below the currently selected node in the ETD tree.

subscribe
See **publish/subscribe**.

Team Registry
Allows multiple users to develop components of a single schema simultaneously by compartmentalizing the e*Gate Registry into work-in-progress and run-time environment areas, implemented by the Sandbox and run-time environments.

Trading Partner component
The trading partner profile component that you use to enter business information about your trading partner. The name of the trading partner, which could be a subdivision of a company, and the people you want to contact are examples of information you enter for a trading partner component.

transaction set
In X12, each business grouping of data is called a transaction set. For example, a group of benefit enrollments sent from a sponsor to a payer is considered a transaction set. Each transaction set contains groups of logically related data in units called segments. For example, the N4 segment conveys the city, state, ZIP code, and other geographic information.

A transaction set contains multiple segments, so the addresses of the different parties, for example, can be conveyed from one computer to the other. An analogy would be that the transaction set is like a freight train; the segments are like the train's cars, and each segment can contain several data elements in the same way that a train car can hold multiple crates.

Specifically, in X12, the transaction set is comprised of segments ST through SE.

.tsc file
A Collaboration Rules file. A **.tsc** file is a Monk translation sub-file.

user group

User groups allow you to grant access permissions to a set of users with similar processing needs without having to specify individual privileges for each user. For example, the User Administrator can set up a group for users who need full access to a specific trading partner profile, but who must not be able to view information about any other profile.

The User Administrator assigns each user that meets this criterion to a particular user group. Then, your e*Xchange Administrator (or another user who has been granted appropriate privileges) grants access privileges to this user group so that all members of the group can view and modify the desired information.

validation Collaboration

A Collaboration that you create to define the syntax and validate the content of electronic business-to-business (B2B) messages. One validation Collaboration is required for each type of electronic message to be processed by e*Xchange. You can use the Validation Rules Builder to automatically generate a validation Collaboration for a specific kind of X12 transaction, according to specific implementation guidelines.

Validation Rules Builder

An e*Xchange Partner Manager tool for converting electronic EDI implementation guides into files that are compatible for use with e*Xchange. This conversion tool accepts Standard Exchange Format (SEF) version 1.4 or 1.5 files and converts then into e*Gate ETD and Collaboration Rules files.

value added network (VAN)

A private network provider that offers secure electronic data interchange (EDI) services to companies. VANs often offer EDI translation, encryption, secure email, management reporting, and other extra services for their customers.

XML

Extensible Markup Language. RosettaNet PIPs are written in XML. XML is different from String in that XML messages can contain both content and information about the content.

Index

A

acceptance testing **170**
access control **28**
acknowledgments
   negative **30**
   positive **30**
Alert Agent **26, 27, 170**
Alerts **27, 206, 207**
analysis and planning overview **38**
analysis of needs cycle **46**
analysis of requirements categories **41**
Analysis of Requirements Phase definition **38**
ANSI X12 **35**
Application Connectivity layer **30**
application program interface (API) **27, 28, 29, 30, 31**
application programming interface (API) **27**
Approved Proposal **51, 52**
architecture, e*Gate **23**
automate the conversion process **241**
automatic generation of Event Type Definitions **27**
avoidance of data duplication **30**

B

basing memory estimates **91**
batching Events **110**
best practices **18, 59, 73, 201**
BOB operation **89**
buffer size, e*Ways **31**
business logic **27, 29, 31**
Business Object Brokers (BOBs) **29, 32**
   improved performance **205**
business process **32**
business process management **32**
business processes
   exporting **173**
   importing **177**
Business Rules and Data Processing layer **29**

C

C Collaboration Service **207**
C Language
   executing **207**

case studies **190–200**
change management **53, 58, 162**
charting eBI Suite flow **203**
CIDX **35**
clustering **84, 243–248**
   See Maximum Availability
Collaboration Editors **120**
Collaboration Rules **29, 31, 234**
Collaboration Rules Editors, SeeBeyond **26**
Collaboration-ID Rules Editor **120**
Collaborations **29, 234**
Compatibility Components **241**
component failure **238**
components
   BOB **86**
   distribute across multiple hosts **205**
   e*Way **86**
   IQ Managers **86**
Components Editor **26**
configuration files, testing e*Way **167**
connectivity, e*Ways **30**
Control Broker **28**
Control layer **28**
conventions, writing in document **18**
CPU
   estimating requirements **61**
custom e*Ways **31**
cXML message format **27**

D

data duplication, avoidance of **30**
data processing logic **27**
data urgency considerations **99**
database access e*Ways **31**
databases **29, 31**
   IBM MQSeries **30**
   Oracle **30**
   Sybase **30**
DB2 UDB **35**
deployment
   estimating CPU requirements **61**
   gathering information **39**
   overview of phases **39**
   planning steps **51**
Deployment Checklist **52, 58**
Deployment Planning Phase definition **38**
Deployment Project Plan **53, 58, 59**
Deployment Project Team **45, 52, 53, 58, 59**
deployment road map
   Analysis of Requirements Phase **202**
   Deployment Planning Phase **202**
   Post-transition Maintenance Phase **203**
   Pre-transition Testing Phase **203**