e*Index Global Identifier Product Suite

e*Index™ Global Identifier Implementation Guide

Release 5.0.5 for Schema Run-time Environment (SRE)

SEEBEYOND

Version 20050429012631

Table of Contents

Chapter 1

Introduction

About this Chapter

Overview

This chapter provides an overview of this guide and the conventions used throughout. It also includes a list of related publications that provide information you may find useful during the implementation.

The following diagram illustrates the contents of each major topic in this chapter.

Introduction	Learn about the purpose of this guide, the intended reader, and how to use the guide
About this Guide	Learn about the layout of this guide and the conventions used throughout the guide
Additional Resources	Learn about additional SeeBeyond publications that provide related information

Introduction

Welcome

This guide provides comprehensive information on implementing e*Index™ Global Identifier (e*Index™). It discusses the architecture and core components of e*Index, the standard schema provided with e*Index, implementation requirements, and customizing the database processing code tables and system parameters. This guide also provides information about analyzing and planning the implementation, creating a project plan, designing and developing components, converting legacy data, testing, training, and moving into production. This chapter of the document provides background information you should know before implementing e*Index.

Whether you are a new or established user, you should review this guide before you begin the implementation. This chapter provides information about standards used in this guide. Chapter 2 provides an overview of e*Index and the e*Gate™ Integrator, and chapter 3 summarizes the implementation process for each component of e*Index. The actual implementation process is described in chapters 4 through 8.

Document Purpose and Scope

This guide explains how to successfully implement the SeeBeyond Technology Corporation™ (SeeBeyond™) e*Index application. The implementation of e*Index includes analysis and planning, system design and development, data conversion, interface implementation, training, testing, and production activities. A summary diagram of implementation activities is provided in chapter 3 of this guide.

This guide does not include information or instructions on using any of the e*Index applications. These topics are covered in the appropriate user's guide. See "Additional Resources" on page 1-8 for more information. These publications are referenced throughout this guide along with specific books from the e*Gate publications suite.

Intended Reader

The reader of this guide is presumed to be a customer who will be implementing an e*Index system. Since many of the implementation activities involve setting e*Gate components for e*Index, it is helpful to be familiar with the e*Gate environment, e*Gate schemas, and the database e*Ways. Knowledge of database administration, specifically of the database platform for the e*Index database, is also helpful. A thorough understanding of e*Index is not necessary to understand this guide.

The intended reader must have a good working knowledge of his or her company's current business processes and information system (IS) setup.

Using this Guide

For best results, skim through the guide to familiarize yourself with the locations of essential information you need. Each chapter begins with a simple graphic that identifies the information contained in the chapter. The beginning of each chapter provides introductory information on the topics covered in that chapter. This introductory material contains background and explanatory information you may need to understand before moving into the more detailed information later in the chapter.

This guide is intended to be used in conjunction with the *e*Xchange eBusiness Integration Suite Deployment Guide*, especially for an e*Index implementation that requires a new e*Gate environment be implemented as well.

About this Guide

Overview

This section provides information about the organization of each chapter and the conventions used throughout this document.

Document Organization

This guide is divided into eight chapters and five appendixes that cover the topics shown below.

Chapter	Topics
Chapter 1, Introduction	■ Introduction ■ About this Guide ■ Additional Resources
Chapter 2, Overview of e*Index	■ Introduction to e*Index ■ e*Index Components and Architecture ■ About the e*Gate Integrator
Chapter 3, Implementation Overview	■ Learning About the Implementation ■ GUI Implementation Overview ■ Schema Implementation Overview ■ Database Implementation Overview ■ Report Implementation Overview ■ Vality INTEGRITY Implementation Overview ■ Candidate Selection Query Overview ■ Initial Load Implementation Overview ■ Implementation Summary
Chapter 4, Analyzing Requirements	■ Gathering Information ■ Analyzing your Requirements ■ Identifying the Project Team ■ Determining Hardware Requirements ■ Determining Software Requirements
Chapter 5, Planning the Implementation	■ About the Planning Phase ■ Defining Overall Objectives ■ Identifying and Scheduling Tasks ■ Meeting Planning Phase Objectives

Chapter	Topics
Chapter 6, System Design and Development	■ About System Design and Development ■ Creating System Design Documents ■ Designing the Quality Workstation ■ Designing the Database ■ Designing e*Gate Components ■ About e*Index Schema Components ■ Developing your e*Index System
Chapter 7, Testing and Training	■ About Application Testing ■ Unit Testing ■ Integration Testing ■ Acceptance Testing ■ Training
Chapter 8, Moving to Production	■ Go-Live Methodology ■ Going Live with e*Index ■ Monitoring the System
Appendixes	■ A: Sample Data Mapping Table ■ B: Initial Load Processing Log ■ C: Sample Implementation Project Plan ■ D: Implementation Survey Questionnaire ■ E: Sample Hardware Worksheets

Conventions Used in this Guide

Before you start using this guide, it is important to understand the icon, special notation, and mouse conventions used.

Typographic Conventions

The following typographic conventions are used in this and other e*Index publications.

Item	Convention	Example
Book titles	Title caps, italic	See the *e*Index Administrator User's Guide*
Button names, key names, and key combinations	Bold	**OK** button **F1** key **Alt+Shift+V** key combination

Item	Convention	Example
Chapter titles (and section titles within chapters)	Title caps, in quotation marks	See Chapter 3, "Implementation Overview" See "Determining Hardware Requirements" later in this chapter
Menu names and commands	Bold, capitalization is identical to the interface	**Functions** menu **Action** menu **Save** command
New terms	Italic	The *Quality Workstation* is the machine on which the e*Index GUIs reside.
Window, page, and dialog titles	Title cap	Detail Inquiry window Print dialog

Icon and Special Notation Conventions

The following conventions are used in this and other e*Index publications to identify special types of information.

Icon or Notation	Type of information
Note	Supplemental information that is helpful to know, but not essential to completing a particular task.
Tip	Information that helps you to apply techniques and procedures described in the text to your specific needs. May also suggest alternative methods.
Important!	Information that is essential to the completion of a task.
Caution!	Advises you to take specific action to avoid loss of data.
▶	Indicates the beginning of a step-by-step instruction.
✓	Specifies a task to perform before you begin a step-by-step instruction.
ℹ	Indicates a cross-reference to other sections of the guide or to other publications.

Mouse Conventions

You can use either a single-button mouse or a multiple-button mouse with e*Index. If you use a multiple-button mouse, the left mouse button is the primary button, unless the mouse is configured differently.

The instructions in this guide may require you to use the mouse in a variety of ways:

- **Point** means to position the mouse pointer until the tip of the pointer rests on whatever you want to point to on the screen.
- **Click** means to press and then immediately release the left mouse button without moving the mouse.
- **Double-click** means to click the left mouse button twice in rapid succession.
- **Right-click** means to click the right mouse button once.
- **Drag** means to point and then hold down the mouse button as you move the mouse. **Drop** means to let go of the mouse button to place the dragged information where you want it to be moved.
- **Move** means to point to an object on the screen, such as an e*Index Security user group, and then drag the mouse to move the object to a screen location of your choice.
- **Highlight** means to select an area of text by dragging the mouse over the desired portion of text that appears on a window.
- **Select** means to point to a list of information on an e*Index window, and then click once to choose the data you want. The information becomes highlighted when selected.
- **Expand** means to double-click a row of information on an expandable list to display more details. The details appear on another row, below the row you double-click.
- **Collapse** means to double-click a row of information on an expandable list to hide the details that appear on the following row.

Additional Resources

SeeBeyond has developed a suite of e*Index user's guides and related publications that are distributed in an electronic library. In addition to the e*Index guides listed below, the *e*Xchange e*Business Integration Suite Deployment Guide* also contains useful information about the e*Gate Integrator, and will be referred to throughout this guide.

- *e*Index Global Identifier User's Guide*
  Helps e*Index quality workstation users to perform database maintenance tasks, such as merging and unmerging records, finding and resolving potential duplicates, adding and updating records, and viewing the audit trail.

- *e*Index Administrator User's Guide*
  Helps system administrators configure e*Index GUI appearance, processing logic, and matching algorithm to meet your business requirements. This guide also describes how to maintain the information in the database that is used to populate the drop-down lists in the e*Index.

- *e*Index Security User's Guide*
  Helps system administrators add users and user groups to e*Index applications, grant security permissions to users and user groups, maintain user and user group information, set up Event notifications, and configure password parameters.

- *e*Index Global Identifier Technical Reference*
  Describes message processing for e*Index, as well as the e*Index schema, database tables and e*Index Monk APIs. This guide also provides a complete listing of e*Index Monk APIs and functions, along with a description, parameters, syntax, return values, and examples for each.

- *e*Index Initial Load User's Guide*
  Provides the background information and instructions that system and database administrators need in order to load legacy data into the e*Index database, including a description of the expected data format and the schema files included with the load program.

- *Working with Reports for e*Index Global Identifier*
  Provides background information about the GUI and standard reports provided with e*Index, and explains how to modify and run the standard reports (for an Oracle installation only).

- *e*Index Global Identifier Installation Guide*
  Helps system and database administrators install a new e*Index environment for the current release, including e*Index schema files, the e*Index GUI, Java API for e*Index Active Integration, and database installation.

- *e*Index Global Identifier Upgrade Guide*
  Helps system and database administrators upgrade an existing e*Index environment to the most current release, including e*Index schema files, the e*Index GUI, the Java API for e*Index Active Integration, and database upgrades.
- *Java Programmer's Guide for e*Index Active Integration*
  Provides background and implementation information about the Java API for e*Index Active Integration. This guide also provides a complete listing of e*Index Java functions, along with a description, parameters, syntax, return values, and examples for each.

Chapter 2

Overview of e*Index

About this Chapter

Overview

This chapter provides a general overview of the e*Index application, including the basic architecture, components, and functions. The e*Index schema is installed into an e*Gate environment, which is described briefly in this chapter. For more information about the architecture and components of the e*Gate Integrator system, see "e*Gate Integrator System" in chapter 2 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

The following diagram illustrates the contents of each major topic in this chapter.

Introduction to e*Index	Learn about the functions and features of the e*Index Global Identifier
Architecture and Components	Learn about the components that make up the e*Index system, and how they work together
The e*Gate Integrator	Learn general information about the e*Gate Integrator and where to find more information

Introduction to e*Index

Overview

This section of the chapter provides background information about the functions and features of e*Index.

About e*Index

In today's business environment, person data is stored in many disparate systems throughout an organization. Each of these systems typically assigns its own, independent local identifiers, making it difficult to share information and create a single, reliable view of a member across the organization. e*Index solves this problem by providing a master person index in a relational database to share information between all of the disparate systems that maintain member data within an organization.

e*Index assigns its own unique global identification number (or UID), which is used to cross-reference all of a member's local identifiers within each connected system. In addition, e*Index incorporates matching algorithm logic to help match member profiles based on specific demographic information. Using this matching logic, e*Index can help to identify potentially duplicate profiles in real-time and can automatically join matching member profiles, providing continuous data cleansing as records are processed.

e*Index shares information with existing systems via the e*Gate Integrator. e*Index also receives information from the Quality Workstation (the e*Index GUIs). The information received through e*Gate includes the member information entered into the legacy systems that are connected to the e*Index system. This information is fed into the e*Index database through the e*Ways that you configure for e*Index. Using the Quality Workstation, you can maintain and monitor the data in the database, add or update member profiles, identify potential duplicate profiles, and merge and unmerge member profiles.

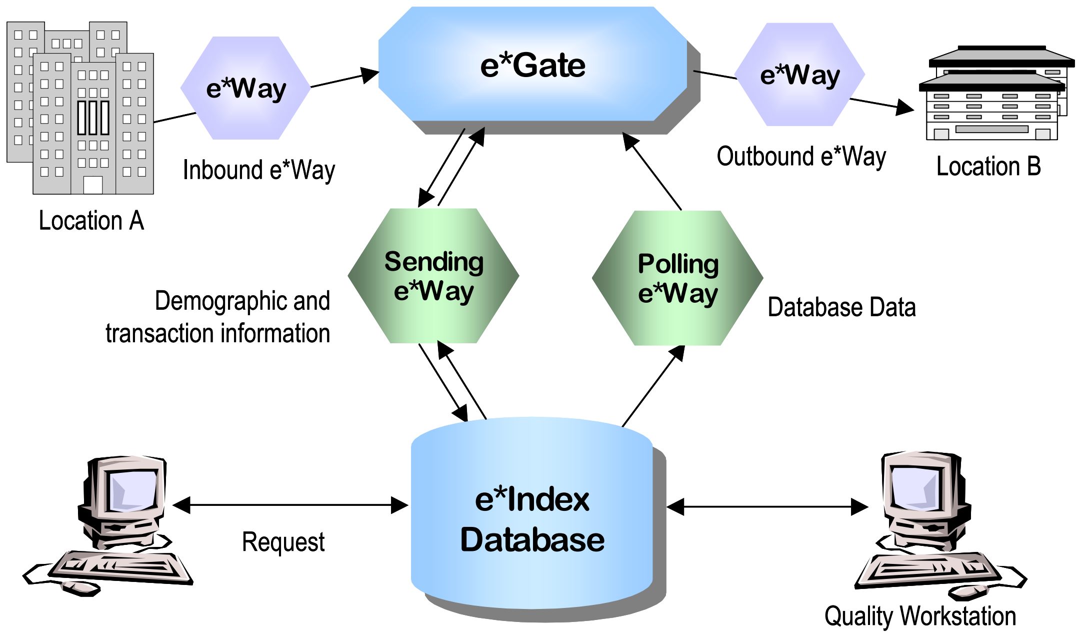
Optionally, e*Index can share information with existing systems via the Java API for e*Index Active Integration. This API set allows you to create custom web-based applications that can be used to search for, add, and update member records in the e*Index database.

e*Index Processing Flow

The data contained in the e*Index database originates in one of two places: the external systems that collect person data and in the Quality Workstation.

Data can also originate in custom applications using the active integration API.

When data is entered into an external system, the information is sent through an e*Way for that system to the e*Gate Integrator. e*Gate then routes the information through the e*Index sending e*Ways, which forward the information to the e*Index database. Matching and identification are performed against the incoming message, and either a new message is inserted or an existing record is updated. e*Index then sends the message, with a UID attached, back to e*Gate through the sending e*Ways. The information is now available to all the connected systems that are configured to accept it. Additionally, e*Index can send information to e*Gate through the e*Index polling e*Way, which reads messages from the *ui_msg_detail* table. This table, known as the *out queue*, typically contains information generated through the e*Index GUI.



e*Index Functions

e*Index was created to uniquely identify, match, and maintain member information throughout your business enterprise. e*Index achieves this by providing the following functions and features.

- **Unique Identifier**
  e*Index assigns a unique, enterprise-wide identifier to each member added to the database. This identifier is known as the unique global identifier, or UID. e*Index uses the UID to cross-reference a member's local IDs throughout the system.

- **Matching Algorithm**
  e*Index uses a sophisticated, real-time matching algorithm to automatically build the cross-index and uniquely identify member profiles. The configurable matching logic helps identify potentially duplicate profiles, identify matching profiles that can be automatically merged, and provide a matching weight to profiles returned from a search.

- **Continuous Cleansing**
  As records are processed through e*Index, the system continually checks for potential duplicate profiles and for exact match profiles. e*Index automatically merges profiles that are found to be exact matches, saving you the trouble of reviewing potential duplicates with a matching probability weight above a threshold you specify. e*Index maintains a record of each automatic merge.

- **Audit Trail**
  The system provides full audit capabilities. The history table records all changes to a member's demographic data. This allows e*Index to generate an audit trail that compares the demographic information before and after each modification.

- **Search**
  You can look up demographic information for a person record using various search criteria. Searches can be performed against the database for a specific member or a set of members. Each record returned as a possible match is assigned a matching probability weight, which indicates how closely each record matches the search criteria you specified.

- **Data Maintenance**
  e*Index provides the ability to add, update, deactivate, and delete data within the database tables. Data updates from external systems can occur in real-time or as batch processes. Updates can also be made directly through the e*Index GUIs.

- **Potential Duplicates**
  Using algorithm matching logic, e*Index has the ability to identify potential duplicate records, and provides the functionality to correct the duplication. A new record is considered a potential duplicate of an existing record when the matching probability of the two records falls within a range that you specify (for more information, see "DUPTHRES", "MATCHTHRES", and "THRESHOLD" in chapter 4 of the *e*Index Administrator User's Guide*). Potential duplicate records are resolved by either merging the records in question or removing their potential duplicate flags.

- **Merge Demographic Records**
  You can merge member records if you find they are actual duplicates of one another. You can specify either the local IDs or the UIDs of the records to be merged. When you merge two records, the record that is

not kept loses its Active status and receives a status of Merged. The information from the old record is retained in the database, providing the ability to unmerge the two records if necessary.

- **User Audit Log**
  e*Index maintains a complete record of all instances in which the *ui_person* table is accessed, including information about the user who accessed *ui_person*, the transaction performed, and the UIDs of the records accessed during the transaction. This function tracks access to member information and helps to maintain member privacy.

- **Security**
  e*Index protects information by limiting access to the system through user name and password login. Access can further be restricted to the function and action level by assigning specific permissions to users and user groups. Access can also be restricted by regions that you define. e*Index also provides the ability to notify certain users when selected transactions occur against the e*Index database.

- **Active Implementation**
  e*Index can be implemented in both passive and active modes. In passive mode, e*Index does not directly interact with the systems that provide the data for the e*Index database, and all matching, identification, and cross-referencing is performed behind the scenes. Active mode is implemented using the *Java API for e*Index Active Integration* to create an application that provides access to both customized web-based applications and the e*Index system. For more information about the Java API, see the *Java Programmer's Guide for e*Index Active Integration*.

About the Matching Algorithm

e*Index uses the Vality® INTEGRITY™ algorithm for probabilistic matching of person records in disparate systems. As records are processed through e*Index, the matching logic, along with configurable matching thresholds, identifies records that potentially or definitely represent the same person. The matching algorithm compares two person records and determines a matching weight for each match key field based on the reliability of each field and certain customizable probability factors. The sum of the weights of the match key fields is the total matching weight between the two records. Processing logic for the matching algorithm is highly configurable, and you can customize this logic by modifying the rule set files provided with e*Index. For more information about working with rule set files and the default matching logic, see chapter 6, "Configuring the Matching Algorithm", of the *e*Index Administrator User's Guide*.

e*Index Components and Architecture

Overview

This section of the chapter provides an overview of the components that comprise the e*Index system and the overall architecture of e*Index within the e*Gate environment.

e*Index GUI Applications

e*Index includes three GUI applications that work together to maintain person data. Each application has a separate graphical user interface (GUI), but each shares the same e*Index database. The three applications are e*Index Global Identifier, e*Index Administrator, and e*Index Security. In combination, the three GUIs allow you to monitor and maintain member data, configure system parameters, create and modify code table values, and configure security for the e*Index system.

- **e*Index Global Identifier**
  This GUI allows you to monitor and maintain member data, including searching for member profiles, adding new member profiles, updating or deactivating existing member profiles, and merging or unmerging member profiles. You can also view potential duplicate listings, resolve or merge potential duplicates, compare member profiles, and view a complete audit trail of each member profile.

- **e*Index Administrator**
  Using this GUI, you can configure certain system parameters, customize the e*Index environment, and create and maintain code table values. System parameters include setting matching thresholds, setting search limits and requirements, specifying a country format, and so on. Other customizations include modifying field, tab, and search type names on the e*Index GUI, specifying which fields to display, specifying which fields are required, modifying the matching logic, and modifying the search criteria for phonetic and candidate selection queries.

  The values you define for the code tables populate the drop-down lists on the e*Index Global Identifier GUI windows. The codes you associate with those descriptions identify the codes in the Events coming into the database from external systems. For example, if the existing systems use a code of "FR" to specify the French language, then the code for French in e*Index can be changed from "FRN" to "FR" to avoid additional code mapping in the schemas for the e*Index system. When a profile for a member whose first language is French appears on an e*Index window, e*Index checks the *ui_language* code table for the code "FR", and then displays the associated description "French" in the Language field.

- **e*Index Security**
  This GUI provides the ability to set up and maintain security for the e*Index system. e*Index Security allows you to create and maintain user profiles and user groups, to assign and expire access permissions to specific functions and actions of the Quality Workstation, and to assign user profiles to user groups. You can also configure certain parameters for security, such as a specific length of time after which a password will expire, a minimum password length, the number of passwords to maintain in the password history table, and so on. The e*Index Security GUI also provides the ability to specify users who will receive e-mail notification when certain transactions occur. This capability also requires the use of an e-mail e*Way.

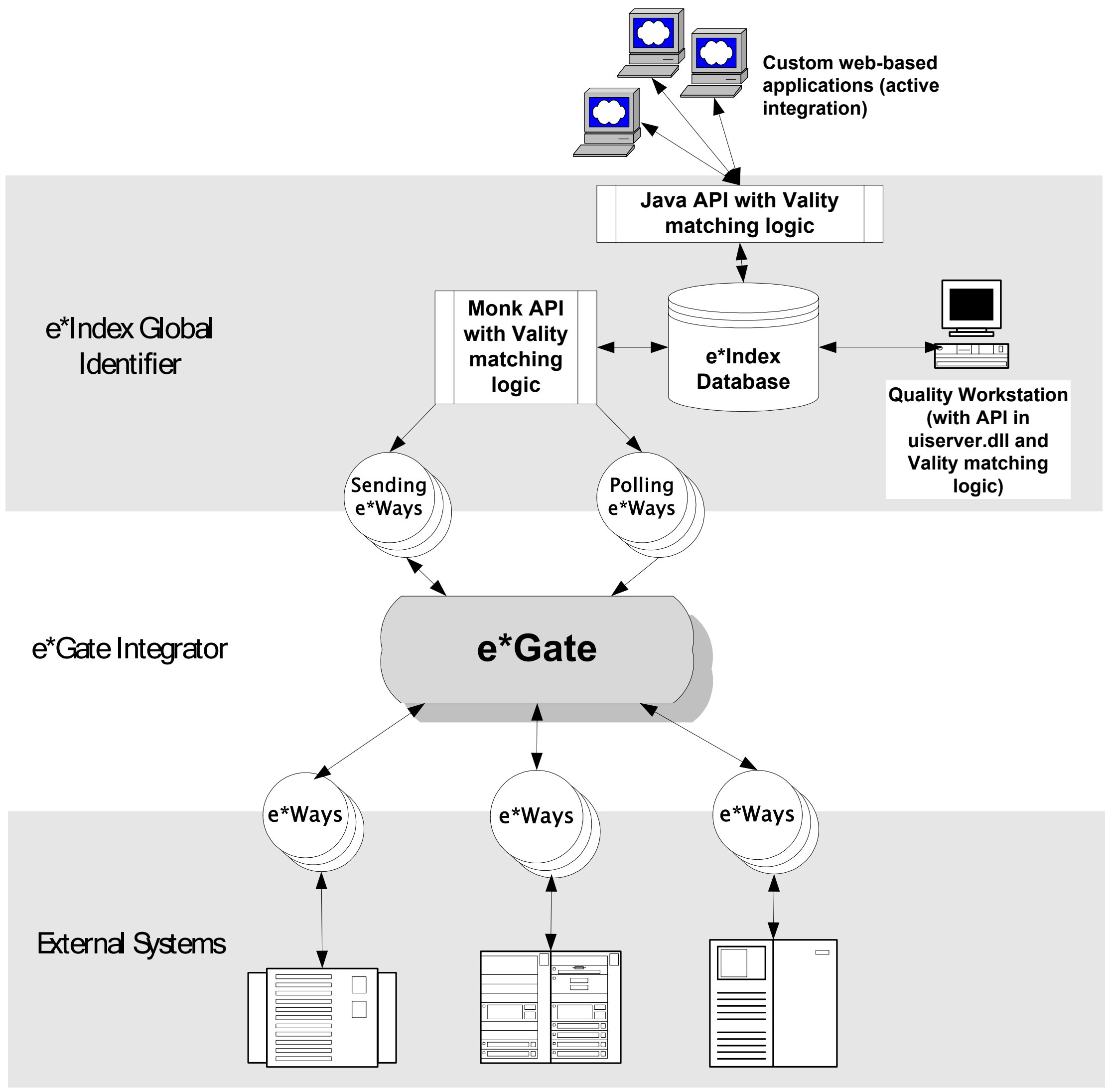
Architectural Overview

The core e*Index system consists of several different components, each acting independently of the others but working together to provide accurate data maintenance and identification. These components interact with other SeeBeyond integration products, such as e*Gate and the Database e*Ways, and through them with the external data processing products and systems used throughout a business enterprise. As data is transferred from one local system to another using e*Gate, the database is consulted and the appropriate local identifier is retrieved and used. This is a transparent process, allowing each computer system within this network to continue to use its own local identifiers.

e*Index can be implemented in a distributed computing environment, meaning that the different components of the system do not need to reside on one machine. The components of e*Index and e*Gate can be distributed across several different machines in the network. The e*Ways for each system, including the e*Index e*Ways, can reside on different servers if that provides a more efficient processing environment. Figure 2-1 on the following page illustrates the architecture of the e*Index system. Note that you should plan to build a primary, or production, system and a secondary system as a backup.

Optionally, your e*Index environment can include customized, web-based applications that use the Java API for e*Index Active Integration to interact with the e*Index database.

Figure 2-1: e*Index Architecture


Custom web-based applications (active integration)
Java API with Vality matching logic
e*Index Global Identifier
Monk API with Vality matching logic
e*Index Database
Quality Workstation (with API in uiserver.dll and Vality matching logic)
Sending e*Ways
Polling e*Ways
e*Gate Integrator
e*Gate
e*Ways
e*Ways
e*Ways
External Systems

The components that make up the e*Index system are:

- **e*Index Database**
  A relational database is used to store member data, security information, code table values, and matching algorithm processing rules. The database also stores all incoming and outgoing Events. The database is configured through the e*Index Administrator GUI on the Quality Workstation.

- **e*Index API**
  The Monk-wrapped C functions are designed to help you access and modify the e*Index database. The API uses the capabilities of the Database e*Ways to connect with the database, and to manipulate and transform the data that moves through the e*Index system. The Vality matching algorithm logic is located in the C code functions. The Monk

APIs in the Database e*Way libraries are also available for use in the Collaboration scripts for the e*Index e*Ways.

- **e*Index Sending e*Way**
  This e*Way is based on the Database e*Ways and works with the Database e*Ways' database connectivity capabilities. The e*Index sending e*Way routes data that e*Gate has received from external sources into the e*Index database, and then sends the data back to e*Gate with a UID attached. You may have several e*Ways sending data to the database. The components of this e*Way include:
  - the executable file, **stcewgenericmonk.exe**
  - configuration files (the sample configuration file included with your installation is **uidb.cfg**)
  - Collaboration Rules script (**uidb.dsc**)
  - Event Type Definition (**eiEvent.ssc**)
  - e*Way Monk functions and APIs
  - Monk external function scripts (provided in the file **ui-stdver-eway-funcs.monk**)
- **e*Index Polling e*Way**
  This e*Way queries the *ui_msg_header* table in the e*Index database for outgoing Events located in the *ui_msg_detail* table. These Events are sent to e*Gate to be routed to the appropriate external systems. These Events typically originate in the Quality Workstation. The polling e*Way uses e*Index APIs to access the e*Index database. The components of this e*Way include:
  - the executable file, **stcewgenericmonk.exe**
  - configuration files (the sample configuration file included with your installation is **uipoll.cfg**)
  - Event Type Definition (**eiEvent.ssc**)
  - e*Way Monk functions and APIs
  - Monk external function scripts provided in the file **ui-stdver-eway-funcs.monk** (the ui-poll core function is located in this file).
- **Quality Workstation**
  Quality Workstation refers to the machines on which the e*Index GUIs reside. These workstations are used to monitor and maintain member data and transactions, print reports, and perform manual changes to member information. On the Quality Workstations, you can also add processing codes, create the data elements that populate the drop-down lists for e*Index, and configure certain e*Index processing attributes, such as data formatting rules, GUI window appearance, search limits, and so on. Security for e*Index also resides on the Quality Workstation.
- **Java API for e*Index Active Integration**
  The Java API for e*Index Active Integration is a set of Java functions that enable highly customized implementations of e*Index. These functions

are based on the non-administrator functions of e*Index, such as searching for, adding, and updating member profiles. Using the API, you can design specialized web-based applications that allow users to perform a limited set of actions against the e*Index database.

For More Information

Other SeeBeyond publications may provide additional information about the components of e*Index.

To learn more about ...	See ...
The files included in the sample e*Index Schema	Chapter 3, "Customizing e*Index", of the *e*Index Global Identifier Technical Reference*
Configuring e*Ways for e*Index	Chapter 3, "Customizing e*Index", of the *e*Index Global Identifier Technical Reference*
General e*Index processing information	Chapter 2 of the *e*Index Global Identifier Technical Reference*.
Event processing in e*Index	"Learning about Event Processing" in Chapter 2 of the *e*Index Global Identifier Technical Reference*
The Java API for e*Index Active Integration	The *Java Programmer's Guide for e*Index Active Integration*
e*Index database tables	"Learning about the e*Index Database" in chapter 2 of the *e*Index Global Identifier Technical Reference*
The e*Index GUIs	The e*Index Global Identifier, e*Index Security, and e*Index Administrator User's Guides
e*Index Monk APIs	Chapter 4, "e*Index Monk APIs", of the *e*Index Global Identifier Technical Reference*

About the e*Gate Integrator

Overview

This section of the chapter provides a summary of the architecture of the e*Gate Integrator system. For a more thorough explanation of e*Gate, see Chapter 2 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

About the e*Gate Integrator System

The e*Gate Integrator system is the foundation for an e*Index implementation. The e*Gate Integrator enables the flow of information throughout an organization by providing comprehensive connectivity to applications and data stores across a network. e*Gate transforms and shares data among the e*Index database and all connected external systems.

e*Gate is based on a distributed and open architecture, and the components of the system can reside on different workstations and servers within a global network. Based on the 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, messaging standards, operating systems, databases, and communication protocols in both real-time and scheduled batch integration modes. e*Gate bridges older and newer systems, resulting in a centrally managed, unified enterprise. This architecture gives administrators the flexibility to incorporate the best applications into their business strategy, without any need to uproot older information systems investments.

Schemas

e*Gate system components are organized into *schemas*. A schema is a configuration scheme that contains all of 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/subscribe information that is at the heart of the data transportation process.

Whenever you define or configure components, establish data routing between components, or exchange files with the e*Gate Registry, you do so within the context of a single schema. You can also import or export schema components or entire schemas to move a working configuration from one environment to another; for example, moving from test to production environments. When you install e*Index, you can install a sample e*Index schema that contains all necessary schema components.

e*Gate Schema Components

The e*Gate Integrator processes events according to user-defined business logic and integrates business processes between applications. This logic is defined in the schemas you create in your e*Gate environment. The following components comprise an e*Gate schema.

- **Event Types and Event Type Definitions**
  Every *Event* (packet of data) that passes through the system is identified as a particular Event Type. An *Event Type* is a class of Events with a common data structure. For example, all Events with a specific set of fields with known characteristics and delimiters could belong to the same Event Type. *Event Type Definitions* (ETD) define Event Types, and are programmatic representations of Event Types. They are used to parse, transform, or route the data through the system.
- **e*Way Intelligent Adapters**
  e*Way Intelligent Adapters provide specialized application connectivity and also provide support for robust data processing such as business collaborations, transformation logic, and publish/subscribe relationships. e*Way adapters are multi-threaded to enable high-performance distributed processing capabilities. This multi-threaded processing allows for ultimate deployment flexibility and load balancing.
- **IQ Transports**
  IQ Transports are open queue services for SeeBeyond or third-party queuing technology, that provide robust data transport with guaranteed once-only message delivery.
- **Business Object Brokers**
  Business Object Brokers (BOBs) enable routing and load balancing between queues for implementing multi-step business processes.
- **Intelligent Bridges**
  Intelligent Bridges deliver pre-packaged process automation for key front and back office applications.
- **Collaborations, Collaboration Rules, and Collaboration Services**
  Collaborations are e*Gate's data processing "powerhouses." Each Collaboration subscribes to a specific Event Type and publishes a specific Event Type. Event Types that are subscribed to must be published by other components in the schema. Collaborations process data using Collaboration Rules. *Collaboration Rules provide* the data processing rules for each Collaboration. They process the data according to a specific set of instructions in a *Collaboration Rules Script* (CRS). *Collaboration Services* provide the mechanism through which e*Gate executes the rules in the file. Collaborations are assigned to e*Ways or BOBs.

For More Information

Other SeeBeyond publications may provide additional information about working with e*Gate components.

To learn more about ...	See ...
General information about the e*Gate Integrator	Chapter 4, "e*Gate Integrator", in the *e*Xchange eBusiness Integration Suite Primer*
e*Gate components and how they work together	The *Creating an End-to-end Scenario with e*Gate Integrator* guide
e*Gate architecture and components	Chapter 2, "Overview of e*Xchange and e*Gate", in the *e*Xchange eBusiness Integration Suite Deployment Guide*

Chapter 3

Implementation Overview

About this Chapter

Overview

This chapter provides general information about the implementation process for each component of e*Index, including the schema, GUIs, database, reports, matching algorithm, and the initial load process. For information about implementing the Java API for e*Index Active Integration, see the *Java Programmer's Guide for e*Index Active Integration*.

The following diagram illustrates the contents of each major topic in this chapter.

About the Implementation	Learn about the suggested steps for implementing e*Index
GUI Implementation	Learn about the GUI implementation components and steps
Schema Implementation	Learn about the e*Gate and e*Index schema implementation components and steps
Database Implementation	Learn about the database implementation components and steps
Reports Implementation	Learn background information about the standard reports provided with e*Index
Vality Implementation	Learn background information about the Vality INTEGRITY matching algorithm and how it is implemented in e*Index
Initial Load Implementation	Learn background information about the initial load procedure

Learning About the Implementation

Overview

This section of the chapter provides a summary of e*Index component implementation and of each phase of the e*Index implementation. SeeBeyond recommends using a proven methodology based on thorough analysis, planning, and documentation prior to beginning the actual implementation.

Implementation Components

In addition to implementing e*Gate, the e*Index implementation includes three primary e*Index components: the GUIs, the e*Index schema, and the database. Implementing the reports, Vality INTEGRITY matching algorithm, and performing the initial load of data are described in their own sections in this chapter though they can be considered part of the database implementation.

- **GUIs**
  The *e*Index GUIs* can be customized to display only the data that is pertinent to your organization. In addition, hardware, software, and networking requirements must be analyzed for each client workstation. Once the GUIs and database are installed, you can use the GUIs to configure security for e*Index, populate code table data, configure system parameters, customize the candidate selection query, customize the e*Index GUI, and define country-specific attributes.

- **e*Gate Schemas**
  *e*Gate schemas* transform data, and transfer the data between e*Gate and specific systems. To implement the e*Gate components of e*Index, you need to create schemas for each external system connecting to e*Index, and you need to create new schemas or customize the sample schema for e*Index.

- **Database**
  Implementing the *e*Index database* is a complicated task, and requires extensive knowledge in database administration. To implement the database, you need to perform data and volume analyses, sizing and distribution analyses, and database configuration. The database implementation also includes the reports, matching algorithm, and initial load, all described below.

- **Reports**
  Several reports are provided standard with e*Index, but you need to determine whether these reports are sufficient for your reporting requirements. Once you determine your reporting requirements you can

customize the standard reports, and you can create additional reports using any standard, ODBC-compliant report writer, such as Crystal Reports. e*Index reports are provided for production and initial load data.

- **Initial Data Load**
  The *initial data load*, also called the *data conversion*, involves the loading of legacy data from your existing systems into the e*Index database. You can load all existing data, or choose an appropriate subset of data to load. The data load should be performed before any live data is fed into the e*Index database from external systems via e*Gate.

- **Vality® INTEGRITY™ Matching Algorithm**
  The *matching algorithm* includes several sets of files known as rule set files. The files in the matching rule set (the **UI** rule set) provide certain logic used by the matching algorithm to perform matching functions against the data. You can customize the rule set logic for your implementation. Additional rule sets are provided to parse address information, providing the ability to search by addresses. Address-parsing rule sets are provided in four country formats: Australia, France, Great Britain, and United States. These rule sets should require little, if any customizations.

- **Candidate Selection Query**
  The *candidate selection query* defines the criteria used to perform phonetic searches and to query the database to select a pool of candidate records that are possible matches to an incoming record. Only the records returned in the candidate pool are weighted for matching purposes. You can customize the query criteria, but the logic is very closely related to the logic defined in the UI rule set files. As you will see later in this guide, customizations to the query may necessitate customizations to the rule set files and vice versa.

- **Java API for e*Index Active Integration**
  Active integration is an optional component of the e*Index implementation, and requires Java programming to create custom applications. For more information about the API, see the *Java Programmer's Guide for e*Index Active Implementation*.

Customizing e*Index

e*Index is highly configurable. In addition to customizing the database and schema, you can customize several other components of e*Index using the e*Index Administrator or e*Index Security GUI. When you are designing the system, be sure to keep in mind how, or if, you want to customize the following configurable components.

- Control keys, including weight thresholds
- GUI field display, tab labels, and search type labels

- Configurable query (for phonetic and candidate selection queries)
- Match fields for the matching algorithm and weighting logic for the match fields
- Address-parsing attributes
- Event notification
- Region-specific security

Active Versus Passive Integration

There are two different modes in which e*Index can operate: active and passive. In the passive mode, e*Index operates in the background, unnoticed by system users. It does not interact with source system applications, and all matching on person records is performed after a transaction is completed on the source system. This mode of implementation requires no changes to the existing source applications, and is commonly the first mode to be put into operation in an e*Index implementation.

In the active mode, e*Index operates in the foreground with the source system application to help facilitate transactions. Active mode can be accomplished through a set of JNI API calls, or by creating your own customized web-based applications using the Java API for e*Index Active Integration. Either way, identification and matching through the e*Index database occurs as the transaction is being processed on the source system.

In either mode, administrators can use the Quality Workstation to resolve and merge potential duplicates, make manual updates to person records, monitor data, and so on. e*Index in either mode continues to cleanse the data and simplify identification. While passive mode is of value, most of the benefits of e*Index can be realized using the active mode.

For more information about using the Java API for e*Index Active Integration, see chapters 2 and 3 of the Java Programmer's Guide for e*Index Active Integration.

Implementation Steps

This guide divides the implementation process into five primary phases. During the implementation of e*Index several of these steps may overlap. For example, data conversion, analysis, and clean up are on-going processes that will continue throughout the implementation. The implementation of each e*Index component occurs concurrently.

- **Analysis**
  The analysis phase is one of the most important phases of the project. The more information you can gather, the better you will be able to plan the implementation. The information you gather during this phase will help form the foundation for the implementation process.

- **Implementation Planning and Development Training**
  During the planning phase of the implementation, you use the information you obtained in the analysis phase to plan and schedule the implementation project, and to determine and allocate resources for the planned tasks. It is also during this phase that the implementation team members are trained on the e*Index components with which they will be working.

- **System Design and Development**
  This phase includes the design and development of the interfaces between the e*Index database and the external systems that need to share information with e*Index. This phase also includes designing and developing the e*Index database, incorporating any required business logic and security requirements, configuring system parameters, customizing the e*Index GUI, and tailoring the Vality matching algorithm and configurable query. Project team members may also begin unit testing during this phase.

- **Testing and End-user Training**
  Once the major components, the schemas, GUIs, and database, are in place, you must thoroughly test the system to verify that the implementation performs as expected. The test plans are created during the planning phase of implementation and completed during this phase. As part of this final phase before moving to production, make sure that all users are trained on the e*Index system.

- **Transition to Production and Review**
  At the successful completion of acceptance testing, you can begin to transition the system to a production environment. At this time, you can begin any necessary preloading of data. During this process, closely monitor the performance of the system and make any adjustments to assure a smooth transition into the production environment. Have a review process in place to ensure proper functioning and performance.

A Note About Planning and Analysis

The planning and analysis phases go hand-in-hand and are the most important phases in the implementation of e*Index. Thorough analysis and planning techniques lay a solid foundation for the entire implementation project. Poor planning can cause serious problems during the later phases, but a good planning process can make system design and implementation easier, more efficient, and less costly. Chapter 4, "Analyzing Requirements", discusses the analysis phase of the implementation. Chapter 5, "Planning the Implementation", discusses the planning phase of the implementation.

GUI Implementation Overview

Overview

This section of the chapter provides information about implementing the GUIs for e*Index. The GUI implementation is not a complicated task, but does require some planning and design. Required tasks for the GUI implementation include:

- Determining client workstation hardware and software requirements
- Installing the GUIs
- Defining Security
- Customizing the GUI Appearance
- Configuring GUI Processing

Determining Client Workstation Requirements

Before you can implement the e*Index GUIs, you need to analyze the hardware, software, and networking requirements for each workstation. Determine the operating system to use on each workstation, the number of client workstations to implement, the number of administrator client workstations, and the location of each client workstation. Each workstation must be able to connect to the e*Index database using ODBC or Oracle connectivity. For detailed information about hardware requirements for the client workstations, see "Quality Workstation Requirements" under "Determining Hardware Requirements" in chapter 4 of this guide. For detailed information about software requirements, see "Quality Workstation Requirements" under "Determining Software Requirements" in chapter 4 of this guide.

Installing the GUIs

e*Index GUIs are installed on *client workstations*. Before you install the GUIs, you must configure the workstations according to the requirements listed in chapter 6 of the *e*Index Global Identifier Installation Guide*. The networking components must be in place and the database client software (Oracle, Sybase, or SQL Server client) must be installed before the GUIs can connect to the e*Index database. Once each workstation is configured properly, the e*Index GUIs can be installed.

When you install the e*Index GUIs, you have the choice of installing some or all of the e*Index GUIs. Only system administrators should have all three

GUIs installed on their client workstations. User workstations should only have the e*Index Global Identifier installed. The three GUIs include:

- **e*Index Global Identifier**
  This GUI allows you to monitor and maintain member data, including checking for and resolving potential duplicate records.
- **e*Index Administrator**
  This GUI allows you to create and modify code table data, configure system parameters, customize processing, customize the GUI appearance, and so on.
- **e*Index Security**
  This GUI allows you to set up security and access permissions for e*Index users. You can also configure password parameters.

Before you can connect to the e*Index database from any of the GUIs, the initialization file, **stc_ua.ini**, must be configured for your database and database platform. Before you can use the online help provided with the GUIs, you need to register the file **d2hPopup.ocx**. Chapter 5, "Installing the GUI", in the *e*Index Global Identifier Installation Guide* provides instructions for performing both of these tasks.

Defining Security

After you install the GUIs on an administrator workstation and create the e*Index database, you can define security for the GUIs by creating user profiles and user groups, assigning user profiles to user groups, and granting access permissions to user profiles and user groups. e*Index also provides the ability for you to group the systems in your organization into regions, and then to further break down access permissions for each user by the regions you define. In addition, you can specify e-mail notifications to be sent to certain users (via the e-mail e*Way) when certain transactions occur in e*Index.

When you first install the GUIs, one administrator user, **UI**, is predefined. By default, this user has the ability to add user profiles and user groups, and to grant access permissions to both. You can use the default user profile to set up your security configuration. For detailed information about setting up security, see the *e*Index Security User's Guide*.

Customizing the GUI Appearance

Three functions of the e*Index Administrator provide the ability to modify the appearance of the e*Index GUI: Display Configuration, Country-specific Options, and Control Key Maintenance. Using the Display Configuration function, you can customize several aspects of the e*Index GUI appearance, including which fields are displayed and which are hidden, which fields are required, and the field names of the fields that are displayed. The Country-specific Options function also allows you to modify the GUI by changing the names of the search types on the Search window, changing the names of the tabbed labels, and defining the format for the SSN, Phone, and Postal Code fields. The control key **PVSUMMARY** controls whether a summary page appears on the View/Edit Person window. Some of the required customizations will result from your data analysis; other customizations will be based on preference.

Configuring e*Index Processing

After you create the e*Index database, you must use the e*Index Administrator application to define processing parameters, such as search limits, date formatting, GUI appearance, matching thresholds, and so on. You also use e*Index Administrator to customize the candidate selection query, Vality rule set files, country-specific options, and so on.

You will also use e*Index Administrator to create and modify code table data. Code table data stores those data elements that are coded in incoming Events. For example, incoming Events generally provide facility codes instead of facility names, or codes for languages instead of their descriptions. Once you complete the data analysis and table mapping for your legacy data, you can determine if you need to create any new code table data elements or if you need to modify any existing ones.

For information about working with code tables and configuration parameters, see chapters 3 through 5 of the *e*Index Administrator User's Guide.* Information is also included later in this guide in Chapter 6 under "Designing the Quality Workstation" and "Designing the Database".

Schema Implementation Overview

Overview

This section of the chapter provides information about implementing the e*Index schemas that will transfer data to and from the e*Index database. It also includes general information about implementing the e*Gate hosts and the schemas for external systems. The tasks required for the e*Gate side of the implementation are:

- Determining host server requirements
- Developing schemas for external systems
- Developing the e*Index schema
- Programming customized translations

This guide only provides summary information for setting up an e*Gate Integrator environment. For detailed information about implementing e*Gate components, see your *e*Xchange eBusiness Integrator Deployment Guide*.

Determining Host Server Requirements

When considering the best configuration for your e*Gate environment, you need to take several things into account, beginning with the hardware, operating system, and networking components to use. Depending on the size of your system, you may want to distribute the e*Gate Integrator over multiple Participating Hosts. The number of servers you use depends on the expected volume of data transactions, the size of each transaction, and the amount of translation required. Weigh these factors against the expected system performance requirements to determine the best configuration for your e*Gate environment. Depending on your expected volume and processing requirements, the e*Index schema and Database e*Way could be installed on a dedicated e*Gate Participating Host.

Developing Schemas for External Systems

e*Gate transfers data between systems using *schemas* that provide the processing rules for data being sent to and from each system. Each connected system will have its own schema. For each external system connected to your e*Gate system, you need to create a schema including these components: e*Ways, Event Types, ETDs, Collaborations, IQ transports, and possibly BOBs. For more information about these components, see "e*Gate Integrator Components" in chapter 2 of this guide.

You may be able to create some of your schemas using standard e*Way Intelligent Adapters from SeeBeyond. Other applications may require more customized schemas, and will need to be programmed.

Developing the e*Index Schema

SeeBeyond provides a standard schema for processing data into and out of the e*Index database. You can use this schema as a template for your own production schema, or you can create a new schema using the binary components of the standard schema. Developing the e*Index schema includes:

- Developing the sending e*Way
- Developing the polling e*Way
- Developing Event Types and ETDs

For detailed information about the components included with the standard e*Index schema, see "Learning About the e*Index Sample Schema" in chapter 2 of the *e*Index Global Identifier Technical Reference*. Chapter 3 of the technical reference provides information about the default configuration settings of the e*Ways in the schema. For more information about e*Index schema components, see "About e*Index Schema Components" in chapter 6 of this guide.

Developing the Sending e*Way

The sending e*Way is similar to a standard Database e*Way. It is a bi-directional e*Way that processes Events into the e*Index database using a Collaboration Rules script to specify how data is processed. Once an Event is processed, the sending e*Way returns the Event to e*Gate with a unique global identifier (UID) attached. The Collaboration Script used by the sending e*Way is **uidb.dsc**, and contains Monk functions written specifically for e*Index. To customize the sending e*Way for your installation, you need to configure the e*Way settings, customize the ETD **eiEvent.ssc**, and modify the Collaboration Script so data is processed according to your specifications. You can also customize **ui-custom.monk**, the file that creates the Monk lists used by the e*Index Monk API. If necessary, you can customize the startup and event-handling functions used by the e*Way. These functions are located in the file **ui-stdver-eway-funcs.monk**.

Developing the Polling e*Way

The polling e*Way empties the *ui_msg_detail* table (the *out queue*), which stores Events created by adding or updating records in the e*Index database through the e*Index GUI. The out queue also includes any trigger events that were processed through the sending e*Way. The polling e*Way makes these Events available to external systems via e*Gate. You do not need to use the polling e*Way to transmit Events, but, at a minimum, use the e*Way to simply empty the *ui_msg_detail* table; otherwise the table and its associated header table, *ui_msg_header*, may grow to be very large.

To use the polling e*Way, you need to configure the e*Way settings and may need to modify the Monk function **ui-poll**. This function is located in the file **ui-stdver-eway-funcs.monk**. This file also contains the definitions for the polling event-handling functions.

Developing Event Types and Event Type Definitions

The default ETD file for e*Index is named **eiEvent.ssc**, and defines a standard HL7 message. You can modify this ETD if needed. By default, all components of the e*Index schema use the same ETD. If you modify **eiEvent.ssc**, the ETD is modified for each component of the e*Index schema. To determine if or how you need to modify the ETD, create a data mapping table that illustrates the data fields in the e*Index database, the character type, and the corresponding field in the incoming Events. For a sample data-mapping table, see Appendix A of this guide.

Programming Customized Translations

Data from certain systems may require additional translation before being sent to e*Index. You can program these translations by creating Collaboration Scripts using the Monk scripting language. e*Index provides a standard Collaboration Script that you can modify using e*Gate Monk functions, Database e*Way Monk functions, and e*Index Monk Functions. For a complete reference of e*Gate Monk API functions, see the *Monk Developer's Reference*. The user's guide for each Database e*Way provides a reference of functions available to that e*Way, and the *e*Index Global Identifier Technical Reference* provides a complete listing of available e*Index Monk functions. You should be very familiar with Monk before creating customized translations.

Database Implementation Overview

Overview

This section of the chapter provides information about implementing the e*Index database. The database implementation is the most complex task of the e*Index implementation, and requires thorough knowledge in database administration. It includes the following tasks:

- Determining server requirements
- Analyzing data
- Defining code table data
- Configuring system parameters
- Customizing the matching algorithm, including address-parsing attributes (described under "Vality INTEGRITY Implementation Overview" beginning on page 3-18)
- Determining match thresholds
- Customizing the candidate selection query (described under "Candidate Selection Query Overview" on page 3-21)
- Loading legacy data (described under "Initial Load Implementation Overview" beginning on page 3-23)

Determining Server Requirements

Before you can begin the database implementation you need to analyze the requirements for the database server and determine the operating system and database platform you will use. Determine the hardware and networking components that are required to support the volume of data you expect, and define a failover system.

This task includes analyzing data flow and volume to help determine how to size the database and the best way to distribute the database. In order to create the database, you need to enter sizing and distribution information in certain database creation files, so it is imperative that you perform the analysis before you create the e*Index database. A thorough data analysis and initial load analysis will also help determine the amount of disk space required and the most efficient allocation.

For more information about determining hardware requirements, see "Database Server Requirements" under "Determining Hardware Requirements" in chapter 4 of this guide. For more information about determining software requirements, see "Database Server Requirements" under "Determining Software Requirements" in chapter 4 of this guide.

Analyzing Data

Analyzing legacy data is an on-going process throughout the implementation. This process begins with a primary investigation of a data extract using the Vality INTEGRITY Data Re-engineering Environment. This investigation helps determine how to customize the matching algorithm processing rules and how to validate the data that will initially be loaded into the e*Index database. It also highlights any data integrity issues that need to be addressed before the data is loaded into the e*Index database. Once you load data into the e*Index database in the initial test run, the results of the process help to further customize the matching rules and to fine-tune the matching thresholds for the e*Index database. You may need to perform several test data conversions before the final conversion to ensure that the algorithm and thresholds are correctly configured for your environment.

About the Data Extract

Analyzing your data requires extracting a set of records from each system that needs to share data with e*Index. At a minimum, each data record should include all fields used for matching. Data extracts from existing systems are used for two purposes – data analysis and the initial load of data into the e*Index database. You only need to extract the data that will be initially loaded into the e*Index database. You can use the same files for both processes, or you can extract a file with the minimal required fields for the data analysis, and a file with complete information for the load process. The format of the data extract is delimited, and is described in detail in chapter 3 of the *e*Index Initial Load User's Guide* in the section "Data File Requirements". It is important that SeeBeyond receives the data extracts for analysis as early in the implementation process as possible.

About the Data Analysis

SeeBeyond or a qualified third party performs the preliminary data analysis using the Vality data re-engineering tools. This process helps identify over-used default field values, field-formatting inconsistencies, frequently unpopulated or incorrectly populated fields, and so on. The analysis process also delivers a frequency analysis of the values in the first and last name fields. You can add this frequency analysis to the Vality rule set to help fine-tune the matching process, but it is not required for accurate identification and matching. Your SeeBeyond representative can help determine if a frequency file should be included in your Vality rule set files. Loading a subset of data through the initial load process can also help with the data analysis by highlighting default values, formatting errors, and so on.

About the MPI Clean up

Once the data has been analyzed, you can define how the data will be cleansed for the initial load process. SeeBeyond can recommend a clean-up approach based on the findings from the initial load data. There are multiple approaches to the MPI clean up, and the approach you choose will largely depend on the current state of your data and the requirements for processing your organization's data. For example, if your organization requires timely linkage of person data for decision support, you may choose a more intensive clean-up approach. If you are also installing an active implementation of e*Index, you may not need to complete the clean-up process until the active portion goes live.

Some examples of MPI clean-up approaches include:

- Resolve all potential duplicates prior to going live. This can be resource-intensive depending on the current state of the data and the number of records included in the initial load.
- Prioritize potential duplicates and resolve all high-priority duplicates before going live. Record priority can be based on recent activity, the matching probability weight, and so on.
- Resolve as many potential duplicate records as possible prior to going live (in order of priority), and then continue the clean-up process after bringing the system live.

Defining Code Table Data

Another part of data analysis involves gathering information about the abbreviations used for specific data elements in each sending system, such as system codes and codes for languages, religions, genders, marital statuses, and so on. In e*Index, these are known as processing codes, and the codes and descriptions of the codes are stored in a set of database tables known as code tables. When a member profile is displayed on the e*Index GUI, e*Index translates these codes into their description so the user is not required to decipher each code. The data elements stored in the code tables are also used to populate the drop-down lists that appear for certain fields in the e*Index GUI. Users can select from these options to populate the associated fields.

For more information about code tables and processing codes, see Chapter 3, "Creating Code Table Data", in the *e*Index Administrator User's Guide*. Information is also included later in this guide in Chapter 6 under "Designing Code Tables".

Determining Match Thresholds

From the data analysis, the appropriate values for the matching thresholds will emerge. As you perform data loads, you can fine-tune these thresholds further. The match thresholds are configured in the e*Index Administrator in the Control Key Maintenance function. The control keys that determine match thresholds are:

- **THRESHOLD**
  The value you set for this control key specifies the minimum matching weight at which a record is returned for a candidate search.
- **DUPTHRES**
  The value you set for this control key specifies the minimum matching weight between two records at which the records are considered potential duplicates of one another.
- **MATCHTHRES**
  The value you set for this control key specifies the minimum matching weight between two records at which the records are automatically merged. An automatic merge only occurs when there is only one record with which the incoming record has a matching weight at or above the matching threshold.

For more information about these control keys, see Chapter 4, "Maintaining Code Table Data", of the *e*Index Administrator User's Guide*. For more information about how thresholds are used for processing member data, see "About Inbound Event Processing Logic" in Chapter 2 of the *e*Index Global Identifier Technical Reference*.

Report Implementation Overview

Overview

This section of the chapter includes background information about customizing and creating reports for your e*Index environment. This component of the implementation is closely linked with the database implementation.

About e*Index Reports

e*Index provides reporting capabilities through both the GUI and through standard report scripts written in PL/SQL. From the GUI, you can print images of the member profiles displayed on the window. Using the standard report scripts, you can print reports on the state of data in the e*Index database. Production reports are run daily, weekly, monthly, and yearly, and provide information about daily transactions. You can create additional reports using PL/SQL or any standard ODBC-compliant report writer.

About GUI Reports

The e*Index GUI provides the ability to print reports for the member profiles currently displayed on the e*Index window. From the GUI, you can print information about member profiles including audit trail reports, potential duplicate reports, detail reports, associated record reports, comparison reports, and search result reports. These reports only display information about the member profiles that are currently displayed on the active e*Index window.

About Production Reports

Production reports can be run daily, weekly, monthly, and yearly. They provide information about the daily transactions that are processed through the e*Index database. These reports provide lists of potential duplicate records (for both same and different facilities), records with duplicate social security numbers, updates, merges, unmerges, deactivated records, and so on. The information you find in these reports helps you to continue to analyze your matching threshold configuration, and provides invaluable information about how data is being processed in your current configuration.

After you perform the initial load, you should run production reports against the new data to analyze its condition. These reports provide information about the initial load data and help to highlight any data issues that should be addressed. Use the production reports to help you analyze the initial load process, and to fine-tune the matching thresholds you have configured.

Designing and Creating Reports

Before you use the standard e*Index reports in a production environment, you need to determine your reporting requirements. Once you make the determination, you can customize the standard report scripts, and design and create any new reports that are required. The GUI reports cannot be customized and require no development effort.

The standard production reports are provided in the PL/SQL scripting language. You can modify any of these reports as needed to solve your reporting requirements. If the reports provided do not provide all the information you would like to see, you can create custom reports. If you are experienced using PL/SQL, you can use the existing reports as a basis for any new reports you would like to create. You can also access the e*Index database using any ODBC-compliant report writer (such as Crystal Reports), providing you with the flexibility to report on any information contained in the e*Index database.

Vality INTEGRITY Implementation Overview

Overview

This section of the chapter provides information about the rule set files you can modify to customize the matching algorithm according to your data processing requirements. For a thorough overview of how Vality is used in e*Index, see chapter 6 of the *e*Index Administrator User's Guide*.

About the Matching Logic

The Vality INTEGRITY matching algorithm provides the matching logic that helps e*Index determine the level of match between two records. The algorithm relies on a set of files, known as rule set files, to determine how the matching logic processes data. When data enters the e*Index system and no existing record is found with a matching local ID and system, the information is compared phonetically against existing records to form a candidate pool of possible matching records. The phonetic comparison is based on the candidate selection query (see "Candidate Selection Query Overview" on page 3-21). The matching algorithm then compares specific fields in the incoming data with the records in the candidate pool, and assigns a number to each candidate indicating the likelihood of a match with the incoming data.

You can configure the matching weight to be dependent on the frequency of certain values in a specific field, and to be influenced by override values defined in the rule set files. Matching weights are also influenced by the matching definitions set in the file **ui.rul**. SeeBeyond can customize the frequency, override, and matching definitions to suit your processing requirements. The matching weights determine absolute matches, potential duplicate matches, and non-matches. For more information about the Vality rule set files, how they work together, and customizations you can make, see chapter 6 of the *e*Index Administrator User's Guide*.

About Rule Set Files

When you install e*Index, several sets of default Vality files are loaded into the e*Index database. These files, known as rule set files, contain processing rules and logic used by the matching algorithm to determine the matching probability between two records and to determine how to parse addresses in the e*Index database. One rule set, named **UI**, is used in all e*Index implementations. This rule set contains logic used for determining the matching weight between records. The logic in these files is very closely related to the candidate selection query (for more information, see "Candidate Selection Query Overview" on page 3-21).

Four address-parsing rule sets are included in the e*Index installation. Each rule set defines address-parsing rules for a specific country format. The country formats supported by the rule sets are Australia, France, Great Britain, and United States. You can modify certain address-parsing attributes using the Country-specific Options function of the e*Index Administrator. Rule set files should only be modified by one who is thoroughly trained in the Vality algorithm.

Storing Rule Set Files

When you first launch the e*Index GUI, the rule set files are created from the information stored in the database and are placed in the directory specified by the VTICFG environment variable. By default, this is set to the current working directory (or the GUI home directory). When you first launch the e*Index schema, the same files are placed in the directory specified by a call to **ui-set-vticfg** in the **ui-config** Monk function. The variable is set to **\get-data-dir\bin** (by default, **\<eGate>\client\bin**). Every time you launch the GUI or schema, the rule set files are synchronized to the database to ensure that data is being processed in the same manner for transactions originating in both the GUI and the external systems. Rule set information in the database is always regarded as the most current.

Customizing the Rule Set Files

While the matching algorithm itself is not configurable, certain processing rules in the rule set files can be customized. SeeBeyond recommends that these files only be modified by someone who is highly trained in the Vality matching algorithm, and a SeeBeyond representative can customize the rule set files for you. Customization of the rule set files helps to ensure that the matching process you are using is the most suitable method for your data set. The customizations made to these files are determined by the initial data analysis and consequent testing of the matching process. The analysis and testing help to highlight special processing requirements for your data.

To modify rule set information, you need to make the changes to the appropriate rule set files, and then load that information to the database using the e*Index Administrator's Rule Set Maintenance function. Once your system has moved into production you can opt to further customize your rule set, but it is important to remember that this changes the way new data is processed and the matching weight distribution may change.

For more information about working with Rule Set Maintenance and Control File Maintenance, see chapter 6 of the *e*Index Administrator User's Guide*.

About Address-Parsing Rules

Address-parsing rules define the characteristics of the parsing performed on addresses in order to search against address components. You can modify these rules if you need to customize the parsing process, but make sure you have a full understanding of Vality address rule sets and parsing capabilities. From the Country-specific Options function of the e*Index Administrator, you can modify the following address-parsing rules:

- The address-parsing rule set to use for your implementation. You can choose Australia, France, Great Britain, or United States.
- Which street address fields to use. e*Index provides four street address fields, which may not all require parsing. If you only use two street address lines, you can modify the address-parsing rules accordingly.
- How to parse house numbers, street names, the street direction, and street types (these are predefined based on standard country formats and should not need to be modified).

Candidate Selection Query Overview

Overview

The Configurable Query function of the e*Index Administrator allows you to modify the criteria used for a standard phonetic search performed from the GUI and the phonetic search performed by e*Index when selecting a candidate pool of records that are potential matches to an incoming record. Make sure you perform a thorough analysis of your data and processing requirements before making any changes to the queries.

Customizing the candidate selection query is entirely optional. With no modification, candidates are selected as described in step 3 of "About Inbound Event Processing Logic" in chapter 2 of the *e*Index Global Identifier Technical Reference*.

About the Candidate Selection Query

When you perform a phonetic search from the GUI, e*Index uses specific information to form a SQL statement to query the database. e*Index uses similar information to form SQL statements when evaluating possible matches of records sent to e*Index from the external systems and from the GUI. When evaluating possible matches, e*Index performs a SQL search to retrieve the *candidate selection pool*. The matching algorithm processes the records in the pool and assigns them matching weights.

By default, the queries for both search types are identical, and are primarily based on the phonetic values of the included fields. You can modify the default criteria requirements and conditions so phonetic and candidate selection searches are performed using the search criteria and conditions that you define.

Vality and the Candidate Selection Query

The Vality rule set files and the candidate selection query are closely linked in the search and matching processes. The query defines the select statements for creating the candidate selection pool when new information is entered into the database. The query also creates the string that is passed to Vality to perform match processing against the candidate pool. This string is defined in the Selected Columns section of the configurable query. Each selected column that is selected for matching is added to the string that is passed to the matching algorithm, therefore each selected column must be defined in the rule set files as well.

Because of the inter-relationship between the Vality rule set and the configurable query, it is very important that if you modify the matching fields for one, you modify the matching fields for the other in the same manner.

Determining the Query Requirements

Before you make any changes to the default queries, perform a thorough analysis of your data to determine the type of information you will be processing and the combinations of that data that will provide the most reliable information for matching member profiles. Make sure you have an understanding of how the queries operate with the matching algorithm, and how the queries correlate with the rule set files. Your SeeBeyond project manager can help you determine the appropriate configuration for your system.

Initial Load Implementation Overview

Overview

This section of the chapter includes background information about performing an initial load of legacy data into the e*Index database. The initial load process is linked with the data analysis performed for the database implementation.

Loading Existing Data

As a part of the implementation process, you must determine how much of the existing data in the systems sharing data with e*Index you will load into the e*Index database. The first data conversion you run will point out areas where the legacy data has duplicate records, where default values are overused (such as default dates or social security numbers), and so on. Along with the data analysis performed through Vality, the initial load also helps you determine your matching and duplicate thresholds (see "Determining Match Thresholds" earlier) and how your SeeBeyond representative should configure the algorithm weighting probabilities. You will most likely run several test conversions on the same data in order to make sure the data is being standardized and validated accurately, and that the matching algorithm and thresholds are customized to meet your processing requirements.

The first phase of the conversion includes running a validation and standardization against the data. For this phase, you need to evaluate the changes that need to be made to the data, validations that should be run against the data, which fields will allow null values, and so on. Make sure you have thoroughly tested and analyzed the conversion data prior to the final data conversion, which occurs when you move to production. Appendix B contains a sample initial load processing log with examples of validations that can be performed against the data fields.

For more information about the validations and standardizations that can be performed on the initial load data, see the *e*Index Initial Load Programmer's Guide*.

About Encounter Dates

The initial load process is date and time driven, and uses the date_of_event and time_of_event fields to determine how to insert or update records. When the initial load program finds two records that are determined to be a match, the record with the most recent transaction date is assumed to have the most accurate information and the information in the record with the older transaction date is overwritten. This can become problematic when initial load data does not contain a transaction date, since automatic matches and updates are dependent upon the date. This can cause more current data to be overwritten by older, inaccurate data.

In order to accurately process the initial data, it is essential that the date and time of event are populated in your initial load data extracts, either with the create date or the last-modified date of the records. Records without dates tend to be from older data, so it should likely be safe to use a date far in the past for the default date for records without encounter dates attached.

About Initial Load Output Files

During the initial load process, several files are created to help you analyze the data. During the validation and standardization phase, a file is created for good data and a file is created that lists each record that failed the validation. Another file is created that describes the reason each record in the bad data file failed. You can use this information to determine additional data problems that must be addressed before the final initial load is processed. During the load phase, an error file is produced that contains information about records that could not be loaded.

Running Initial Load Reports

The production reports provided with e*Index can give you extremely useful information about the current state of your person data. They help to determine whether the matching thresholds are set correctly, whether you need to perform additional validations against the data you load, and whether the Vality matching algorithm is accurately configured. These reports are in the PL/SQL scripting language, and you can customize them to suit your reporting requirements. One good test of the matching thresholds is to run potential duplicate reports where the potential duplicate threshold is 0, and the matching threshold is very high. This way, you can get a feel for how the matching weights are being generated and how the weights are distributed. You can then see where the thresholds should be set.

For more information about production reports, see Chapter 3, "Working with Production Reports", in *Working with Reports for e*Index Global Identifier*.

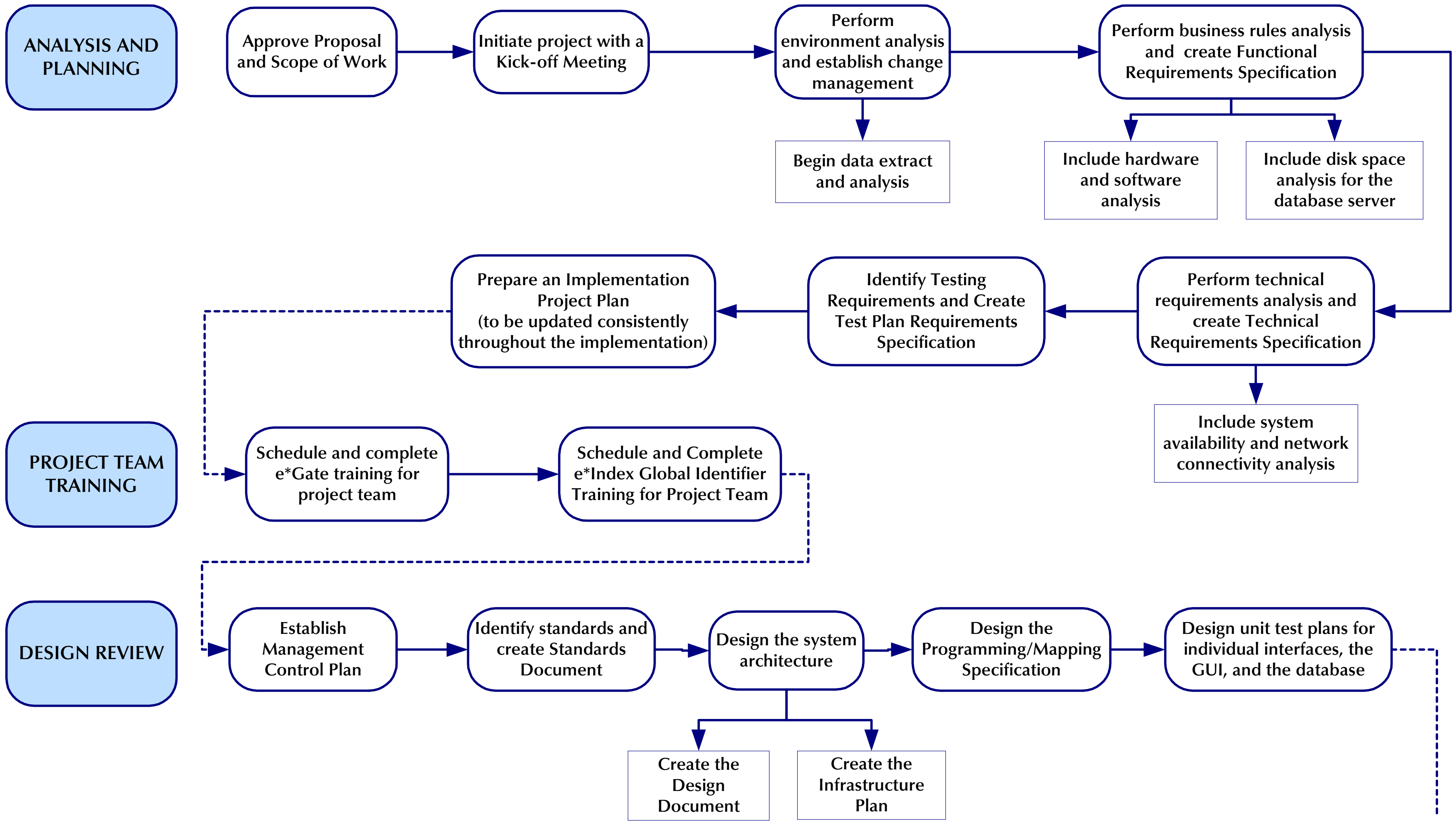
Scheduling Initial Load for Production

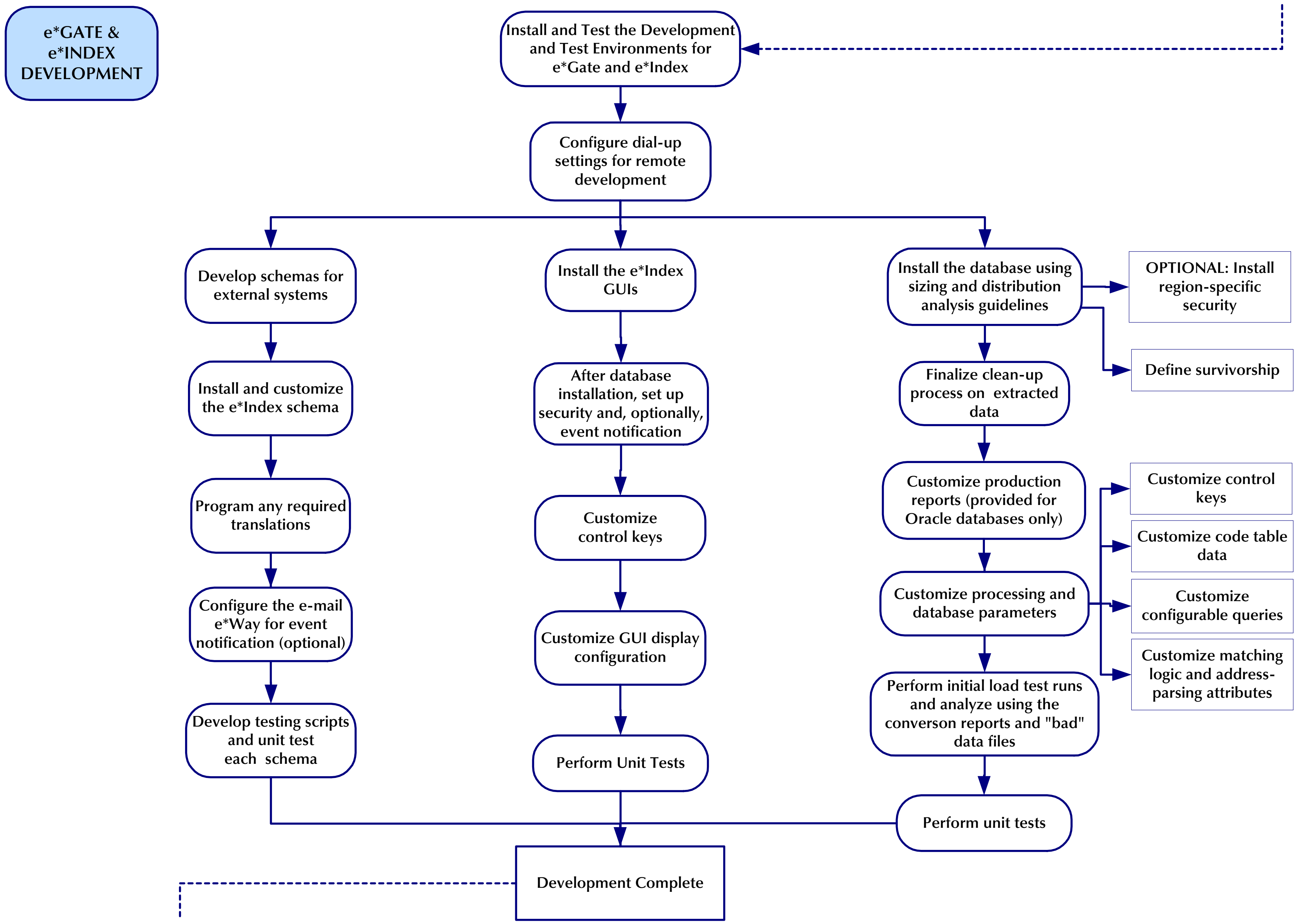
The initial load for the production phase must be completed before the full e*Index system goes into production. This is accomplished by starting the e*Ways for the external systems, and then queuing the Events coming into e*Index until the initial load is complete. Schedule the initial load carefully. You can use the test runs of the initial load to determine the length of time required for the production initial load process, and the best time to schedule the process to begin.

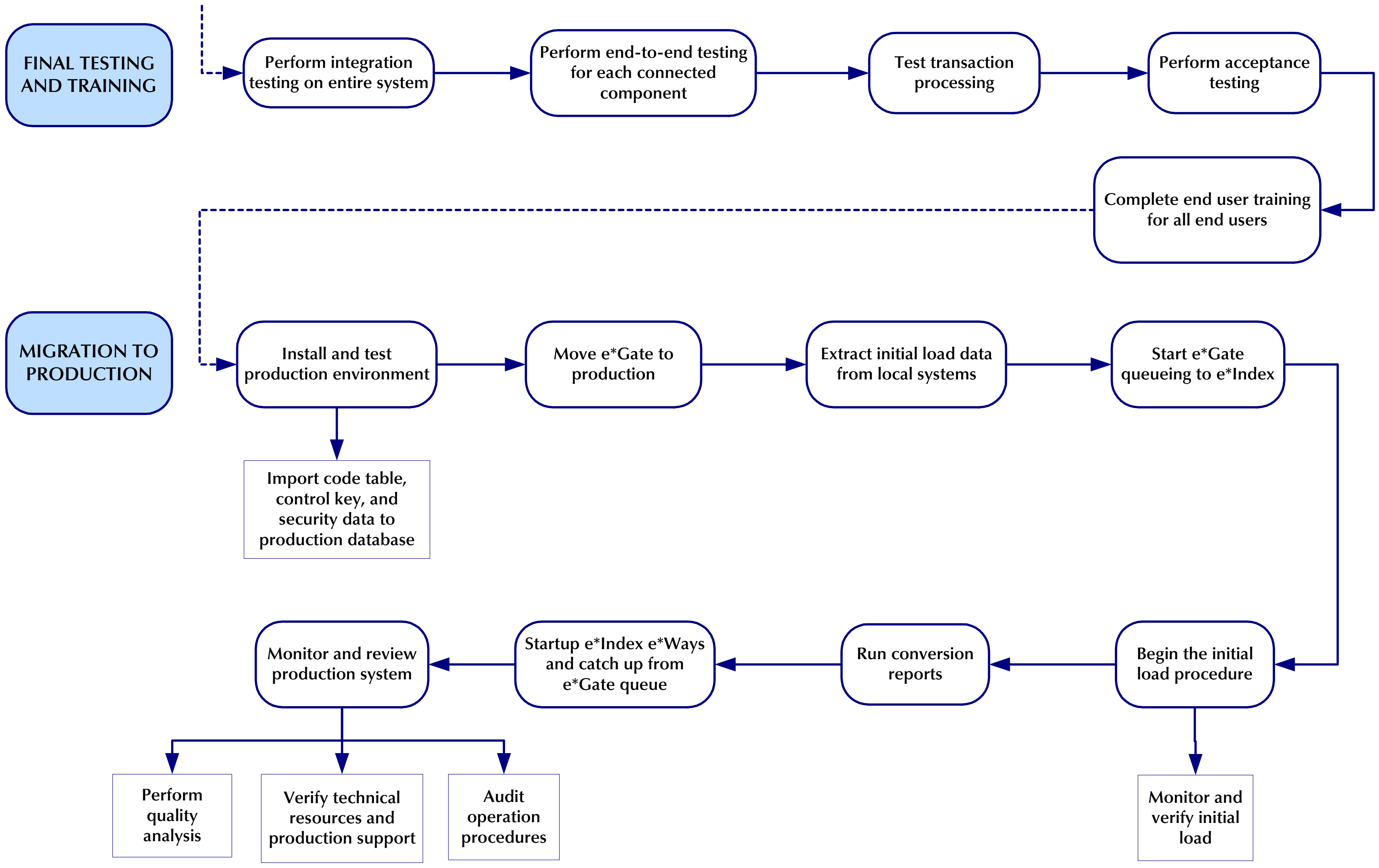
Implementation Summary

The following diagram provides a summary of the workflow of the e*Index implementation. This diagram does not provide detailed tasks for each step, but rather gives an overview of high-level tasks for the implementation. For more detail about each task, see the sample project plan provided in Appendix C of this guide.

Figure 3-1: Implementation Steps Summary Diagram


ANALYSIS AND PLANNING
Approve Proposal and Scope of Work
Initiate project with a Kick-off Meeting
Perform environment analysis and establish change management
Perform business rules analysis and create Functional Requirements Specification
Begin data extract and analysis
Include hardware and software analysis
Include disk space analysis for the database server
Prepare an Implementation Project Plan (to be updated consistently throughout the implementation)
Identify Testing Requirements and Create Test Plan Requirements Specification
Perform technical requirements analysis and create Technical Requirements Specification
Include system availability and network connectivity analysis
PROJECT TEAM TRAINING
Schedule and complete e*Gate training for project team
Schedule and Complete e*Index Global Identifier Training for Project Team
DESIGN REVIEW
Establish Management Control Plan
Identify standards and create Standards Document
Design the system architecture
Design the Programming/Mapping Specification
Design unit test plans for individual interfaces, the GUI, and the database
Create the Design Document
Create the Infrastructure Plan


e*GATE & e*INDEX DEVELOPMENT
Install and Test the Development and Test Environments for e*Gate and e*Index
Configure dial-up settings for remote development
Develop schemas for external systems
Install and customize the e*Index schema
Program any required translations
Configure the e-mail e*Way for event notification (optional)
Develop testing scripts and unit test each schema
Install the e*Index GUIs
After database installation, set up security and, optionally, event notification
Customize control keys
Customize GUI display configuration
Perform Unit Tests
Install the database using sizing and distribution analysis guidelines
OPTIONAL: Install region-specific security
Define survivorship
Finalize clean-up process on extracted data
Customize production reports (provided for Oracle databases only)
Customize processing and database parameters
Customize control keys
Customize code table data
Customize configurable queries
Customize matching logic and address-parsing attributes
Perform initial load test runs and analyze using the converson reports and "bad" data files
Perform unit tests
Development Complete


FINAL TESTING AND TRAINING
Perform integration testing on entire system
Perform end-to-end testing for each connected component
Test transaction processing
Perform acceptance testing
Complete end user training for all end users
MIGRATION TO PRODUCTION
Install and test production environment
Import code table, control key, and security data to production database
Move e*Gate to production
Extract initial load data from local systems
Start e*Gate queueing to e*Index
Begin the initial load procedure
Monitor and verify initial load
Run conversion reports
Startup e*Index e*Ways and catch up from e*Gate queue
Monitor and review production system
Perform quality analysis
Verify technical resources and production support
Audit operation procedures

Chapter 4

Analyzing Requirements

About this Chapter

Overview

This chapter provides information about the analysis phase of the implementation. It discusses the type of information you need to obtain and provides sample questions to get you started in the right direction.

The following diagram illustrates the contents of each major topic in this chapter.

Gather Information	Learn about information-gathering tools you can use for your analysis
Analyze Requirements	Learn about the different types of information you need to analyze
Personnel Requirements	Learn about the number of team members needed and the required areas of expertise
Hardware Requirements	Learn how to determine the hardware requirements for the implementation
Software Requirements	Learn about the software required in order to install and run e*Index

Gathering Information

Overview

Prepare for your implementation project by gathering as much relevant information as possible. The more comprehensive your analysis and information are, the more organized the implementation will be. You should use any survey and polling tools available to your organization and go to the right sources to get the information you need. Then organize the relevant information into useful categories. You can use any of the tools described in this section to obtain your requirements data.

Research and Interviews

Researching existing documentation and interviewing key personnel are established, traditional methods of gathering information. Your company probably has reams of paper, cabinets full of files, and databases overflowing with useful information, from management directives to marketing papers to MIS memoranda. You will find much useful implementation information in your existing documents.

Interview and talk to the employees of your organization, especially administrators of the systems you will connect through the e*Index system. While input from relevant management and MIS personnel is necessary, some of the most important sources of information are your records personnel. You want to put together a complete picture of your organization’s current and future requirements for the e*Index system. Since e*Index will be cross-referencing and storing data from multiple systems, you need to obtain information from key people for each system to determine their unique processing and data security requirements.

Your SeeBeyond representative can help you in answering specific questions on how to gather data and what kinds of data are relevant for your own implementation project.

Questionnaires

Formal surveys and questionnaires are excellent tools for obtaining information. Surveys allow you to organize your thoughts and processes, as well as help gather the desired information from others. There are several helpful publications available on creating, giving, and analyzing surveys and questionnaires. Reading some of this literature can provide a helpful background for performing these tasks.

Appendix D provides sample questions you can use to gather information for your analysis. Use this form as a guideline for the information you need to gather, or utilize modified versions of it to survey and gather information from others. Feel free to print, copy, and use it as needed. This chapter also provides some questions to help you start thinking about what information you need to obtain.

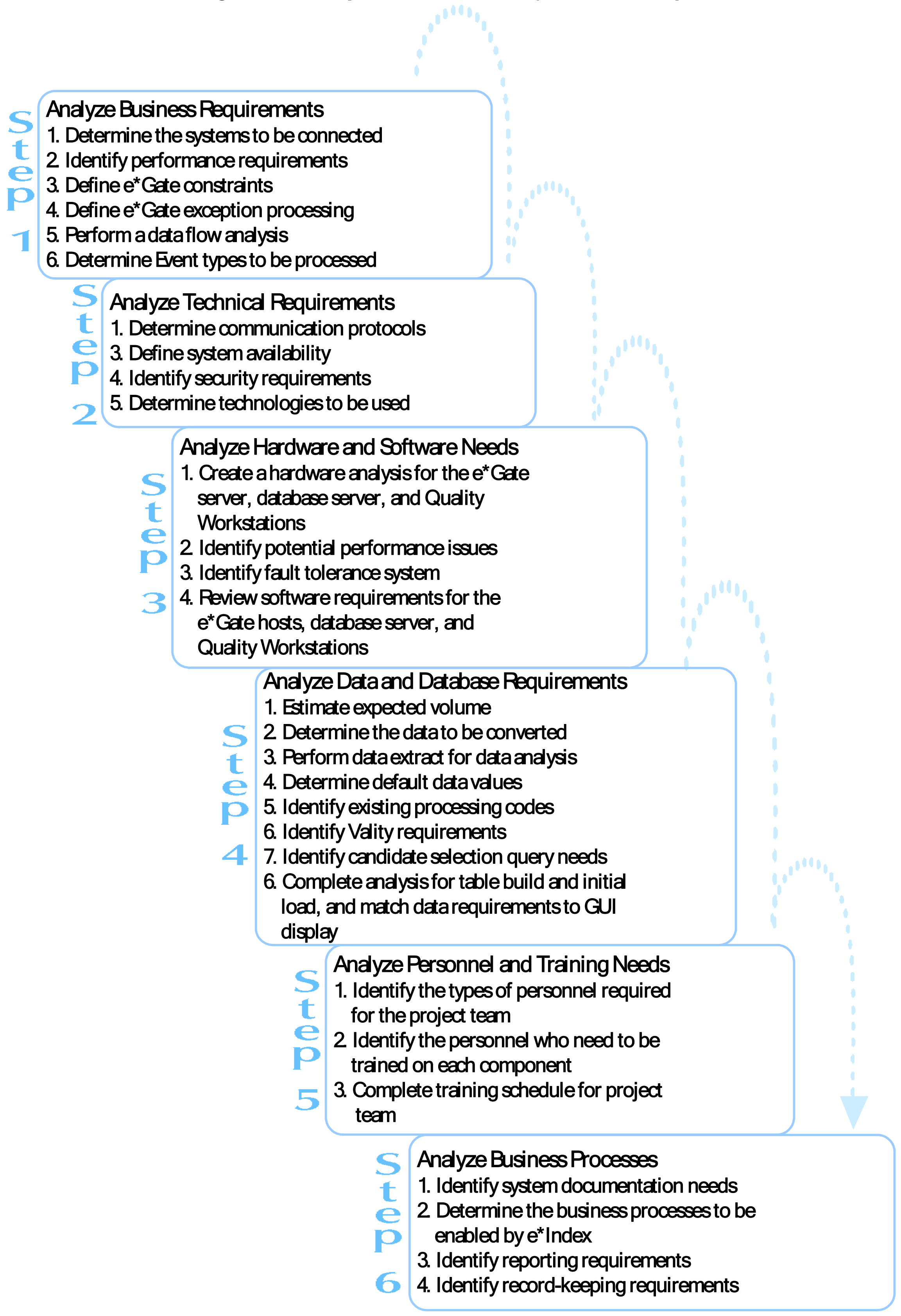
Existing Project Documents

Before the project is initiated, you already have several documents that can help you get started. If SeeBeyond responded to a request for proposal, you should have a copy of the finalized response. This document may include useful information for the project, such as a preliminary hardware quotation, third party software that is required, and so on. In addition, the approved proposal and contract contain detailed information about the scope of the project. Other helpful documents include SeeBeyond publications for e*Gate and e*Index. Your SeeBeyond Project Manager can help you select the publications you should be familiar with before beginning the implementation. The SeeBeyond Project Manager and Sr. System Engineer are also valuable sources of information for the implementation.

Analysis Phase Tasks

Figure 4-1 on the following page illustrates a summary of the steps required to perform the analysis phase of the implementation. Use this as a guideline when beginning the analysis phase.

Figure 4-1: Implementation Analysis Phase Steps


Step 1
Analyze Business Requirements
1. Determine the systems to be connected
2. Identify performance requirements
3. Define e*Gate constraints
4. Define e*Gate exception processing
5. Perform a data flow analysis
6. Determine Event types to be processed
Step 2
Analyze Technical Requirements
1. Determine communication protocols
3. Define system availability
4. Identify security requirements
5. Determine technologies to be used
Step 3
Analyze Hardware and Software Needs
1. Create a hardware analysis for the e*Gate server, database server, and Quality Workstations
2. Identify potential performance issues
3. Identify fault tolerance system
4. Review software requirements for the e*Gate hosts, database server, and Quality Workstations
Step 4
Analyze Data and Database Requirements
1. Estimate expected volume
2. Determine the data to be converted
3. Perform data extract for data analysis
4. Determine default data values
5. Identify existing processing codes
6. Identify Vality requirements
7. Identify candidate selection query needs
6. Complete analysis for table build and initial load, and match data requirements to GUI display
Step 5
Analyze Personnel and Training Needs
1. Identify the types of personnel required for the project team
2. Identify the personnel who need to be trained on each component
3. Complete training schedule for project team
Step 6
Analyze Business Processes
1. Identify system documentation needs
2. Determine the business processes to be enabled by e*Index
3. Identify reporting requirements
4. Identify record-keeping requirements

Analyzing your Requirements

Overview

When gathering and analyzing information about your e*Index system requirements, you must first know what kind of information you need. The e*Index system will link your current business systems with a single, cross-indexing database. The purpose of this system is to ensure data integrity and accurate identification of person records within the system. In as much detail as possible, you need to find out what you have and what you need.

About the Analysis Phase

Before you can implement e*Index, you must analyze your current needs, determine the goals of the implementation, and plan how the goals will be met. Your first step is to gather information about your current operating environment, data, and processing needs. Then you can determine the needs e*Index can meet and develop a plan to implement the system. There are six areas to analyze: business requirements, technical requirements, hardware requirements, data and database analysis, personnel and training requirements, and business process requirements.

Business Requirements

Your business requirements analysis defines much of the design of the e*Index system. There are three components to analyze: the e*Gate environment, the database server, and the Quality Workstations. For the e*Gate environment, you need to identify the systems that will transfer data to and receive data from e*Index, and then analyze the current information stored in those systems so certain processing codes can be standardized within e*Index. It is important to determine the systems that will share information with e*Index, what information each system will share, and the format of the Events sent and received by each system. You also need to identify codes or abbreviations used for specific fields and the length and the format of the local IDs assigned by each system.

Finally, you should examine security requirements for the e*Index system. The business requirements analysis provides the information you need to create the Functional Requirements Specification in the planning phase.

Technical Requirements

The technical requirements analysis also needs to be performed for all three components. The technical analysis includes analyzing security requirements, system availability, and the technologies to use. To analyze your security requirements, review your confidentiality standard, determine the level of security you can create using e*Index and e*Gate, identify users and their required levels of security, and so on. Your technical requirements analysis, along with the hardware and software analysis, forms the backbone of the Technical Requirements Specification that you create in the planning phase.

Hardware and Software Requirements

You need to determine your permanent hardware and software requirements very early in the implementation process. e*Index requires an e*Gate Participating Host server, a database server, and several client workstations. For each component, you need to determine the operating systems you will use, the type of fault tolerance to use, potential performance issues, the CPU processing power and disk space required, and so on. In addition, determine the number of client workstations you will require. If you are not currently using e*Gate, refer to chapter 4 in the *e*Xchange eBusiness Integration Suite Deployment Guide* to help determine your needs for the e*Gate server.

To analyze database server requirements, you need to determine the number of records that will be converted into the e*Index database and the anticipated number of daily transactions that will flow through the e*Index system. At this point, you may also want to take into consideration the potential increase in daily transactions per year, and any systems that you may want to incorporate into e*Index in the future. This analysis helps you determine database sizing, disk space allocation, performance requirements, and hardware requirements.

Finally, you need to determine the number of users who will require access to a Quality Workstation, and the number of workstations you need to install. One thing to keep in mind is that the number of concurrent users on Quality Workstations may affect the performance of the database server. Depending on your current environment, you may be able to use existing hardware for many of the client workstations.

For more information about determining hardware requirements for e*Index, see "Determining Hardware Requirements" later in this chapter. Appendix E contains hardware and transaction worksheets that may also be helpful. For more information about the required software, see "Determining Software Requirements" later in this chapter.

Data and Database Analysis

A thorough analysis of your current data and the database requirements is key to a successful e*Index implementation. You should begin analyzing your existing data as early as possible in the e*Index implementation.

Data Analysis

This part of the analysis phase includes reviewing legacy data to determine how much to load into the e*Index database, the types of data elements to be stored in the e*Index database, and the current state of the data in the legacy systems. The analysis should indicate the extent of the clean-up process and help identify a clean-up approach. The data analysis also highlights default values, identifies processing codes, and identifies mapping requirements from external systems. The data analysis will be the basis for most of the customizations you make to the e*Index system, including the e*Index schema; GUI field, tab, and search label options; candidate selection query; Vality matching logic and address-parsing logic; field formats; and database sizing.

To begin the analysis, the legacy data that will be converted into the e*Index database should be extracted and analyzed, by either SeeBeyond or Vality. Once that analysis is complete, you can run an initial test conversion program to determine the validations that need to be performed against the existing data, to help fine-tune the matching and duplicate thresholds, and to determine the level of potential duplication in the existing data. This is an on-going process during the implementation and one of the first steps in an e*Index implementation should be obtaining a data extract from the legacy systems for analysis and initial load testing. As a part of the analysis, SeeBeyond can customize portions of the Vality matching algorithm processing rules to ensure that the matching process works most effectively for your data set.

Database Analysis

In addition to the factors that affect the database hardware requirements, you need to analyze the size of the database based on the number of records in the initial load, expected volume, and the possibility of future initial loads. During the implementation, you can configure certain parameters of the e*Index database, such as search limitations and requirements, match thresholds, candidate selection query, address parsing rules, and automatic matching logic. As part of setting up the database, you must add or modify processing codes using the e*Index Administrator GUI. For more information, see "Configuring System Parameters" and "Designing Code Table Data" in Chapter 6 of this guide. Completing the transaction worksheet in Appendix E of this guide can give you useful information for sizing the e*Index database and database server.

Personnel and Training Requirements

Before you begin the e*Index implementation, determine the personnel who will implement the product, as well as the required areas of expertise. For detailed information about determining the personnel requirements for your implementation, see "Identifying the Project Team" later in this chapter.

In the initial phases of the implementation, the implementation team should receive preliminary training. Each team member needs to be trained in the components on which they are working. Before moving to production, you need to ensure that all end-users are trained on the e*Index GUIs and that the personnel who will maintain the product are trained on the system. Arrange end-user training (provided by the SeeBeyond training group) through your SeeBeyond representative, making sure all training is completed prior to the production date.

Business Planning Requirements

In order to effectively implement e*Index, you need to create documentation to help determine the scope and schedule of the implementation project. Also, develop a method of documenting the progress of the implementation and any test results. You may want to create a template for the documents that will record the progress of the implementation, flowchart and diagrams that illustrate the flow of data through the e*Index system, preliminary workflows, and so on. Make sure that all documents are approved and finalized.

Examining Your Needs

During the analysis phase, you evaluate your requirements and define the properties the system must possess in order to meet those needs. In this phase, you also identify any constraints and performance requirements. Define the functions you want the new system to perform, but not how those functions will work (this is defined during system design, which is described in Chapter 6, "System Design and Development").

This section provides some questions to help you start gathering the information you need in order to perform a thorough analysis. Keep in mind that these questions are general in nature. Begin by gathering general information, and then expand on that information with the necessary details to fully explain each category. Appendix D provides a sample questionnaire containing a more thorough list of questions to help you gather all of the necessary information.

These questions fall into the following categories:

- Business Requirements Analysis
- Technical Analysis
- Hardware Analysis
- Data and Database Analysis
- Personnel and Training
- Business Planning

Business Requirements Analysis

Business requirements determine the basic application, performance, and interfacing requirements you want your e*Index system to meet. Determine your business requirements by asking the following questions.

What existing applications do we need to cross-index?

Gather information about the existing applications that need to share information through the e*Index system. Include platform information, message type, and any other relevant information.

Example – A customer registration system on a Windows operating system sending HL7 messages, a pharmacy application on an AIX operating system sending HL7 messages, a laboratory application on a TRU64 operating system sending HL7 messages.

How should e*Index process records?

Gather information about how you want e*Index to process information. You can configure certain parameters for e*Index to conform to your processing needs (for a complete list of configurable attributes, see "Configuring System Parameters" in Chapter 6 of this guide). You can modify most of the configurable attributes after e*Index moves to production if you find they are not set to the optimal value.

Example – To conserve system resources when performing a search, we want to limit the number of records returned from a potential duplicate or audit trail search to 300. We want to be able to verify all assumed matches (records that were matched because their matching probability was above the match threshold), so we want to write those record pairs to the *ui_assumed_match* table.

What are our system performance requirements?

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

Example – Our customer databases must be completely accurate and detailed since the information is used often and is vital to the company. The most up-to-date information must be available as quickly as possible to external systems. Both detailed maintenance of this data and speed of processing are important. We need a balance between speed and maintainability.

What are our error-handling and data validation requirements?

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

Example – All Events passing through our e*Index system must be validated for specific values in certain fields. To facilitate this process, we have compiled a complete list of all the different types of data that need to be validated.

Technical Analysis

Technical requirements determine the networking, security, and system availability requirements. Determine your technical requirements by asking the following questions.

How do we want the systems to connect?

Gather information about communication protocols used by the various systems and the direction of the communications. When answering this question, you may want to think about how you will use the e*Index polling e*Way, if you use it at all.

Example – We have systems A, B, C, and D. Systems A, B, and C need to send information to and receive information from the e*Index database. System D only needs to receive information from the e*Index database.

What are our internal security requirements?

The e*Index system has security features that allow only certain persons to log on to the system and each person to have specific privileges after logging on.

Example – We only allow 30 people to log on to the system at one time. We will have four users requiring system administrator privileges, 45 users requiring access only to the e*Index GUI, and five users requiring access to the e*Index and e*Index Administrator GUIs.

What kind of fault tolerance system will we use?

Gather information about the data security level required from the operating system, the log files, and the data files. SeeBeyond can help with suggestions for fault tolerance configuration suggestions.

Example – The standard configuration for the data security level for the operating system and log files is RAID-1, and for the data files is RAID-5.

Are the systems that require information from e*Index able to accept broadcast transactions?

The e*Index system includes broadcasts to e*Index from external systems and broadcasts from e*Index to external systems. The systems receiving broadcasts from e*Index must be able to accept these Events.

Example – The customer registration system is able to accept the broadcast transactions with no modifications.

Hardware and Software Analysis

Planning the hardware and software configuration requires special considerations, for example, how many machines you need, memory (RAM) required, the number of CPUs you need, and total disk space.

What are our system and hardware limitations and constraints?

Installing e*Index requires that you have the necessary hardware and operating system software or purchase additional hardware and software to house the e*Index system. Will you use Unix or Windows? What is your budget for additional hardware and software? Are there space limitations in the area where this hardware will reside?

Example – Create a detailed checklist showing all the hardware and software you need in order to install and operate the e*Index system, and a list of what you currently have available. SeeBeyond can provide a sample hardware configuration to help you with this task.

How large will the e*Index database be, and over how many disks will it be distributed?

Gather information about the number of records that will be converted into the e*Index database and the number of expected daily transactions. Refer to the documentation for the database platform you are using (Oracle, Sybase, or Microsoft SQL Server) for recommendations on sizing and disk distribution.

Example – System A currently contains 600,000 records and processes approximately 1,000 records per day. System B contains 800,000 records and processes approximately 1,500 records per day. The database will initially store 1,400,000 records, and will process 2,500 records per day. In addition, we plan to integrate System C next year, with an initial load of 500,000 records.

What are the processing requirements for the e*Gate server?

If you are not currently using the e*Gate Integrator, you must determine the processing requirements for the e*Gate server. Determine the number of schemas you need to implement, and the expected number of transactions flowing through each schema. The size of each transaction also affects processing requirements. Your processing requirements help determine the number of e*Gate Participating Host servers you need.

Example – System A requires one inbound and one outbound e*Way, with very few data transformations. System B requires one outbound e*Way, one inbound e*Way, and one BOB. System B requires several data transformations in order to send the information to e*Index. The e*Index database requires two sending e*Ways and one polling e*Way, with moderate data processing.

Data and Database Analysis

Analyzing existing data and the e*Index database are very important parts of the implementation project. This analysis can be complicated, so be sure to obtain as much information as you can.

How many records in each system need to be converted into the e*Index database?

Gather information about the number of records that exist in each legacy system that will connect to the e*Index system, and the condition of the data in those records. This helps determine the length of time required for the initial conversion of data into the e*Index database.

Example – System A contains 750,000 records that will be converted into the database, System B contains 500,000 records, and system C contains 775,000 records. The total number of records for conversion is 2,025,000. Many of the records in System B are missing a transaction date, so will need to be assigned a default date. Records in System C contain several default values in the SSN and date fields. These default values need to be converted to null during the conversion.

How many records will be processed through the database each day?

Gather information about the number of transactions that are processed every day by each system connected to the e*Index system. This gives you an idea of how many records will be processed through the e*Index database.

Example – System A processes approximately 1,000 records per day and System B processes approximately 1,500 records per day. You can expect that approximately 2,500 records will pass through the e*Index database each day.

What is the format of the local ID used in each application?

Gather information about the format of the local IDs that are assigned by each application that will be connected in the e*Index system. Include the types of characters allowed, any punctuation, and the length of the ID.

Example – The customer registration system assigns a local ID that is ten characters long, allows only numeric characters, and has no punctuation. The pharmacy system assigns a local ID that is nine characters long, allows alphanumeric characters in the first position and only numeric characters for the remaining positions, and contains hyphens (A##-##-####).

What system codes are used in each application?

Gather a list of the current system codes that are used for the systems that assign the local ID. These codes will need to be created in the e*Index Administrator.

Example – The name of the customer registration system is System 1 and the code assigned to this system is S1. The name of the pharmacy system is System 2 and the code assigned to this system is S2.

What processing codes does each application use?

Gather a list of the current processing codes that are included in the Events generated by the systems that will send information to e*Index. These codes need to be created in the e*Index Administrator, and are assigned for such fields as Religion, Race, Language, Veteran Status, and so on (for a complete list of code tables in e*Index, see "Designing Code Tables" in Chapter 6 of this guide). If the individual systems use different codes for a specific field, select one code to use. The field must be modified during the initial load and in the e*Ways for the systems whose codes differ.

Example – One system uses the code FRN for the French language, and another uses FR. We will use FR for our processing code for the French language, create that code in e*Index Administrator, and map the discarded code (FRN) to FR.

What are the data requirements?

Gather information about the types of data messaging you use (such as HL7, XML, and so on), how many records currently exist in the external systems, and how many transactions each system sends per day.

Example – System A uses the HL7 data type. On average, the system processes about 100,000 messages per day at about 20KB per message. 70% of that data moves between 7:00a.m. and 7:00p.m. Monday through Friday. Peak transactions are generally between 1 and 5 p.m. on weekdays (40% of volume).

What default values are used in key fields?

If you use default values to populate matching fields when the actual value is unknown, you may need to define "dynamic matching thresholds" for these records to avoid mistaken assumed matches and potential duplicates. Records with specified values in the matching fields can be excluded from the matching algorithm or their thresholds can be modified to reduce false matches and duplicates.

Example – In our customer registration system, when there is no date of birth, the value defaults to 01/01/2001. If a person's name is unknown, we enter Jane or John Doe.

Personnel and Training

Deploying e*Index 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 e*Index 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 on implementing.

Example – We need three people to deploy and later operate the new system. One new person will have to be hired. All three will need to be included in the preliminary training (by SeeBeyond). The new hire must be thoroughly trained in and familiar with database administration (training in database administration is not offered by SeeBeyond).

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

Post-transition maintenance of the e*Index system could also require additional personnel or training.

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

Who will need to be trained on the Quality Workstation?

Gather information about the personnel who will need to be trained to use the e*Index GUIs, such as records personnel, database administrators, and so on. You may need data entry clerks to enter information into the e*Index GUI, and records personnel to review information from the GUI to make sure only the most current data exists. In addition, the administrator must be trained on the e*Index Administrator and e*Index Security GUIs.

Business Planning

Business planning can help you facilitate and improve your data maintenance processes. This involves documenting the requirements for the implementation and providing a project plan. Assess your needs in these areas by asking the following questions:

What are our record-keeping and documentation needs?

Be sure to set up a system for documenting your e*Index system implementation and operation.

Example – We must put a new system in place to document and diagram the total operation of our e*Index system. In addition, we must keep complete records on that operation.

How do we create an implementation project plan?

Plan your implementation well. Choose an Implementation Project Team to carry out the project, and be sure to document your plan in writing.

Flowcharts and system diagrams are definitely helpful (see Chapter 5 of this guide, "Planning Your Deployment").

Example – Several of the documents SeeBeyond recommends for the implementation are described in chapter 5 of this guide. A sample project plan is included in Appendix C.

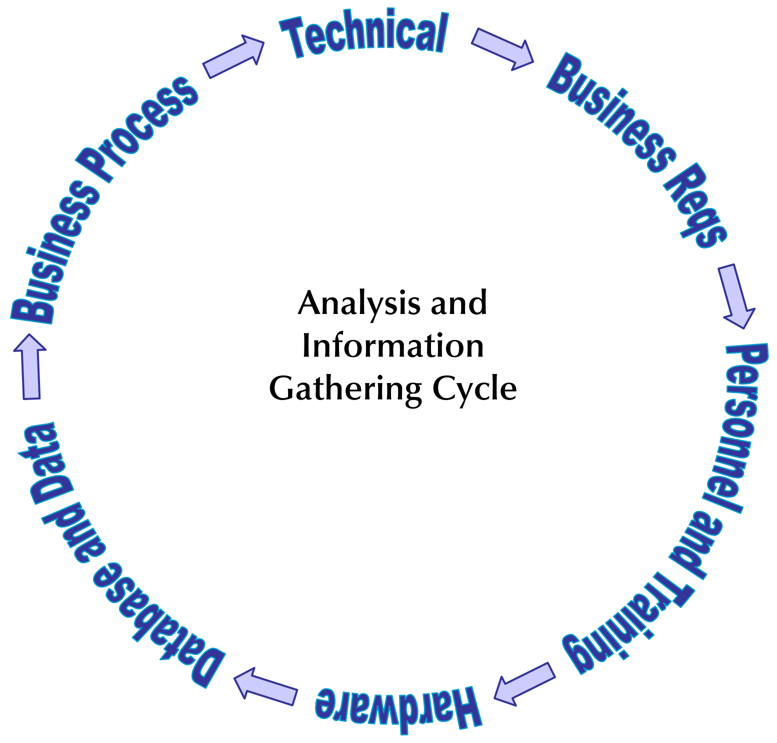
Once we have the information, what do we do with it?

Once you have answered the above questions, and the relevant questions from Appendix D, complete the process of documenting and organizing your information as correctly and comprehensively as possible. When you are finished with the analysis phase, you will use this information to help with the next phase, planning your e*Index implementation project.

Final Note:

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. Proper analysis is a cycle of analyzing information, gathering additional data, and then re-analyzing. This method helps to ensure the accuracy and usability of the requirements you define.

Figure 4-2: Analysis of Requirements Phase Information-gathering Cycle



Identifying the Project Team

Overview

This section of the chapter provides information about determining the members of the implementation project team, including their levels of expertise and required training.

Required Resources

The personnel required for an e*Index implementation varies from site to site depending on the complexity of the requirements. You should work with your SeeBeyond Project Manager to determine the best mix of personnel for your e*Index implementation. The table below outlines personnel requirements from both the client site and SeeBeyond for a typical e*Index installation.

Table 4-1: Implementation Resources

Resource	Source	Role
Project Manager	SeeBeyond	Manage project plan and resources, coordinate installation, communicate progress, oversee design, monitor adherence to scope and business objectives, and monitor integration.
Project Manager	Client	Manage project plan and SeeBeyond resources, communicate progress, oversee design, monitor adherence to scope and business objectives, and advise project team.
Senior Systems Engineer	SeeBeyond	Programmer and installer from SeeBeyond who will help plan the architecture and design of the e*Index system.
Database Administrator (DBA)	Client	Must have DBA experience. The DBA will make sure that all fields in the database are populated properly through the back end, and will monitor the database daily.
Security Administrator	Client	Work with various departments to design and establish security policies and procedures for use of e*Index. Set up and document security, and participate in unit and integration testing.
System Engineer	Client	Programmer and installer who will be responsible for implementing e*Gate schemas. Should have knowledge of the connected systems.

Resource	Source	Role
Systems Analyst	Client	Primary HIS resource for implementation. Assist in analyzing workflow and with analysis, design, and implementation of application; coordinate initial set-up and maintenance for e*Index, development of test plans, and unit and integration testing.
System Engineer	SeeBeyond	SeeBeyond expert in e*Gate and e*Index, to assist in the programming of all interfaces based on the contract agreement.
Source System Engineer/Programmer	Client	Work with SeeBeyond, records department, registration, and application team to design and program interfaces to external systems.
Network Administrator	Client	Needs to be able to assist in any network task as needed.
Test Team	Client/ SeeBeyond	The test team will consist of all the resources mentioned above to carry out the approved test plan. You may want to employ additional records personnel to help during the testing period.

Implementation Teams

SeeBeyond recommends dividing the implementation resources into four teams and one sponsor to help the implementation run smoothly. Some resources will be included in more than one team. The recommended team breakdown is as follows.

- **Executive Sponsor**
  The Executive Sponsor is a member of the management team who oversees the project, can prioritize the project within upper management, and can help allocate resources based on priorities.
- **Steering Committee**
  This committee oversees the project and keeps the Executive Sponsor informed of any important issues or changes. This group also coordinates efforts between different working groups. The steering committee should include several key team members from client sites, the SeeBeyond and client Project Managers, and a technical resource from SeeBeyond. This team should meet at least once monthly.
- **Integration Team**
  This team focuses on e*Gate, and coordinates the e*Gate installation or upgrade with other integration projects. This team should include members from each system to be integrated and one technical resource from SeeBeyond. The integration team should meet twice each month to ensure that any issues are addressed and resolved in a timely manner.

- **Application Team**
  This team focuses on changes to functionality and how the new system will impact the end users. They identify any new procedures and functions to be supported, and analyze and determine security procedures. This team also works with users to set matching weights and thresholds. The application team should include the SeeBeyond and client Project Managers, and the technical, records, and registration staff from client sites. This team should meet twice each month.

- **Technical Team**
  This team is responsible for the technical components of the implementation, including database issues, network considerations, and desktop analysis and planning. The technical team should include technical resources from client sites, an integration representative, and a senior technical resource from SeeBeyond. The team should meet twice each month.

Scheduling Training

Make sure you schedule training for your implementation team as soon as the roles have been assigned. Training for e*Gate can be extensive, and should be completed as early in the implementation cycle as possible. Recommended training classes for the implementation team include:

- e*Gate Business/Analyst and Managers
- e*Gate Basic Implementer
- e*Gate Advanced Implementer
- e*Gate DART e*Ways
- Classes for specific e*Ways as required

In addition, each member will need to be trained on the e*Index system. See your SeeBeyond Project Manager to schedule this training.

Determining Hardware Requirements

Overview

This section of the chapter provides information to help you determine the hardware required for the e*Index implementation you have planned. If the test and production systems are identical, you can use the test environment as a failover system for the production environment. An e*Index implementation project will require the following environments be installed. In addition to the environments listed below, you may need to set up an initial training environment if preliminary e*Index training is performed on-site. When designing the production system, be sure to include redundancy features for the network, servers, and subcomponents (such as CPU, hard drives, and so on).

Development

- A Host server running the e*Gate Integrator
- e*Gate Participating Hosts for developing schemas
- A server running the e*Index database
- Client workstations

Test and Backup

- A Host server running the e*Gate Integrator
- e*Gate Participating Hosts as needed
- A server running the e*Index database
- A server running the e*Index database for testing the initial load process (optional)
- Client workstations

Production

- A Host server running the e*Gate Integrator
- e*Gate Participating hosts as needed
- A server running the e*Index database
- Client workstations

Note: *For each environment, the machines must be connected over a network.*

e*Index Hardware Components

The e*Index system works with the e*Gate Integrator to create a master index of the person information in the various systems of your enterprise. This system includes client workstations and a database server in addition to the e*Gate Integrator Host and Participating Host servers. Determining the requirements for your implementation requires the analysis of each component:

- e*Gate servers
- Database server
- Quality Workstations

Appendix D provides a sample questionnaire that includes questions you need to ask when determining your hardware requirements. In addition, Appendix E provides sample hardware and transaction worksheets that can help you determine hardware needs.

e*Gate Server Requirements

The e*Index system has been deployed in a wide variety of environments, from a simple implementation cross-indexing only two systems to more complex implementations cross-indexing several different systems distributed across several different locations. In determining the requirements for the e*Gate environment for e*Index, there are several considerations to keep in mind. You need to determine the following requirements for the e*Gate server.

- Operating Systems
- Networking
- CPUs
- Hard-disk space
- Random-access memory (RAM)

This discussion does not provide in-depth information about the e*Gate server. For detailed information about determining your hardware requirements for the e*Gate side of the implementation, see chapter 4, "Determining System Requirements", in the *e*Xchange eBusiness Integration Suite Deployment Guide*.

Consideration Factors

There are several factors to consider when determining hardware requirements, such as the number of external connections, the type and volume of data being processed, how the data is processed, the number of systems transferring data, customizations to existing systems, and so on. Because these factors vary so widely between organizations, each e*Index and e*Gate implementation is different. Depending on the number and size of Events flowing through the e*Gate environment, you may implement all e*Gate components on one Registry Host server, or you may need to implement a Registry Host and one or more Participating Hosts for the various schemas in the system.

e*Gate can be distributed across multiple servers and can operate on many platforms. System stability and redundancy are important considerations. Another factor is the issue of performance versus maintainability. Remember that design choices that increase speed may decrease maintainability. Throughput, speed, or any other performance measure is dependent upon a number of parameters, including network traffic, volume, number of users, number of queries against the system, hardware speed, and so on. The flexibility of the product's architecture offers a multitude of server configurations. Once installed, the e*Gate system can be fine-tuned, in terms of both hardware and software, to optimize performance.

Operating System Requirements

One of the first steps in determining the hardware requirements is to decide which operating systems you will use for the e*Gate Host and Participating Host(s). The supported platforms for e*Index on the e*Gate environment include Windows 2003, 2000, and XP; Solaris 8 and 9, HP-UX 11 and 11i (PA-RISC), IBM AIX 5.1L and 5.2, and TRU64 5.1A and 5.1B (all with required patches). If you currently have the e*Gate Integrator in production, you can install the e*Index schemas on any existing Participating Host that runs on one of these platforms, or you can create a new Participating Host for e*Index.

Networking Requirements

The e*Index system is distributed across a network, and as such, each server you incorporate into the system must include the appropriate network components. The network hardware supported by the e*Index system includes Ethernet, FDDI, Hyperchannel, RS232, SDLC, and Token Ring. e*Index and e*Gate support a wide variety of communication protocols. SeeBeyond recommends high-speed networking whenever possible.

Server Configuration

Due to the varying requirements for each e*Gate and e*Index implementation, it is impossible to recommend a configuration of CPUs, RAM, and disk space that would suit each site. However, SeeBeyond has created a set of guidelines you can use to determine these requirements. For a detailed explanation of these guidelines, see "Hardware Requirements: Summary" in chapter 4 of the *e*Xchange eBusiness Integration Suite Deployment Guide*. Be sure to include the space required by the Oracle, Sybase, or SQL

Server client installation when estimating disk space for the participating host on which the e*Index schema will run.

Database Server Requirements

Estimating the requirements for the e*Index database can be a complex issue. There are many factors to take into consideration, such as the initial number of records to be loaded into the database, the number of transactions processed each day, the size of the incoming records, the number of queries against the system, and the potential for future growth. When you configure your database server, make sure you make room for the possibility of incorporating additional systems into the database. This discussion is limited to configuring the database as it applies to e*Index. For additional information about database server requirements, refer to the appropriate Oracle, Sybase, or Microsoft SQL Server installation guide. Appendix E provides a sample hardware worksheet and transactions worksheet that can help you determine your hardware and processor requirements.

Note: *SeeBeyond recommends that a database administrator (DBA) perform the database analysis. Configuring a database is a complex task with many factors to consider, and should only be performed by someone with a strong knowledge of the database platform you are using and database sizing. The appropriate Oracle, Sybase, or Microsoft SQL Server installation guide provides several useful suggestions for determining the requirements for your database. If you are installing an Oracle database, SeeBeyond recommends reviewing the Oracle tuning guide.*

Consideration Factors

Database server requirements vary from implementation to implementation, depending on the expected size of the database, the number of records processed daily, archiving requirements, and so on. Many possibilities and factors affect the database server configuration. Some factors are more critical than others, depending on the circumstances.

Some factors that you may want to consider in determining your hardware requirements for the e*Index database include:

- Performance versus maintainability
- Network traffic and volume
- System stability and redundancy requirements
- Number of concurrent users
- Number of e*Ways sending information to and from the database
- Desired response times
- Archiving requirements

- Initial data conversion volumes
- Typical data record size
- Database platform used

Networking Requirements

The e*Index system is distributed across a network, and as such, each server and client machine you incorporate into the system must include the appropriate network components. The network hardware supported by the e*Index system includes Ethernet, FDDI, Hyperchannel, RS232, SDLC, and Token Ring. A wide variety of communication protocols is supported.

General Guidelines

The minimum recommended hardware configuration for a typical e*Index database installation is one of the following options. These requirements are based on the minimum requirements recommended by Oracle for the installation of a *Typical* installation. Depending on the size of the database and expected volume, you should increase these recommendations as needed. For a Microsoft SQL Server installation, only the recommendations for a Windows database server apply.

- For a Windows database server, the following configuration is recommended as a *minimal* installation:
  - Windows XP, 2000, or 2003 with required patches
  - Pentium 300
  - 256 MB RAM (increase this based on the number of users, connections to the database, and volume)
  - 2.5 GB disk space plus an additional 2 KB for each person record to be stored in the database (note that this is a conservative estimate per person record, assuming that most records do not contain complete data)
  - 256-color video
  - CD-ROM device
- For a Unix database server, the following configuration is recommended as a *minimal* installation:
  - 256 MB RAM (increase this based on the number of users and connections to the database)
  - Swap space should be a minimum of twice the amount of RAM
  - 1 GB disk space plus an additional 2 KB for each person record to be stored in the database (note that this is a conservative estimate per person record, assuming that most records do not contain complete data)
  - CD-ROM device

Important!

- *Disk space recommendations do not take into account the volume and processing requirements, or the number of users. These are minimal requirements to install a generic database. At a minimum, the empty database and the database software will require 400MB of disk space.*
- *It is strongly recommended that you use striping of disks, with 1-4MB stripes, into a single logical volume.*
- *In general, Oracle Corporation recommends that disk space be spread across several small drives, rather than a few large drives for improved performance and fault tolerance.*

Operating System Requirements

One of the first steps in designing the e*Index database is deciding the operating system on which you will run the database. You can install an Oracle database on a server running Windows XP, 2000, or 2003; Solaris 8 or 9, HP-UX 11 or 11i (PA-RISC), IBM AIX 5.1L or 5.2, or TRU64 5.1A or 5.1B (all with required patches). See the appropriate Oracle or Sybase installation guide for specific information about operating system requirements. If you are using Microsoft SQL Server, you must install the database on a Windows machine. See the appropriate Microsoft SQL Server installation guide for information specific to SQL Server operating system requirements.

Estimating Disk Space

There are no firm requirements for the disk space of the e*Index database server. The documentation provided with your database software can provide solid guidelines to follow based on the expected size of the database and the volume of data flowing through the database each day. You should analyze two primary requirements: component storage and operational data. Component storage for Oracle includes the Oracle home directory files plus the default database included with the installation. These components are approximately 900 MB. Component storage for SQL Server is approximately 250 MB and for Sybase is approximately 450 MB. In addition, the default components of the e*Index database are about 350 MB. The total amount of disk space required for *minimal* database component storage in a default installation is approximately 1.3 GB for Oracle, 600 MB for SQL Server, and 800 MB for Sybase.

Generally, you can estimate the required disk space for the operational data by analyzing the estimated number of person records that will be stored in the database (providing space for potential growth). The recommended disk space for each person record is approximately 2 KB. Thus, if you expect the database to store 2 million records, the operational data would require about 4 GB of disk space (non-mirrored). The estimate of 2 KB per person record is based on an average data set. If the person data you will be storing is very complete, you may need to increase the estimate to 3 or 4 KB per person record.

When estimating required disk space, make sure to allow additional space for archiving and logging, and for your backup or failover system.

Estimating Processing Requirements

To estimate the processing requirements for the e*Index database, you must consider several factors. Some of these factors were mentioned earlier in this chapter under "General Guidelines".

- CPU type, architecture, and speed
- Presence of CPU cache and size
- Physical memory size
- Swap size
- Disk subsystem (bandwidth, latency, block size, RPM, seek time, and the presence and size of the cache)
- Network bandwidth and load
- Volume of data being processed
- Data processing requirements
- Response time requirements
- Additional loads on the database, such as archiving, logging, and backups

There is a standard benchmark in the database management system industry known as the Transaction Processing Performance Council (TPC) benchmark. For more information, visit their website at http://www.tpc.org/.

Estimating Memory Requirements (RAM)

Unix and Windows-based operating systems differ from one another in how they allocate memory. In Unix, memory allocated to a process at run time is not released to the operating system until that process terminates. In Windows, the system returns some memory to the operating system before the process to which the memory was allocated terminates. These different types of memory allocation, in addition to the unique memory allocation schemes of each operating system, make it difficult to interpret absolute memory requirements.

To determine the total RAM required for your e*Index database implementation, take into account the following information, which can then be fed into the database hardware sizing rules for the database platform you are using.

- Number of records initially being converted into the e*Index database
- The number of records that will be processed each day (on average)
- The number of Quality Workstation users accessing the database each day

Configuring for Performance Optimization

The Oracle, Sybase, and Microsoft SQL Server installation guides provide detailed information about installing the database software for optimal performance. For Oracle databases, Chapter 2 of the Oracle administrator's reference includes information about monitoring and fine-tuning your database, including tuning memory, swap space, I/O, CPU usage, block and file size, CPU usage, and so on. For SQL Server, similar information is provided in the HTML help files included with the database software. For Sybase, the *Performance and Tuning Guide* provides information about optimizing your Sybase environment.

Quality Workstation Requirements

Determining the hardware requirements for the Quality Workstations is a simple task. The Quality Workstation can run on any machine running any of these operating systems: Windows XP, Windows 2000, or Windows 2003. You do not need to run the same operating system on each workstation, and you can use any combination of those listed. The minimum requirements for these workstations are:

- Windows XP, Windows 2000, or Windows 2003
- Pentium 200
- 32 MB RAM (64 MB or more recommended)
- 1 GB disk space
- 256-color video (recommended)
- Network connectivity
- CD-ROM drive

Determining Software Requirements

Overview

This section of the chapter describes the software, from both SeeBeyond and third parties, that is required in order to implement e*Index.

Requirements

Before you begin the implementation, you should know the software requirements for e*Index. You must perform the e*Index application installation on a computer running Windows XP, 2000, or 2003.

e*Gate Integrator Server Requirements

In order to run the e*Index schema, the following software must be installed on the e*Gate host or participating host on which the e*Index schemas will be installed.

- e*Gate Integrator
- Oracle e*Way™
- HL7 Templates Add-on (only if you will be processing HL7 messages)
- e*Index schema components
- One of the following database client software versions:
  - Oracle 8.1.7 or 9*i* Client (including database and network administration tools)
  - Sybase 11.9 or 12.0 Client
  - Microsoft SQL Server 7.0 (only client files are required)

The e*Index schema can be installed on any of the following platforms:

- Windows XP, 2000, or 2003 with required patches
- HP-UX 11.0 or 11i (PA-RISC)
- IBM AIX 5.1L or 5.2, required Maintenance level patches
- Solaris 8 or 9, with required patches
- HP Tru64 5.1A, patch 5, or 5.1B

Database Server Requirements

The e*Index database can be installed in an Oracle, Sybase, or Microsoft SQL Server environment. You need to have the following software installed on the database server:

- One of the following database software versions:
  - Oracle 8.1.7 Server (recommended version is 8.1.7.2.1) or Oracle 9*i* Server
  - Sybase 11.9 or 12.0 Server
  - Microsoft SQL Server 7.0
- e*Index database scripts

The e*Index database can be installed on any of the following operating system platforms. See the installation guide appropriate to your database platform to find out which versions of these operating systems are supported. SQL Server databases can only be installed on Windows operating systems.

- Windows XP, 2000, or 2003 with required patches
- Solaris
- HP-UX
- IBM AIX
- HP Tru64

Quality Workstation Requirements

In order to connect to the e*Index database, the following software must be installed on the e*Index Quality Workstations:

- One of the following database client software versions:
  - Oracle 8.1.7 or 9*i* Client (including database and network administration tools)
  - Sybase 11.9 or 12.0 Client
  - Microsoft SQL Server 7.0 (only client files are required)
- e*Index GUI components
- Internet Explorer 4.01

The e*Index GUIs can be installed on any of the following platforms: Windows XP, 2000, or 2003.

Chapter 5

Planning the Implementation

About this Chapter

Overview

This chapter provides the information you need to start planning your e*Index implementation. The planning phase of the e*Index implementation process is built from the information you obtained in the analysis phase, and it is during this phase that you develop an action plan for the implementation and create the required planning documentation.

The following diagram illustrates the contents of each major topic in this chapter.

About the Planning Phase	Learn about the tasks to be performed in the planning phase of the implementation
Objectives	Learn how to define the overall objectives for the implementation
Identify and Schedule Tasks	Learn about the steps you need to take to start the implementation, and the documentation you need to create
Meet the Objectives	Learn how to determine when you have met all of your planning objectives

About the Planning Phase

Overview

Once you have obtained and analyzed your project information, you can begin planning the implementation project. As mentioned previously, this is one of the most critical phases of the implementation, and thorough planning helps to make the remaining phases run more smoothly.

Implementation Planning Objectives

The primary purpose of the planning phase is to initiate the project, define the system to be developed, create top-level design documents, and create a formal project plan. During this phase, you need to determine each task to be performed, assign resources for each task, and define a schedule for the project. The project plan you create sets up the process and schedule for the entire implementation project, and includes information such as resources, schedules, goals, and objectives. A sample e*Index project plan is included in Appendix C.

One of the key tasks of the planning phase is to develop detailed specifications of all required components, including message sources, destinations, and required translations; database sizing and distribution; hardware; data clean up; the initial data load; code tables; and system parameter, GUI display, matching algorithm, and address-parsing configuration. These specifications provide the basis for the requirements of the e*Index system design.

Implementation Planning Steps

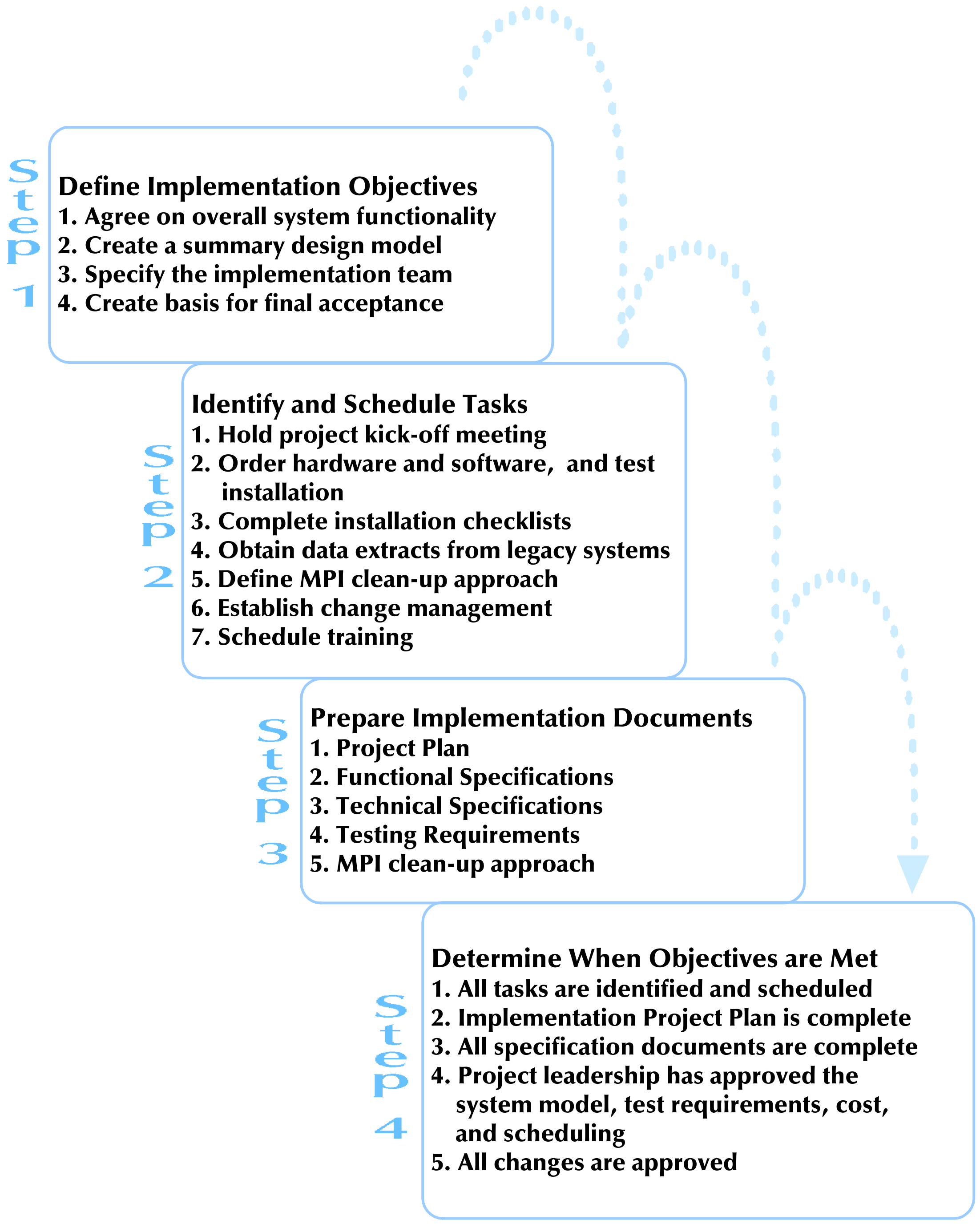
Implementation planning requires defining the resources, technical requirements, required functionality, and processes to be used during the implementation process. The major steps in implementation planning are:

- Defining overall objectives
- Identifying and scheduling tasks. This step includes creating the implementation documentation.
  - Creating a project plan
  - Creating functional specifications
  - Creating technical specifications
  - Creating a test plan
  - Developing an MPI cleanup approach
  - Scheduling and completing development training

- Determining when objectives have been met

Figure 5-1 below illustrates a summary of the implementation planning phase.

Figure 5-1: Implementation Planning Phase Steps


Step 1
Define Implementation Objectives
1. Agree on overall system functionality
2. Create a summary design model
3. Specify the implementation team
4. Create basis for final acceptance
Step 2
Identify and Schedule Tasks
1. Hold project kick-off meeting
2. Order hardware and software, and test installation
3. Complete installation checklists
4. Obtain data extracts from legacy systems
5. Define MPI clean-up approach
6. Establish change management
7. Schedule training
Step 3
Prepare Implementation Documents
1. Project Plan
2. Functional Specifications
3. Technical Specifications
4. Testing Requirements
5. MPI clean-up approach
Step 4
Determine When Objectives are Met
1. All tasks are identified and scheduled
2. Implementation Project Plan is complete
3. All specification documents are complete
4. Project leadership has approved the system model, test requirements, cost, and scheduling
5. All changes are approved

Defining Overall Objectives

Overview

Before you can plan how you will reach your objectives, you need to define the objectives you want to reach. To achieve this, follow these four steps.

- Agree on the scope
- Create a system model
- Determine the development team
- Determine the validation requirements

Agree on the Scope

Before you can proceed with the implementation, all parties must agree on the scope of the project. Form an agreement among the various involved departments on the e*Index system's overall functionality and scope. It is imperative to achieve agreement on the scope of the project before proceeding in order to avoid unnecessary delays during the implementation process. The following steps help achieve this.

- Define organized technical and functional teams or roles to handle individual phases and aspects of the implementation.
- Ensure that the system's functionality is clearly stated and agreed upon.
- Ensure that the data to be shared from each system is clearly defined and agreed upon.
- Document the functionality and scope of the project based on information obtained during the analysis phase, and ensure that this information matches the agreed upon contract.
- Resolve any differences between the scope of the approved contract and your prepared analysis and requirements information, if necessary.

Create a System Model

Creating a general model of the functions to be performed by the e*Index system provides a basis for the design phase of the implementation. Include information about the data being shared, the direction the data will flow, the type of data being processed, and the systems sharing the data. Be sure to include diagrams and supporting documents in this model. The model serves as the foundation architectural plan for the entire e*Index system design, and the diagrams and documentation you create represent the design strategy for the required e*Gate interfaces.

Determine the Development Team

Determine the team members who will design and develop the e*Index system. This team must be trained on the system to be implemented, including e*Gate, Database e*Way, and e*Index training. Some members may also require training on third-party software, such as operating system or database platform software. In addition, you must provide them with approved design documentation that clearly describes the requirements for the e*Index system. For information about the recommended personnel and the expertise required for each resource, see "Identifying the Project Team" in Chapter 4 of this guide. During this phase, you will also assign specific tasks to each team member, and possibly divide them into implementation teams.

Determine Validation Criteria

Finally, the key personnel leading the e*Index implementation must determine the criteria for validating the final product during acceptance testing. The criteria should include the testing that is required to validate the functionality of the system and ensure that it works as stated. See "Acceptance Testing" in Chapter 7 of this guide for more information about this type of testing.

Identifying and Scheduling Tasks

Overview

Once you define your objectives, you must identify the tasks that your team needs to accomplish in order to meet those objectives and a deadline for the completion of each task. Identifying and scheduling tasks consists of two primary steps.

- Initiating the implementation
- Creating implementation documents

Initiating the Implementation

The e*Index implementation typically begins with a kick-off meeting with the SeeBeyond Project Manager and key personnel in attendance. Two of the first tasks you must complete are ordering the hardware for the implementation and obtaining a data extract from the existing systems. To initiate the e*Index implementation, complete the following steps:

- **Project Kick-off Meeting**
  This meeting identifies all members of the implementation project team, and the tasks and responsibilities of each member. During the kick-off meeting, outline the reporting structure for the project and identify the personnel with whom the Project Manager will communicate to ensure that other tasks in the project are completed as planned. In addition, establish documentation requirements at this time and review any existing specifications. Depending on where you are in your analysis, you may also perform a complete analysis of requirements and business rules to be implemented during this meeting.
- **Order Hardware and Begin Installation**
  One of the first steps you should take in the implementation is to place the hardware order. As soon as the hardware arrives, install it within your network, and make sure that it meets all requirements for an e*Index implementation. For information about the software requirements for each component, see "Analyzing Software Requirements" in Chapter 4 of this guide. For information about hardware requirements, see "Determining Hardware Requirements" in Chapter 4 of this guide.

  The purpose of this task is to ensure that your hardware and software are in place, tested, and ready for the e*Index installation. For this task, perform a detailed hardware analysis before placing the hardware order and, once installed, ensure that your hardware platform and operating system fully support the e*Index software.

- **Obtain Data Extracts from External Systems**
  Early in the implementation process, extract person data from the systems that will share data with e*Index. Make sure you have determined how much of the existing data should be loaded into the e*Index database before performing the extracts, because you only need to extract the data that will be used for the initial data load. The data extracts you provide serve five purposes.
  - **Data Analysis**
    Either SeeBeyond or a qualified third-party performs the data analysis. This analysis helps determine the amount of data cleansing required for this project.
  - **Frequency File Creation**
    If you choose to use a frequency file in the Vality rule set, this file is created during the data analysis step. The matching algorithm uses this file to help fine-tune the matching process based on how commonly each name is used. This file is not required for accurate identification and matching, and this file can simply be used as an analysis tool to help you determine any modifications or additions to make to the Vality nickname file (**ui.tbl**).
  - **MPI Clean-up Approach**
    Before you can load the existing data into the e*Index database, the data may require cleansing. The data analysis will help to determine the best approach to cleansing the data. You can also analyze the initial load process to help determine the clean-up approach.
  - **Initial Load**
    Running the initial load program loads existing data from legacy systems into the e*Index database, processing each record through search and matching logic. This is an ongoing process throughout the implementation and is performed using the data extracts taken from legacy systems.
  - **Configuration Determination**
    By analyzing the types of data elements you want to store in the e*Index database, you can determine which fields to display on the e*Index GUI, the name for each field label, which fields will be required, and whether to modify the names of the tabbed pages and search types. The data analysis provides information about how or if the candidate selection query and address-parsing rules should be modified.

For more information about the data extracts and the initial load process, see your e*Index *Initial Load Programmer's Guide*. The initial load processing log in Appendix B also contains useful information for planning the initial load process.

- **Complete Installation Tests and Checklists**
  Initially, installation can be performed on temporary hardware, and then transferred to the testing and production hardware once that hardware is fully installed and configured. Perform this task as early as possible to

ensure there are no issues with your technical environment. Use an installation checklist to detail the exact hardware and operating systems on which you perform the installation. This step includes the following tasks:

- Install and test the total e*Index environment. Update the installation checklist to identify the tasks that were completed and to document outstanding issues that prevented tasks from being completed.
- Identify and verify the production, training, and test hardware, as well as the software and network requirements.
- Test the end-to-end communications with external systems to ensure that communications perform correctly and messages process correctly.

- **Establish Change Management**
  A critical factor through all phases of the project is change management. Change management identifies and tracks all changes that depart from the original implementation plan. All changes must be identified and tracked because many small changes can and will impact an implementation project as strongly as a large-scale change. Tracking all changes allows the Project Manager to plan and control a project and to track all changes to the project's scope.

- **Training Scheduled and Complete**
  The purpose of this task is to be sure that each member of the implementation team is fully trained on the appropriate components, and they are ready to begin the design and development phase. Make sure that all team members who will be working with the e*Index schema are fully trained in e*Gate, Database e*Ways, and any other applicable e*Ways. Any team members who will be working with the e*Index database and GUIs should be fully trained in e*Index. You can schedule all necessary training through your SeeBeyond Project Manager. For a list of available classes, go to the following web address: http://www.seebeyond.com/services/educationCatalogServices.asp. Click on **Registration/Schedules** for a schedule of classes.

Creating Implementation Documents

It is very important that you document not only the project plan and system specifications, but also the progress of the implementation project. Documenting specifications for the new e*Index system is critical. These documents will help you direct, monitor, and analyze the project as it progresses. SeeBeyond suggests that, at a minimum, you create these four documents: Implementation Project Plan, Functional Requirements Specifications, Technical Requirements Specifications, and Test Plan Requirements Specification. Use these documents to form the basis for system design, development, testing, and final project acceptance.

Implementation Project Plan

This is the most important document of the implementation project. It lists each required task for the implementation, and is your project guide and schedule. The project plan defines the roles and responsibilities of each group, provides a schedule of tasks, expected milestones, and any estimates. Make this plan as detailed as possible, and be sure to assess and document any project risks. Your necessary resources are budgeted using this plan. The primary purposes of this document are to define the project, show the required resource allocation for the project, and define the schedule by which the implementation tasks must be performed.

All groups involved in the implementation project must review and agree on the Implementation Project Plan, and any changes must be communicated to all affected groups. Table 5-1 below lists the general subject matter the plan should contain. A sample project plan schedule is included in Appendix C.

Table 5-1: Implementation Project Plan Structure

Contents	Description
Scope of work	This section includes a summary of the implementation project without indicating specific tasks. Base this information on the purchase contract or any equivalent document.
Project organization	Include the implementation project team organization, development organization, review organization, and any external organizations involved in the project. Clearly define the roles and responsibilities of each.
Delivery schedule	Indicate the schedules for all specified e*Index deliverables, including the final delivery date after all validation and verification tasks are complete.
Estimates	This section includes a work breakdown structure.
Overall schedule	This section contains a schedule for all implementation project tasks including resource assignments. Make sure to indicate key phase completion milestones. This schedule is further elaborated by developing various phase work plans. A sample schedule is included in Appendix C.
Resource requirements	Include the manpower, hardware, and software resources required for each phase up to production, including maintenance.
Issues and risks	Identify potential project issues and risks, and create contingency plans for any risk factors.

Contents	Description
Organizational interfaces	Identify, document, and communicate dependencies on other projects and needed inputs from other groups.

Functional Requirements Specification

In creating this document, you identify and analyze the specific functions that need to be met by the e*Index system. Carefully analyze and document the behavior of the various application components (Event Types, data flow, data processing, database operations, GUI operations, and so on). Check and verify each of these components. The typical tasks involved in creating this document include:

- Study and identify system requirements to determine the business process functionality required, and identify the system architecture needed to meet functional requirements.
- Create an e*Index architecture model to show the proposed system (see Figure 5-1 on page 4-12 for a sample architecture model).
- Create interface models to define interface requirements between e*Index and e*Gate, and between external systems and e*Gate.
- Obtain reviews of this document, and get approval in writing.

Table 5-2 below lists the subject matter of this document.

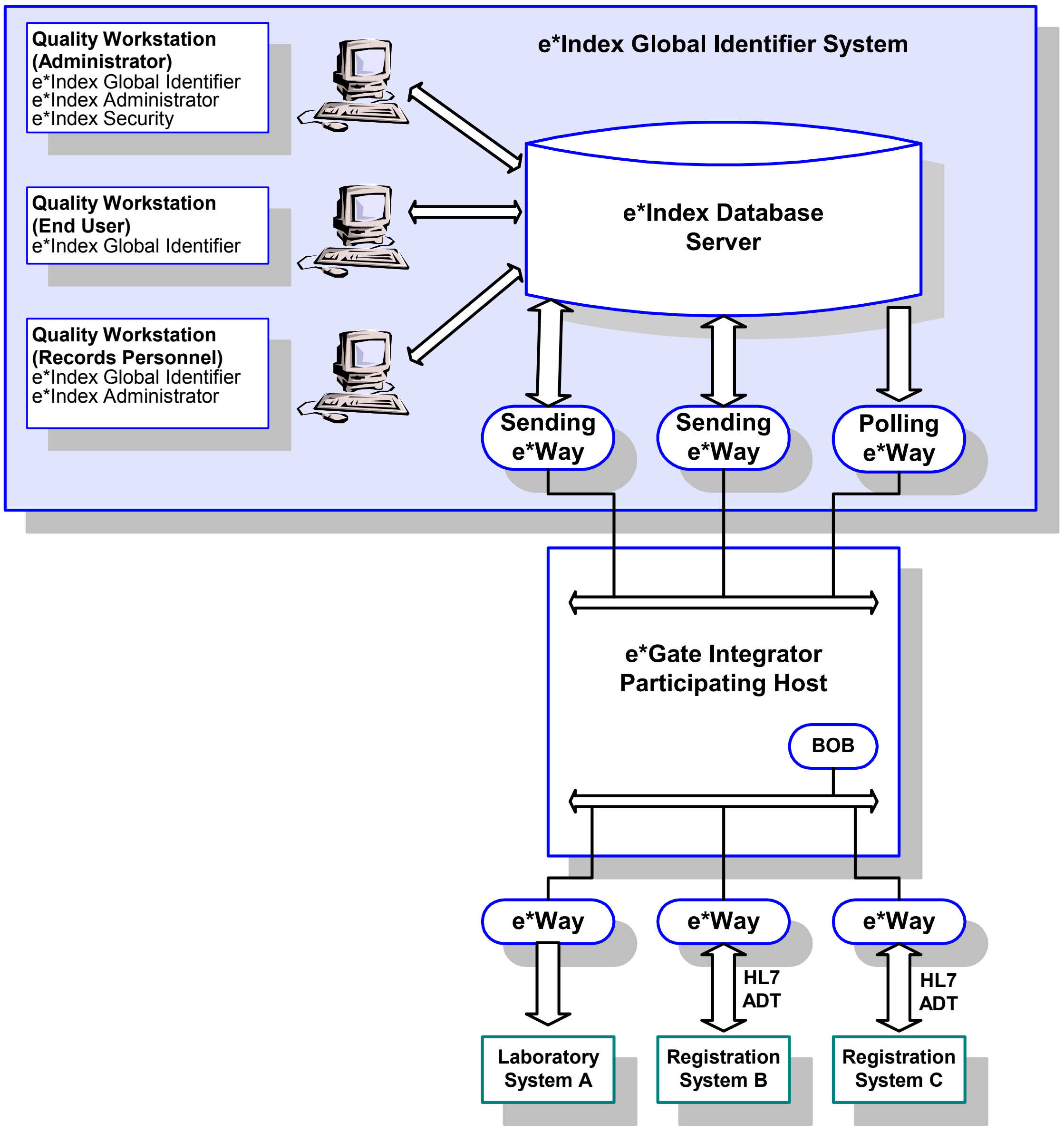
Table 5-2: Functional Requirements Specification Structure

Contents	Description/Methods
Statement of requirements	Define the objectives you want the e*Index system to meet.
Proposed e*Index architecture	Show a summary design model of the sending and receiving systems, e*Way Intelligent Adapters or BOBs to be used, Intelligent Queues, database server, Quality Workstations, and so on (see Figure 5-1 on page 5-12)
Proposed directory structure and Events that trigger e*Index processing	Provide a map of the sending and receiving systems, directory structures, and Events, including the e*Index processes that are initiated by the Events.
Exception processing	Define requirements for processing errors or Events.
Constraints	Define data volumes, performance, and any backup or archiving requirements.

Contents	**Description/Methods**
GUI specifications	Specify the fields to be displayed on the e*Index GUI, as well as any changes to tabbed page labels and search type labels. Also define processing parameters for the GUI, such as search limits, phonetic search criteria, and so on.
Search and matching specifications	Define the criteria to be used when searching for a candidate selection pool (this defines how the Configurable Query will be customized). Also define which fields will be used for matching purposes and how much value should be given to each field used for matching. Assign reliability values to each field.
Interface diagrams	Produce a diagram for each proposed interface, showing the sending or receiving system, Event processing, and any interdependencies.
Data mapping tables	Table that displays the data elements stored in the e*Index database, their character types, and the corresponding fields in Events being sent from external systems (for a sample data mapping table, see Appendix A)
Hardware and software diagrams	Show the hardware and software environment and high-level related schematics for development, testing, and production systems.

The summary design model created for the Functional Requirements Specification forms the basis of the next phase, System Design and Development. Use this model as the foundation for your e*Index system architecture and for the design documents. The figure below provides a sample diagram of an e*Index summary design model. For a sample summary design model for an e*Gate system, see chapter 3, "Analysis and Planning", of the *e*Xchange eBusiness Integration Suite Deployment Guide.*

Figure 5-1: Sample e*Index Summary Design Model


e*Index Global Identifier System
Quality Workstation (Administrator)
e*Index Global Identifier
e*Index Administrator
e*Index Security
Quality Workstation (End User)
e*Index Global Identifier
Quality Workstation (Records Personnel)
e*Index Global Identifier
e*Index Administrator
e*Index Database Server
Sending e*Way
Sending e*Way
Polling e*Way
e*Gate Integrator Participating Host
BOB
e*Way
e*Way
e*Way
HL7 ADT
HL7 ADT
Laboratory System A
Registration System B
Registration System C

Technical Requirements Specification

In creating this document, you identify and analyze your specific technical requirements. Carefully analyze and document the behavior of the various application components (Event Types, data flow, data processing, database operations, GUI operations, and so on). The typical tasks involved in creating this document include:

- Create a hardware and software model to define the environment in which the e*Index system operates.

- Create a network diagram to illustrate the network connectivity between each component.
- Describe security requirements for each component.
- Obtain reviews of this document, and get approval in writing. Table 5-3 below lists the contents of this document.

Table 5-3: Technical Requirements Specification Structure

Contents	Description/Methods
Technical requirements specification	Requirements for security, system availability, and the technology employed to meet these requirements.
Hardware and software models	Diagrams that illustrate how the hardware and software components of the system interrelate, including networking requirements.
Network diagram	Diagram that illustrates connections between external systems and e*Gate, e*Gate and the e*Index database, and the client workstations and the e*Index database. Identify points of failure and potential bottlenecks.
Security requirements	Include security requirements for each component, including the e*Index GUIs, database, and schema.
Miscellaneous	Any additional related requirements.

Test Plan Requirements Specification

You must produce a high-level test plan, highlighting the testing tasks the team must perform during each phase. This document specifies the test approach, the type of tests to perform, and the organization responsible for carrying out the tests for each test phase. Your SeeBeyond Project Manager can help you define some of the tests you must perform.

Important! *During the design phase, you develop a detailed test plan (see Chapter 6). You perform the actual testing during the testing phase before transition to production (see Chapter 7).*

The Test Plan Requirements Specification can be a single document or it can consist of a separate document for all the test phases, depending on the size and complexity of the implementation project. Table 5-4 below provides a list of the contents to include in this document.

Table 5-4: Test Plan Requirements Specification Structure

Contents	Description/Methods
Test plan	Contains a general description of the testing phase of the implementation project. Preferably, an independent test team produces this plan based on your requirements for application testing and moving the system to production.
Test phases	Includes programmer's tests, unit test, integration test, system test, rollout test, acceptance test, and so on.
Test approach	Provides details about the testing methods, such as the validation process for each test and whether testing is manual or automated. Be sure to specify message types and data requirements.
Organization	Specifies the testing team (functional and technical) and their roles.
Schedule	Defines the system availability for testing and the system resources needed for each test phase.
Resource requirements	Defines system, individual, or team resources needed for the test phases.

Chapter 7 contains a complete description of the testing phase, and Chapter 8 provides information about transitioning to production and post-transition maintenance for the implementation project.

Note: *The structure of this document indicates that you complete the planning phase prior to the design and development phase. In an actual implementation, overlap between the two phases most likely occurs. This applies to all phases of the e*Index implementation.*

Meeting Planning Phase Objectives

Overview

While the implementation team can move forward with the design phase of the e*Index implementation before the planning phase is finished, you must ensure that certain planning tasks are completed. Once the design phase begins, you may need to modify the planning documents.

Determining When Objectives are Met

You have met all implementation planning phase objectives when the following steps are completed:

1. The implementation project is successfully initiated, including the following completed tasks:
   - You have determined your project team and your design and development team, and have identified their assignments and responsibilities.
   - The implementation checklist is complete.
   - You have identified the prerequisite hardware and software, and it is installed and tested in a test environment. An order has been placed for the production hardware.
   - Preliminary testing resources are identified.
   - You have obtained a data extract from each connected system, and have begun the data analysis. The MPI clean-up approach has been defined and documented.
   - A working change management process is established.
   - All preliminary training for the implementation team is scheduled or complete.
2. The Implementation Project Plan is completed, updated (if necessary), and approved by the appropriate personnel.
3. The functional, technical, and test plan requirements specification documents are completed and approved by the appropriate personnel.
4. The implementation project leadership has reviewed and approved the above documents and the architecture design documents.

Note: *Any subsequent issues resulting in a change of the project scope or resources must be communicated and approved.*

Chapter 6

System Design and Development

About this Chapter

Overview

This chapter provides the information you need to design and build your e*Index system. The design and development phases are built around the requirements, specifications, and models you created during the planning phase. The following diagram illustrates the contents of each major topic in this chapter.

About Design & Development	Learn background information about designing and developing the e*Index system
Design Documents	Learn about the documents you need to create to define the system and enable the development process
Workstation Design	Learn about the factors to consider when designing the Quality Workstations
Database Design	Learn about the process of designing the e*Index database
e*Gate Design	Read overview information about designing the e*Gate Integrator system
e*Index Schema Components	Learn about the e*Gate schema components that are installed for e*Index
System Development	Learn about developing the e*Index system

About System Design and Development

Overview

Once you complete the analysis and planning phases, you can begin designing and developing the e*Index system. This chapter explains how to design and develop your complete, functioning e*Index system based on your implementation analysis and planning. This chapter only discusses design and development as it pertains to e*Index components. For complete information on designing and developing your e*Gate Integrator system, see chapter 5 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

System Design Documents

Designing an e*Index system is a process of successive refinement. Start with a general overview of the entire system and then develop documents with the detailed information for each component and task. This approach results in the most effective application of the technology to the implementation. To apply this top-down approach, create design documents based on the documents created during the planning phase, specifically the Functional Requirement Specification, the Technical Requirements Specification, and the Test Specification.

Designing each Component

A good starting point for the design phase is the summary design model you created during the planning phase. From the summary design model, you can design each component using the specifications developed in the planning phase. There are three primary e*Index components for you to design.

- Quality Workstations
- Database (including data analysis, data loading, reports, and candidate selection query and matching algorithm configuration)
- e*Gate components
- Java APIs for e*Index Active Integration (optional)

On the e*Gate side, you need to develop the schemas for each external system and for e*Index. If you are not currently running the e*Gate Integrator, you also need to design and develop the e*Gate system. The summary design model should indicate the e*Gate components you need to design.

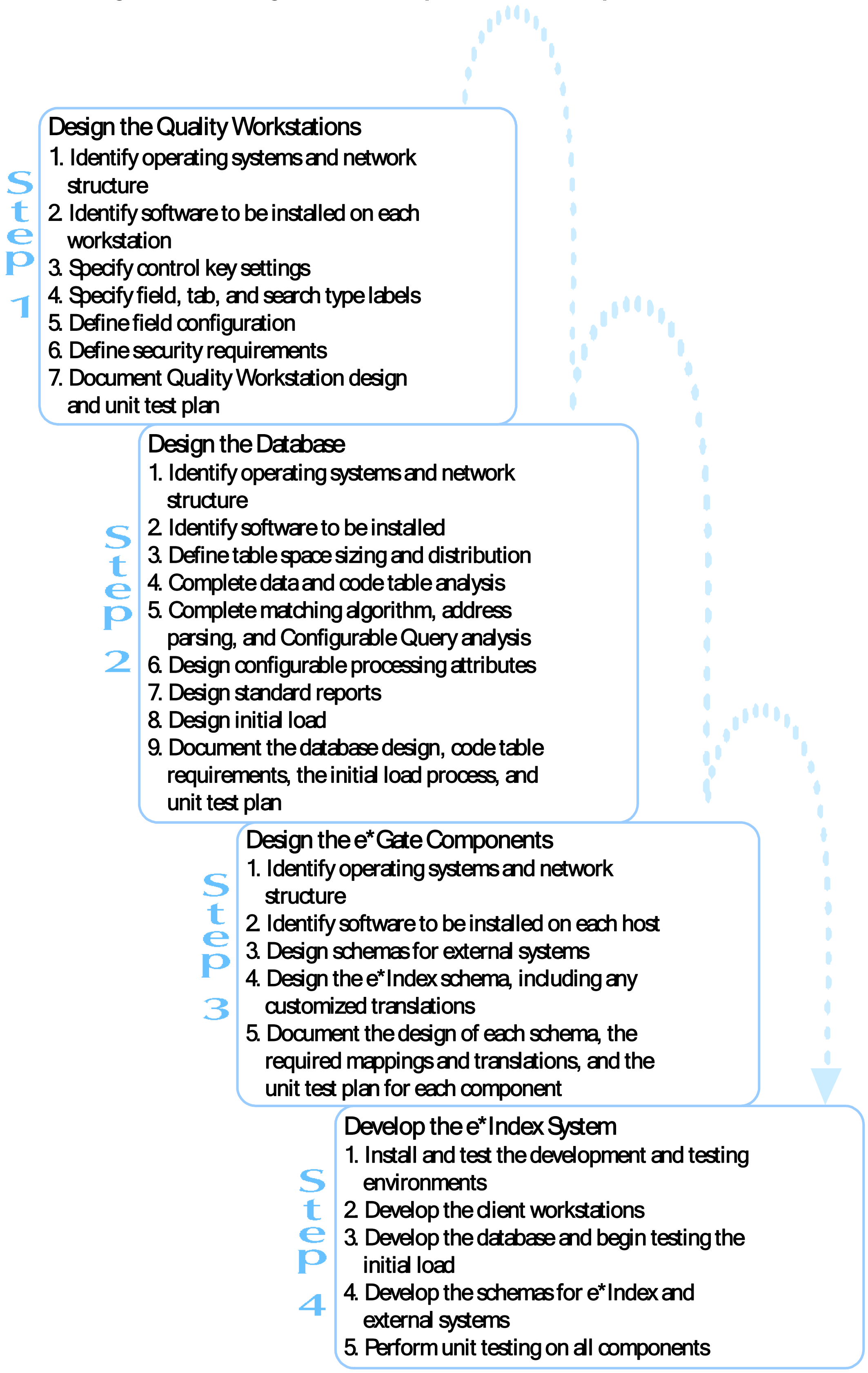
System Development

After you complete the design of the system, you can begin the actual development based on the design you created. Before development can begin, however, both the development and the test environments must be installed, networked, and tested. Development consists of the tasks required to create each component of the e*Index system according to the system design documents. Development for all components of e*Index should occur concurrently. As part of system development, each implementer must unit test the components they develop. "Developing your e*Index System", later in this chapter, provides checklists of the development tasks for each component.

Design and Development Phase Tasks

Figure 6-1 on the following page illustrates a summary of the steps required to perform the design and development phases of the implementation. Use this as a guideline when beginning the design phase of the implementation.

Figure 6-1: Design and Development Phase Steps


Step 1
Design the Quality Workstations
1. Identify operating systems and network structure
2. Identify software to be installed on each workstation
3. Specify control key settings
4. Specify field, tab, and search type labels
5. Define field configuration
6. Define security requirements
7. Document Quality Workstation design and unit test plan
Step 2
Design the Database
1. Identify operating systems and network structure
2. Identify software to be installed
3. Define table space sizing and distribution
4. Complete data and code table analysis
5. Complete matching algorithm, address parsing, and Configurable Query analysis
6. Design configurable processing attributes
7. Design standard reports
8. Design initial load
9. Document the database design, code table requirements, the initial load process, and unit test plan
Step 3
Design the e*Gate Components
1. Identify operating systems and network structure
2. Identify software to be installed on each host
3. Design schemas for external systems
4. Design the e*Index schema, including any customized translations
5. Document the design of each schema, the required mappings and translations, and the unit test plan for each component
Step 4
Develop the e*Index System
1. Install and test the development and testing environments
2. Develop the client workstations
3. Develop the database and begin testing the initial load
4. Develop the schemas for e*Index and external systems
5. Perform unit testing on all components

Creating System Design Documents

Overview

This section provides information about the documents to be created during the design phase of the e*Index implementation. The documentation you create here will provide the system developers with the information they need to begin their tasks.

About the System Design Documents

The system design documents are the blueprints used by the system developers to create a system that meets all of your defined requirements. These documents should be as detailed as possible to ensure that the implementors have a clear idea of the project direction. We recommend creating the following documents.

- **Configuration Management Plan**
  The purpose of this document is to ensure that the any changes made during the development process are documented and approved.
- **Standards Document**
  The purpose of the standards document is to ensure that correct development and implementation standards are adhered to during the course of the implementation.
- **Design Document**
  This document, along with the infrastructure plan, forms the backbone of the system architecture. The design document should include specific development information about each component of your e*Index system. Be sure to discuss each component described in chapter 3 of this guide, and also include processing attributes defined by the control keys.
- **Infrastructure Plan**
  This document defines the infrastructure in which the e*Index system will be implemented, including a system architecture, messaging standards, data exchange protocols, and so on.
- **Programming/Mapping Specification**
  The program/mapping specification defines the data mapping requirements between the systems connected through the e*Index system. Appendix A provides a sample data mapping table based on HL7 messaging to help you determine your mapping requirements.
- **Unit Test Plan Document**
  The purpose of the unit test is to ensure the functionality and operability of each developed component in the e*Index system before testing the system as a whole. Be sure to include expected results in this document.

Designing the Quality Workstation

Overview

This section of the chapter discusses designing the components of the e*Index system that will reside on the client workstations. Most of the information you need to develop the Quality Workstation design specifications is in the documents created during the planning phase of the implementation.

Considerations

To design the client workstations that will be a part of the e*Index system, you need to review the planning documents for the following information. This information should be located in the Technical Requirements Specification, the Functional Requirements Specification, or the Test Plan Requirements Specification.

- The number and physical location of the workstations
- Hardware, software, and networking requirements
- The type and quantity of data to display
- Formatting for configurable fields
- The country format to be used for address parsing
- System parameter configuration
- Security requirements
- Unit testing requirements

Use the information found in the planning documents to define the design and the development process for the Quality Workstations. For more information about hardware, software, and networking requirements for the client workstations, see Chapter 4 of this guide.

Note: *The database must be installed and operational before system parameters and security can be configured.*

Physical Configuration

To design the physical structure of the network of client workstations, list each machine on which the e*Index GUIs will be installed, and be sure to include the following information:

- Physical location

- User or user type (if applicable)
- Hardware configuration
- Operating system
- Networking components and configuration
- Database client software
- e*Index software (for example, one client workstation may have all three GUIs installed while another only has the e*Index GUI)
- Any third-party software required, such as Internet Explorer

Display Configuration

Using the data analysis performed earlier, design the appearance of the e*Index GUI. Determine which fields you want to display on the GUI and the field labels for each displayed field. Also determine which fields will be required for records entered via the GUI application. In designing the display, you can also specify new names for the tabbed pages that appear on each detail window and for the search type labels on the Search window. The final step in designing the GUI display is to define formats for the SSN, Phone, and Postal Code fields. Make sure the display configuration you decide upon is thoroughly documeted.

You can also specify that an additional tabbed page, the Summary page, appear on the GUI windows. This page displays a summary of the important information in the person record, and is enabled and disabled by the control key PVSUMMARY. This page can be enabled and disabled at anytime without affecting processing.

Parameter Configuration

You can configure several system parameters using a set of *control keys* provided in the e*Index Administrator. With the control keys, you can define search limits, specify certain field formats, specify a country-specific format, and so on. Some of these parameters control attributes of the database or of general data processing. These control keys are described later in this chapter under "Configuring Processing Attributes". The following list provides a brief explanation of the control keys that are available for GUI configuration. For detailed information about each control key and how to modify their values, see Chapter 5 of the *e*Index Administrator User's Guide*.

Search Parameters

These parameters allow you to customize how certain searches are performed in the e*Index GUI.

- **ALSRCHLMT**
  This control key allows you to limit the number of records that can be retrieved for an audit log search.
- **ATSRCHLMT**
  This control key allows you to limit the number of records that can be retrieved for an audit trail search.
- **DEMOSRCLMT**
  This control key allows you to limit the number of records that can be retrieved for a demographic search.
- **DOBYYREQ**
  This control key allows you to specify whether the date-of-birth year field is required for an alphanumeric search.
- **ENDTIME**
  You can specify a default value for the **End Time** field on the Audit Trail and Potential Duplicate Search windows with this control key. The default value of this control key is **23:59** in order to include the entire **End Date** specified.
- **EXTNSVSRCH**
  This control key allows you to specify whether e*Index will search through alias names when performing a search. Note that if you want extensive searching to be performed for records entered through the e*Index schema, you need to modify the configurable query as described in "Configuring Extensive Searching" in chapter 5 of the *e*Index Administrator User's Guide*. In determining whether to use extensive searching, take into account how it may affect processing speed.
- **LNEXCTRSRCH**
  This control key allows you to specify the number of characters that must be entered in the last name field to perform an exact search for members. For more information about exact searches, see "What are Partial and Exact Match Searches" in chapter 3 of the e*Index Global Identifier User Guide.
- **PDSRCHLMT**
  You can limit the number of records than can be retrieved for a potential duplicate search using this control key.
- **SEEDEACTIV**
  This control key allows you to specify whether deactivated profiles are displayed in the results of member searches.

- **SEEMERGED**
  This control key allows you to specify whether merged profiles are displayed in the results of member searches.
- **SNDXSRCH**
  This control key allows you to disable or enable the phonetic (Soundex) search in e*Index.
- **SRCHDOB**
  Use this control key to define a range of years surrounding the specified date of birth to use for a demographic member search.
- **STARTTIME**
  Use this control key to specify a default value for the **Start Time** field on the Audit Trail and Potential Duplicate Search windows. The default value for this control key is **00:01** to include the entire **Start Date** specified.
- **THRESHOLD**
  This control key allows you to specify the minimum matching probability weight for the records retrieved during a search.

Formatting Parameters

The following parameters define how certain fields are formatted in the e*Index GUI.

- **ALLOWNUM**
  Use this control key to specify whether numeric characters can be entered into the **First Name** and **Last Name** fields in e*Index.
- **DATEFRMT**
  You can specify the date format displayed by e*Index using this control key.
- **MIXEDCASE**
  This control key allows you to specify whether users can enter person information in both uppercase and lowercase in the e*Index GUI.
- **SHORTID**
  This control key allows you to specify that the value entered in the Social Security Number field can be less than nine characters.

General GUI Parameters

The following parameters define general debugging parameters as well as whether peripheral software is being used. Currently the Encounter module is not available with e*Index.

- **AUDITONOFF**
  This control key allows you to log and display the amount of time it takes to perform certain functions in the e*Index GUI.

- **DEBUGSQL**
  Use this control key to specify whether to display the active SQL statement before executing a search from the GUI.
- **ENCOUNTER**
  This control key allows you to specify whether the encounters add-on module for e*Index is being used.
- **SEEMSGCODE (GUI only)**
  This control key allows you to specify whether the message code of a message dialog appears on the dialog next to the message dialog title.
- **SMARTCARD**
  The value you specify for this control key determines whether the smart card capabilities for e*Index are enabled.
- **UVAUDITLOG**
  Ths control key allows you to specify whether instances of access to the *ui_person* table are stored for retrieval by the Audit Log function. In determining whether to use this functionality, take into account disk space and how it may affect processing speeds.

Designing Security

There are several factors you need to take into consideration when designing the security requirements for e*Index. e*Index Security allows you to create profiles for the users who will operate e*Index and user groups to which you can assign those users. When designing security, determine the users who need to use the e*Index GUIs, and categorize those users into groups that share similar processing requirements. From this information, you can define the user groups to which the users will belong. There are six steps to designing security.

- Determine who requires access to the e*Index GUIs
- Determine the regions to which each user requires access (optional)
- Divide users into groups of users who require the similar security access permissions
- Determine the access permissions to assign to each user and user group you define
- Configure security parameters
- Define event notification (optional)

For complete information about creating user profiles and user groups, and assigning access permissions, see Chapter 3 of the *e*Index Security User Guide*.

***Note:** e*Index Security only defines security for the e*Index GUIs. You may want to further define security to the database using Oracle's security capabilities. Event notification requires configuring an e-mail e*Way in addition to setting up security.*

Defining Users and User Groups

Defining security for e*Index involves a variety of administrative tasks, including adding user profiles and groups, assigning user profiles to groups, expiring and reinstating user profiles, and assigning access permissions to user profiles and user groups. Each user who accesses the e*Index GUIs and database must have a user profile defined in e*Index Security, including a user logon ID and a password. You can assign user profiles to user groups to automatically assign those profiles the access permissions assigned to that group, or you can grant users access permissions on an individual basis. When you initially install e*Index Security, you can only log on as the predefined user, the *Super User*. By default, this user automatically has access to all functions and windows of the e*Index GUIs, and can be used to set up security components.

Region-specific Security

e*Index allows you to further control access to the e*Index database by *regions*. A region can be a physical location of a system or systems, or it can be a category in which certain systems are grouped. This function allows you to limit the information each user can view to data originating in a system or a set of systems. Regional-specific security is not required, but if you use it you need to document a name and processing code for each region, along with the systems that will be included in each region. A system can only be associated with one region.

Configuring Security Parameters

You can configure a set of parameters for e*Index Security that define requirements for the login passwords used by the personnel who log into the e*Index GUIs. There are only four security parameters to configure.

- **DAYSPASEXP**
  This control key specifies the number of days a password can be active before a user is prompted to create a new password.
- **MINPWDLEN**
  This control key specifies the minimum length for an e*Index password.
- **IDLETIMER**
  This control key specifies how long the e*Index GUIs can remain inactive before the application automatically closes.

- **PASSHIST**
  This control key define a specific number of passwords that are retained in the password history. Passwords in the history list cannot be reused.

Event Notification

Using e*Index Security, you can define the events for which certain users should receive e-mail notifications. Designing this component requires that you document each user who should receive notifications and the events for which they should be notified. This component also requires that you design an e-mail e*Way to pick up messages from the out queue (the *ui_msg_detail* table) and then send the messages to the e-mail addresses specified in the ZEN segment of each message.

Customizing Code Table Data

Once both the GUIs and the database are installed, you can use the e*Index Administrator GUI to add and modify the information in the processing code tables of the database. Since this task falls under database design, it is described later in this chapter under "Designing Code Table Data" in the section "Designing the Database".

Designing the Database

Overview

This section of the chapter discusses designing the database and database server configuration. The person responsible for this task should be a database administrator familiar with the e*Index database and with your data processing requirements. Most of the information you need in order to develop the database and initial load design specifications is located in the documents created during the planning phase of the implementation.

Considerations

To design the e*Index database, review the planning documents for the following information. This information should be located in the Technical Requirements Specification, the Functional Requirements Specification, and the Test Plan Requirements Specification.

- Database server hardware, software, and networking requirements
- Locations of the dump files, backup files, log files, and table files (database distribution)
- Tablespace requirements, index sizing, and extents
- Initialization settings
- Processing code table requirements
- Data processing parameters
- Matching algorithm and address-parsing customizations
- Configurable query customization
- Data load requirements
- Reporting requirements

Information about Oracle Database Specification Files

This discussion concerns files you may need to modify for Oracle database installations only. For Microsoft SQL Server and Sybase, you only need to specify file sizes, and the process is fairly simple. Before reading this, you should be familiar with the procedure for installing an Oracle database for e*Index and the files used for the procedure (for more information, see chapter 3, "Installing an Oracle Database", in the *e*Index Global Identifier Installation Guide*). It is also useful to review the information in the Oracle administrator's guide before you modify any of the following files.

There are several factors to consider when creating an Oracle e*Index database. The primary issues are sizing, extent, and distribution attributes. These attributes are controlled by several scripts and initialization files that you can modify in order to customize your installation. These files, created when you install the database scripts, include **defs.sql**, **initSID.ora**, **create_db_step1_*.sql** (where * is either **NT** or **unix**), and **create_ui_tablespaces.sql**, and **create_ui_tables.sql**. The installation guide provides information about modifying **defs.sql** and **initSID.ora**. This section provides more detailed information about these two files.

For more information about modifying the **defs.sql** file, see "Step 6: Modify **defs.sql**" in chapter 3, "Installing an Oracle Database", chapter 4, "Installing a SQL Server Database", or chapter 5, "Installing a Sybase Database", in the *e*Index Global Identifier Installation Guide*. For more information about modifying the **initSID.ora** file, see "Step 7: Modify **initSID.ora**" in chapter 3 of the installation guide.

About defs.sql

The **defs.sql** file for Oracle defines several database attributes such as pathnames, the database home, the database SID name, and tablespace sizes. The files **create_ui_tablespaces.sql** and **create_ui_tables.sql** use information specified in this table to create the e*Index tablespace files and indexes for specific tables. The scripts that create the actual database tables also use information specified in **defs.sql**.

In the first section of **defs.sql**, you define the database SID name, the database home directory, the log file home directory, and the tablespace home directory. These variables are referenced throughout the file.

The "Log File Paths" section defines the path and file names for the log files, as well as the size of the log files (all are the same size). The "Tablespace Paths" section defines the path and file names for the Oracle system and e*Index tablespace files. The Oracle system tablespace files include **system01.dbf**, **rbs01.dbf**, **temp01.dbf**, **tools01.dbf**, and **users01.dbf**. The tablespaces specific to e*Index include **ui_data.dbf ui_indx.dbf**.

The "Oracle tablespace info" section defines the sizes of the Oracle system tablespace files. For recommendations on sizing these files, see your Oracle documentation. The "Tablespace info" section defines the file sizes, initial extent sizes, and next extent sizes for the tablespaces specific to e*Index.

About initSID.ora

The **initSID.ora** file is the startup file, or *pfile*, for the e*Index database. In the first section of this file, you specify the location and names of the control files, background dump files, and user dump files. The **initSID.ora** file specifies several additional initialization parameters, such as buffer information, shared pool size, database locks, and so on. This file is used each time the database starts up. For more information about defining the parameters in this file, see the Oracle administrator's guide".

About create_db_step1*.sql

The **create_db_step1_unix.sql** and **create_db_step1_NT.sql** files contain the SQL commands that create the Oracle system tablespaces. These files use the pathnames and sizes specified in the **defs.sql** file to determine the size of the Oracle system tablespace files and where to create them. In **defs.sql**, this information is defined in three sections: Tablespace Paths, Log File Paths, and Oracle tablespace info. In **create_db_step1_*.sql**, you can specify additional information about tablespaces such as the sizes of the initial and next extents, the minimum and maximum number of extents, the percent increase, and so on. This file also creates the rollback segments for the database. You only need to modify the file for the type of system on which the database resides (Windows or Unix). This file is called when you run **create1**(**.bat**) to create the database instance.

About create_ui_tablespaces.sql

The **create_ui_tablespaces.sql** file creates the e*Index tablespaces **uidata** and **ui_indx**. This file uses variables defined in **defs.sql** for tablespace locations and sizes. In **create_ui_tablespaces.sql**, you can specify the number minimum and maximum extents for each tablespace, as well as the percent increase. The sizes of the extents for each e*Index tablespace are defined in **defs.sql** in the "Tablespace info" section and the tablespace locations are defined in "Tablespace Paths". **create_ui_tablespaces.sql** is called when you run **create2.bat** to create the structures of the e*Index database.

About create_ui_tables.sql

The **create_ui_tables.sql** file creates all the e*Index database tables and the indexes against each table. This file does not reference the **defs.sql** file for sizing information. You can specify the number of minimum and maximum extents in **create_ui_tables.sql**, and you can specify the percent increase. **create_ui_tables.sql** is called when you run **create2.bat** to create the structures of the e*Index database.

Distributing the Database

SeeBeyond recommends creating a distributed database with mirrored backup. A database administrator specializing in the database platform you are using should perform the distribution and backup design. The distribution of the database is defined in the files **defs.sql** and, for Oracle only, **initSID.ora**. Both files are provided in the database installation scripts. For Oracle, the log file home directory is defined by the variable LOG_FILE_HOME, and the tablespace home directory is defined by the variable TBL_SP_HOME. For a SQL Server or Sybase installation, the log file and table directories are defined by the variable @DBPATH, which is referenced in the last section of the file where all paths and filenames are defined.

Before designing the database distribution, view the **defs.sql** file to learn about the defined log file and tablespace paths. For Oracle, review the information under "Information about Oracle Database Specification Files" earlier in this chapter.

Determining Tablespace Sizes

The **defs.sql** file defines the sizing configuration for the database. In this file, you specify how to allocate storage by defining the space to allocate for each file and extent. Many of the modifications you make to this file are dependent upon the expected size of the database, and SeeBeyond recommends that these modifications be performed by a database administrator (DBA) specializing in the platform you are using. For a SQL Server database, sizing is defined by these three variables: @PFSIZE, @FGSIZE, and @LOGSIZE. For a Sybase database, sizing is defined by these four variables: @DBDEVSIZE, @LOGDEVSIZE, @DBSIZE, and @LOGSIZE

For an Oracle installation, sizing is defined in the sections "Oracle tablespace info" and "STC tablespace info". Additional sizing specifications for Oracle are defined in the file **initSID.ora** (for more information about **initSID.ora**, see "About initSID.ora" earlier in this section and "Determining Initialization Settings for Oracle" next).

Determining Initialization Settings for Oracle

For an Oracle database, SeeBeyond provides an Oracle initialization file, **initSID.ora**, to specify startup attributes when you create the e*Index database. **initSID.ora** is also referenced each time the database starts up. In this file, you need to specify certain parameters such as the Oracle SID name and pathname for the database, and pathnames and filenames for the dump and control files. You also need to specify information about block sizes, buffers, logging and audit processes, and so on. Many of the parameters you need to enter are dependent upon the expected size of the e*Index database. SeeBeyond recommends that the requirements for this file be determined by an Oracle database administrator (DBA). For more information about the initialization file, see "About initSID.ora" earlier in this chapter.

Note: *SeeBeyond recommends that you make the database block size (defined by the variable **db_block_size**) large, for example 8K or higher. For more information about this variable, see "DB_BLOCK_SIZE" in the Oracle administrator's guide.*

Designing Code Table Data

The e*Index database stores processing codes for many of the attributes assigned to each member record. For example, e*Index stores a list of languages spoken by the people whose information is stored in the database, along with a processing code for each language. The code for French is "FRN", the code for English is "ENGL", and so on. e*Index uses these codes to map the same types of codes from the external systems into the e*Index database. The descriptions you enter for each code populate certain fields and their corresponding drop-down lists in the e*Index GUIs so users do not have to translate the processing codes. Users can select information from the drop-down lists when adding or updating person records. Be sure the code table data includes all of the possible data elements that will be sent to e*Index from external systems.

Most of the code tables are automatically populated when you create the e*Index database, but you may need to modify some of these codes to match the codes used by the external systems with which you are sharing data. You can also insert new codes as needed. Document the various codes used by each external system, and determine the default codes you want to use in the e*Index database. Then, you can map the codes for external systems that use codes other than the default codes you define for e*Index.

Following is a list of the different types of code tables in the database. Some of these code tables are country-specific, such as zip codes, postcodes, and districts of residence, and may not be required for each implementation.

- Address Types
- Citizenships
- Countries
- Departments
- DORs (districts of residence)
- Driver License Issuers
- Ethnicities
- Events (these codes should not be changed)
- Event Notifications
- Genders
- Languages
- Marital Statuses
- Nationalities
- Non-unique ID Types
- Application Messages (these codes should not be changed)
- Person Categories
- Phone Types

- Predefined Messages
- Races
- Regions
- Religions
- Source Applications
- States
- Statuses (these codes should not be changed)
- Suffixes
- Systems
- Titles
- Veteran Statuses
- VIP Statuses (these codes should not be changed)
- Zip Codes

Tip: *For complete information about these tables, see chapter 3 of the* e*Index Administrator User's Guide. *This chapter describes each table and how to insert new entries. Chapter 4 of that guide describes how to modify existing entries.*

Configuring Processing Attributes

As with the Quality workstation, you can configure several processing parameters using the *control keys* provided in the e*Index Administrator. With the control keys, you can set matching thresholds, define the length of the UID, specify a checksum length, and so on. Additional control keys that only control GUI attributes are described earlier in this chapter under "Parameter Configuration". The following list provides a brief explanation of the control keys that are available for processing configuration. For detailed information about each control key and how to modify their values, see Chapter 5 of the *e*Index Administrator User's Guide*.

Tip: *You will use three control keys (DUPTHRES, MATCHTHRES, and THRESHOLD) to specify the matching thresholds for e*Index. They define the minimum matching weight at which two records are considered potential duplicates, the minimum weight at which two records are automatically merged, and the minimum weight at which a record is considered a possible match for a search. Configuring these thresholds is an on-going process, and should begin when you perform the initial data analysis. Once the Vality rule set files are fully customized, you will be able to more accurately tailor the threshold values. Your SeeBeyond representative can help you fine-tune the matching thresholds.*

Matching Parameters

These parameters specify how records are matched within e*Index, and affect processing from both the e*Index GUIs and e*Ways. The appropriate values for these parameters will become more clear as you continue to analyze the data, test the initial load, and review the initial load reports.

- **1XACTMTCH**
  This control key allows you to specify whether all records that match your search criteria and have weights above the match threshold (see MATCHTHRES) will be treated as potential duplicates or if the record with the highest matching weight will be automatically merged with the new record.
- **DUPTHRES**
  This control key allows you to specify a minimum matching weight for which a profile is considered a potential duplicate of the profile being added.
- **MATCHTHRES**
  Use this control key to specify the matching probability weight at which e*Index automatically merges a new profile with an existing potential duplicate profile.
- **MAXPROB**
  This control key allows you to specify the maximum matching probability weight to be used by e*Index.
- **SMEFACREP**
  You can enable or disable same facility potential duplicate reporting using this control key.

Data Processing Parameters

The following parameters affect how data is processed and stored in the e*Index database. You can also specify the location to which error information is written.

- **ASSMTCH**
  Use this control key to specify whether all assumed matches are written to the *ui_assumed_match* table.
- **BLNKONUPDT (e*Ways only)**
  Use this control key to specify whether to treat a blank field as null when performing an update from the sending e*Way.
- **CKSUMLEN**
  Use this control key to specify a checksum length to be appended to each UID. Carefully review the information contained in chapter 5 of the *e*Index Administrator User's Guide* under "CKSUMLEN" and "UIDLENGTH" if you will use the checksum functionality.

- **COUNTRY**
  Use this control key to specify the country format for the GUIs and database tables.
- **DEBUGLVL**
  Use this control key to specify whether debug messages for e*Index should be displayed on screen or printed to a file (for UNIX systems only).
- **DUPCHK**
  This control key allows you to specify whether to automatically perform a check for potential duplicates each time a key field in a profile is updated.
- **UIDLENGTH**
  This control key allows you to limit the number of digits that e*Index uses when it creates a new UID. If you change the UID length, you must also change the value of the seq_no column of the u*i_seq_no* table where table_name equals *ui_person*.

Matching Algorithm Customizations

Designing the customizations to the matching algorithm from Vality should be performed by someone with a solid knowledge of the Vality Real-time environment and the available Vality functions. e*Index provides a standard set of rule set files that define specific business processing rules for the match process. Once your data is analyzed, the requirements for these processing rules can be defined. You may want to customize two files in particular. The file controlling most of the processing logic is called the rules file, and is named **UI.RUL**. The other file is the nickname table, **UI.TBL**. This file contains a list of first names and nicknames along with a standardized name for each. You may want to add several entries to this table after analyzing your data. The data string that is sent to Vality for matching is defined by the candidate selection query (see "Configurable Query Customizations" below). This string must match the data string defined in the **UI.DCT** file.

Frequency Files

There is an optional file, a frequency file, that can be added to the rule set. This file provides a frequency analysis of first and last names in your legacy data, and influences probabilities used during the match process based on each name's frequency. This file is not required. Your SeeBeyond representative can help you determine if this file would be useful in your implementation.

Loading the Customizations

Once the files are modified, they must be loaded into the e*Index database using the Rule Set Maintenance function of the e*Index Administrator. For more information about rule set files and the functions provided for maintaining rule sets, see chapter 6 of the *e*Index Administrator User's Guide*.

Configurable Query Customizations

If you want to change the search criteria used when e*Index retrieves a set of records to be matched against an incoming record, you need to modify the candidate selection query using the Configurable Query function of the e*Index Administrator. This is a complex task, and should only be done if it is necessary to accomplish the best searching and matching configuration for your data. By default, the query searches for possible matches using the following criteria in the order listed.

- Phonetic first name, date of birth, and gender
- Phonetic last name and phonetic first name
- Phonetic last name and mother's maiden name
- Social security number

Customizing the candidate selection query is entirely optional. With no modification, candidates are selected as described in step 3 of "About Inbound Event Processing Logic" in chapter 2 of the *e*Index Global Identifier Technical Reference*. If your data contains fields other than those listed above that provide a more reliable comparison, you may want to modify the query to use those fields instead.

In designing the configurable query, take into account that if you modify the candidate selection query, you must also modify the Vality rule set file **UI.DCT** accordingly. Make sure that any fields you select for matching in the query are also used for matching in the rule set files (**UI.DCT** and **UI.RUL**).

Initial Data Load Design

Designing the initial load process is a process of continual refinement. You will use test runs of the initial load on sample data to ensure that the matching thresholds and Vality INTEGRITY matching algorithm are configured correctly. Before you begin your design of the initial load, review the data analysis, which should include information about the layout, default data values, data available for the initial load, and required code table values. Once you perform a test run on the extracted and cleansed data, you can analyze the record count of successfully loaded records and the timing of the procedure. You can customize the production reports and run them against the initial load data to help fine-tune the matching algorithm and the matching thresholds. By the end of the development and testing phase, you should have specifications detailing the process of performing the production initial load procedure.

Appendix A provides a sample data mapping table for HL7 messages for the initial load and e*Ways. Appendix B provides a sample initial load processing log.

Customizing Standard Reports

For Oracle installations, e*Index provides several data analysis reports, called *production reports*, in the PL SQL query language. You can modify the appearance and content of these reports as required for your data analysis requirements. Use production reports to analyze and monitor data and transactions in the e*Index database once the system has gone into production. You can also customize these reports in order to analyze the data loaded into the database through the initial load process. For some reports, the required customizations may be as simple as hard coding the facility code for the facility on which you want to report. These reports should be reviewed and modified by someone with a strong knowledge of PL SQL. For more information about the reports provided, their content, and customizing the reports, see chapter 3 of *Working with Reports for e*Index Global Identifier*.

Designing e*Gate Components

Overview

This section of the chapter discusses designing of the components of the e*Index system that reside in the e*Gate Integrator environment**.** Because this part of the design is primarily an e*Gate implementation, the information provided here is only intended to supplement the information contained in Chapter 5, "System Design and Development", of the *e*Xchange eBusiness Integrator Suite Deployment Guide*. The deployment guide provides complete information about designing and developing the e*Gate environment.

Design Steps

There are three basic steps to designing the e*Gate components for your e*Index system:

- Identify external systems
- Configure e*Gate components
- Identify hardware and network connections

Identify External Systems

The first step in designing the e*Gate components is to identify all the external systems that need to share data within the e*Index system. Determine which systems will send data to e*Index, which will receive data from e*Index, and which will both send and receive data. At this stage, you should create a diagram of the systems that will share data in the e*Index system, and connect the systems with arrows indicating the direction of the flow of information. This will define the number of schemas you need to design, and the direction of the e*Ways within each schema. In identifying the external systems, do not include information about the hosts on which the e*Gate components will reside, or about the format of the data being shared. You can use the summary design model you created in the planning stage as a basis for this diagram.

Configuring e*Gate Components

The second step in e*Gate component design is to define a configuration of e*Index components, including e*Way Intelligent Adapters, Business Object Brokers (BOBs), Event Type Definitions (ETDs), Collaboration Rules, and Intelligent Queue (IQ) Managers. You may also need to define customized translations for the data being shared. These components run on the hosts you designate in the following step. For the most efficient design, follow the recommendations for designing e*Gate components in chapter 5 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

Hardware and Network Connections

The third step is to design a configuration of hardware and network connections that allows the external systems to communicate as required. Because the e*Gate Integrator system is designed to run as a distributed system, the only relevant consideration here are the network performance and the demands on each host. The hardware and network configuration you define can be adjusted as needed if system demands change. If you are already using the e*Gate Integrator, you may be able to use your existing hardware and network configuration to support the new components for e*Index and the external systems. If you expect an extremely high volume of Events to be processed into the e*Index database, you might want to consider using a dedicated Participating Host for the e*Index schemas and the e*Way Intelligent Adapter for the database platform you are using.

Designing the e*Gate Components of e*Index

In designing the e*Gate components of the e*Index system, there are several performance considerations to take into account. These considerations are described in chapter 5 of the *e*Xchange eBusiness Integrator Suite Deployment Guide* beginning with "Performance Considerations" and ending with "Performance Summary".

You need to design schemas for both the external systems sharing data with e*Index and for the e*Index database. While the schema components for the external systems connecting to e*Index must be designed separately, SeeBeyond supplies a sample schema that you can use as a template for your production e*Index schema. You can customize this schema using the suggestions from the deployment guide. For detailed information about the standard components of the e*Index sample schema, see the following section, "About e*Index Schema Components".

The *e*Xchange eBusiness Integrator Suite Deployment Guide* provides information about achieving high performance by incorporating certain e*Gate components, such as BOBs or subscriber pooling, into your e*Ways. You can also use multiple schemas or multiple sending e*Ways in the e*Index schema to transfer data through the e*Index database. The number to use depends on the expected volume of Events, the size of each Event, and the processing capabilities of the database and database server.

About e*Index Schema Components

Overview

This section of the chapter describes the components included in the sample e*Index schema, and how to work with these components to create schemas specific to your processing requirements. For information about designing schemas for the external systems that will be sharing data with e*Index, see chapter 5 of the *e*Xchange eBusiness Integrator Suite Deployment Guide***.**

Schema Components Overview

When you install the e*Gate components of e*Index, you can install a sample schema in the e*Gate environment, or you can simply install the default schema files in an installation directory. If you choose not to install the sample schema, the component files (that is, the binary files, **ui-fns.monk**, and **eiEvent.ssc**) are installed in the default schema directory of the e*Gate Registry for use with the e*Index schema. You can use the sample schema or just the schema files as a template for designing your production schemas. The schema includes a bi-directional e*Way that provides access to and from the e*Index database, one *polling* e*Way that reads specific information from the database, and three standard e*Ways. This schema provides the basic components that are required to move and transform the data handled by e*Index and to unit test each e*Way.

Figure 6-2 on the following page illustrates the components of the sample schema. In a production environment, you may not use all components of the schema. Typically, you will use **ewUIDB**, **ewUIPOLL**, and possibly **ewUIPOLLEATER**. Table 6-2 beginning on the following page describes all of the e*Ways of the e*Index schema. You may need to configure some of these e*Ways to meet your processing requirements.

Figure 6-2: e*Index Schema Components

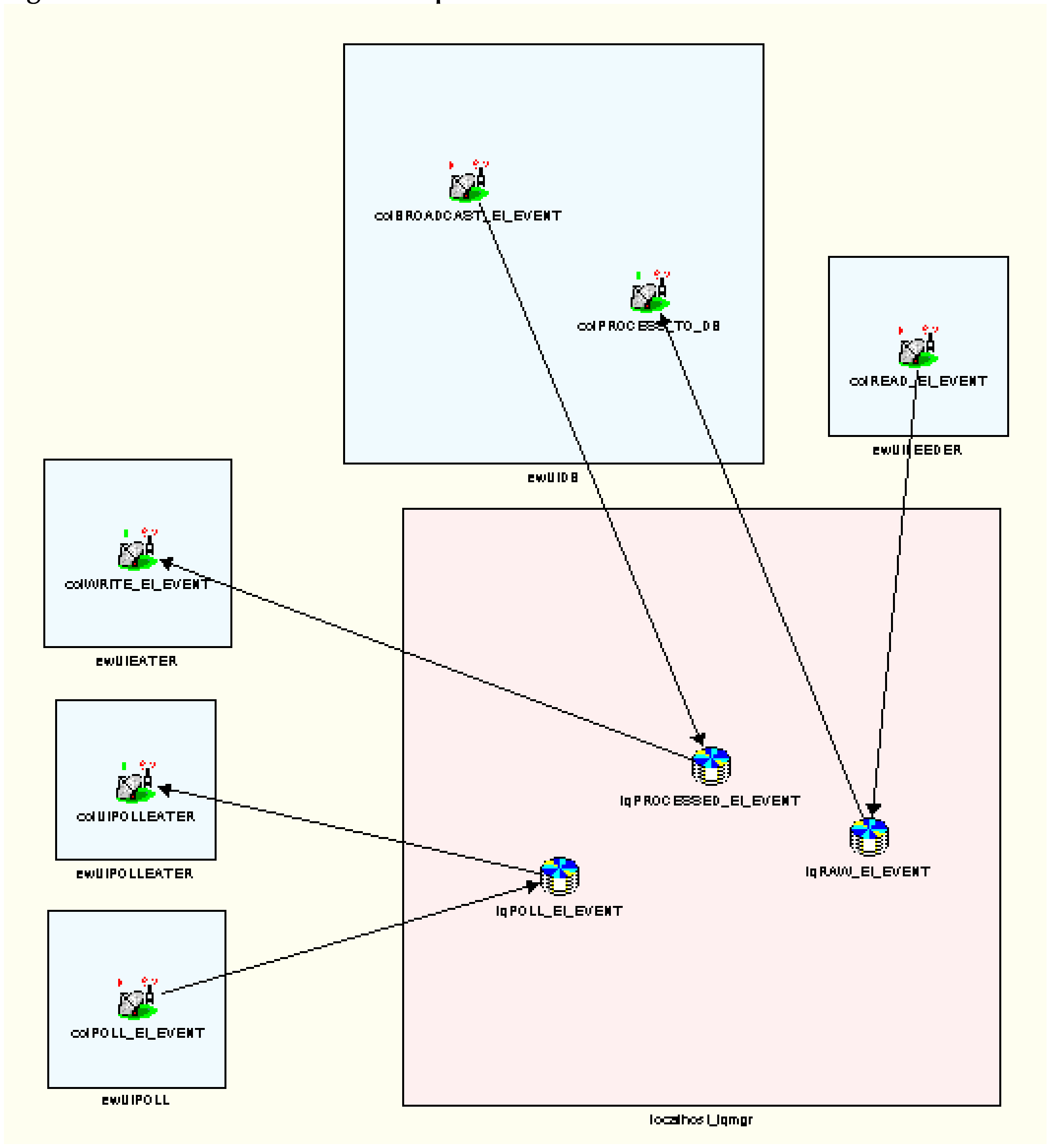


Table 6-2: e*Index Schema e*Ways

e*Way	Description	More Information
ewUIDB	This e*Way, known as the sending e*Way, processes all information coming into the e*Index database. It picks up Events from the iqRAW_EI_EVENT IQ. After the database processes the Events, they are returned to e*Gate through the same e*Way with each record's UID attached. This e*Way publishes to the iqPROCESSED_EI_EVENT IQ.	**Sending e*Way on page 6-26**

e*Way	Description	More Information
ewUIEATER	This e*Way reads the information that is returned from the database through ewUIDB. It picks up Events from the iqPROCESSED_EI_EVENT IQ and prints information to a flat file. This e*Way is used primarily for testing purposes or to remove unneeded messages. This e*Way is also called the eater e*Way.	**Eater e*Way on page 6-35**
ewUIFEEDER	This e*Way reads a flat file into the iqRAW_EI_EVENT IQ so the data can be picked up by the ewUIDB e*Way to be processed into the database. This is primarily used for testing or batch file processing. This e*Way is also called the feeder e*Way.	**Feeder e*Way on page 6-34**
ewUIPOLL	This e*Way, known as the polling e*Way, processes Events from the *ui_msg_header* and *ui_msg_detail* tables of the e*Index database (known as the out queue). These Events typically originate in the GUI, or are triggered by specific transactions. ewUIPOLL writes to the iqPOLL_EI_EVENT IQ.	**Polling e*Way on page 6-31**
ewUIPOLLEATER	This e*Way reads the Events that are processed by the ewUIPOLL e*Way (in the iqPOLL_EI_EVENT IQ). This e*Way is used for testing or in situations where the polling e*Way is not being used. In this case, the polling e*Way removes Events from the out queue, and ewUIPOLLEATER writes the messages to a file, removing them from the IQ.	**Poll Eater e*Way on page 6-36**

Sending e*Way

The sending e*Way processes all Events coming into the e*Index database. At least one sending e*Way is required to process Events, and you can add as many sending e*Ways as needed to provide additional processing capacity when handling a large number of transactions. Make sure you take into account the effect that multiple sending e*Ways will have on the response time of the e*Index database. You can use the sample sending e*Way, ewUIDB, as a template for all of your sending e*Ways.

This e*Way is specially configured to process incoming data using a specific set of API calls along with the matching algorithm logic. You can customize the logic that defines how data is processed through this e*Way using the Monk API provided with e*Index. To change the processing logic, you need to modify the file **uidb.dsc**, described on the following page.

Sending e*Way Components

The sending e*Way is configured to transfer data into and then back out of the e*Index database. It consists of two Collaborations using one Event Type. You can modify the Event Type or create new Event Types if required, keeping in mind the minimum requirements of the database. Modifying Event Types may require that the e*Way Collaboration Service be changed from Pass Through to Monk, and that you create additional processing rules. The standard ETD, **eiEvent.ssc**, defines a message structure to match the e*Index database structure. You can modify this ETD if needed.

- **colPROCESS_TO_DB**
  This Collaboration subscribes to the Event Type **eiEvent** in the **iqRAW_EI_EVENT** IQ and publishes the Event Type **eiEvent**, publishing Events to the database.

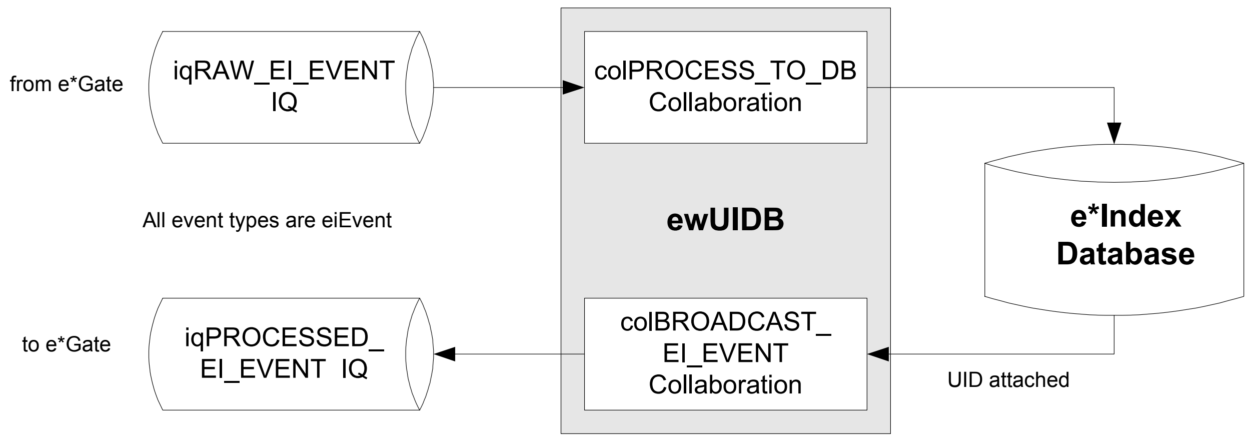
- **colBROADCAST_EI_EVENT**
  This Collaboration subscribes to the Event Type **eiEvent** from the database, and publishes the same Event Type to the **iqPROCESSED_EI_EVENT** IQ once the Event has been processed through e*Index and has been assigned a UID.

- **uidb.dsc**
  This Collaboration Rules script contains the rules that specify how data is processed once it leaves e*Gate and before the database receives the data. The script called the Monk file **ui-process-person.monk** for processing rules. The processing rules contained in both files are defined by Monk functions, and you can modify the files using the Monk APIs provided with e*Index, the Database e*Way, and e*Gate. The lists used by the e*Index Monk APIs are defined in **ui-custom.monk**.

  For more information about how data is processed using **uidb.dsc** and **ui-process-person.monk**, see "About Inbound Event Processing Logic" in chapter 2 of the *e*Index Global Identifier Technical Reference*. For more information about the e*Index Monk APIs you can use in **uidb.dsc**, see chapter 4 of the technical reference.

Figure 6-3 illustrates the components of the sending e*Way.

Figure 6-3: Sending e*Way Components



Configuring the Sending e*Way

The sending e*Way configuration file, **ewUIDB.cfg**, requires minimal customizations. Table 6-3 lists the parameters and default settings for the sending e*Way. For more information about these parameters, see chapter 3 of the *e*Index Global Identifier Technical Reference*.

Table 6-3: Sending e*Way Configuration Settings

Window	Parameter	Default Setting
General Settings	Journal File Name	No default. To create a journal file for the e*Way, enter a file name and path.
	Max Resends Per Message	Default: **1** You can modify the number of times an error message can be resent.
	Max Failed Message	Default: **1024** You can modify the maximum number of error messages the e*Way will allow.
	Forward External Errors	Default: **No** Change this value to **Yes** to queue error messages received from the external that start with **DATAERR** to the configured queue.
Communication Setup	Start Exchange Data Schedule	No default. Since the e*Way is event-driven, this can remain blank.
	Stop Exchange Data Schedule	No default. Since the e*Way is event-driven, this can remain blank.
	Exchange Data Interval	Default: **120** Since the e*Way is event-driven, you do not need to modify this value.
	Down Timeout	Default: **15** Modify this value to change the number of seconds to wait before attempting to reconnect to the database.
	Up Timeout	Default: **15** Modify this value to change the number of seconds to wait between checks to verify the database connection.
	Resend Timeout	Default: **10** Modify this value to change the number of seconds to wait before resending a message that received an error.

Window	Parameter	Default Setting
	Zero Wait Between Successful Exchanges	Default: **No** Since the e*Way is event-driven, you do not need to modify this value.
Monk Configuration	Additional Path	Default: **monk_scripts/ui** ***Note****: If you are using the standard e*Index scripts and database connectivity functions, you do not need to modify any of the Monk parameters. For more information about these parameters, see "Monk Configuration" in chapter 3 and "External Monk API Descriptions" in chapter 4 of the* e*Index Global Identifier Technical Reference.
	Auxiliary Library Directories	Default: **monk_library/dart; monk_library/ui** If you are using additional Monk libraries in this e*Way, add the paths to those libraries.
	Monk Environment Initialization File	Default: **ui-stdver-init** This function is defined in the file **ui-stdver-eway-funcs.monk**, along with most of the following functions.
	Startup Function	Default: **ui-stdver-startup**
	Process Outgoing Message Function	Default: **monk_scripts/ui/uidb.dsc** ***Note:*** *This is the Collaboration Rules script that specifies how data is processed for e*Index.*
	Exchange Data with External Function	Default: **ui-stdver-data-exchg**
	External Connection Establishment Function	Default: **ui-stdver-conn-estab**
	External Connection Verification Function	Default: **ui-stdver-conn-ver**
	External Connection Shutdown Function	Default: **ui-stdver-conn-shutdown**
	Positive Acknowledgement Function	Default: **ui-stdver-pos-ack**
	Negative Acknowledgement Function	Default: **ui-stdver-neg-ack**
	Shutdown Command Notification Function	Default: **ui-stdver-shutdown**

Window	Parameter	Default Setting
Database Setup	Database Type	Default: **Oracle8i** Enter any of the following values: ▪ **Oracle9i** for an Oracle 9*i* database ▪ **Oracle8i** for an Oracle 8*i* database ▪ **ODBC** for a Microsoft SQL Server database ▪ **Sybase11** or **Sybase12** for a Sybase database
	Database Name	No Default. Enter the Oracle SID name of the e*Index database.
	User Name	Default: **UI** Modify this value to the Superuser login ID of the e*Index database.
	Encrypted Password	Default: ** Enter the Superuser password to the e*Index database.

Polling e*Way

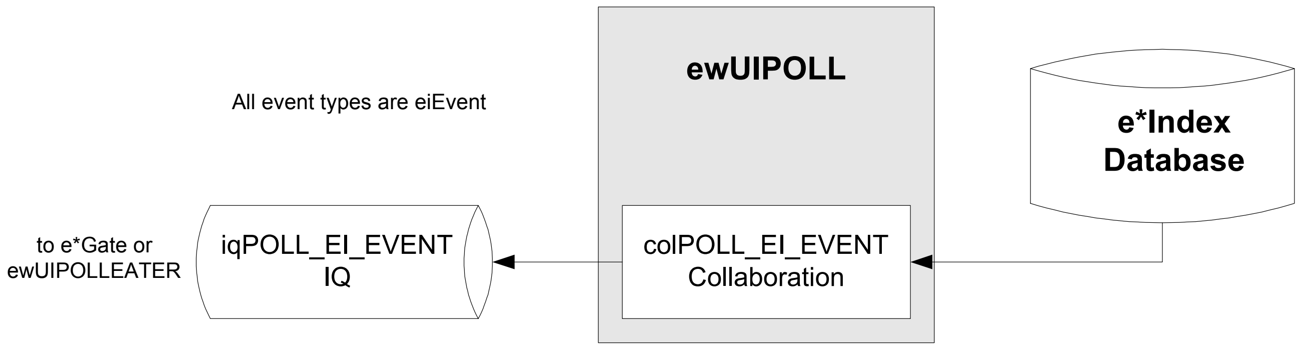
The polling e*Way checks the *ui_msg_header* and *ui_msg_detail* tables of the e*Index database for any outgoing Events and makes that information available to external systems via e*Gate. Events in this table generally originate from the e*Index GUI or are triggered by a specific transaction type, such as a merge transaction. You do not need to use this e*Way to transfer Events to external systems. If you choose not to use the e*Way for this purpose, use the polling e*Way to empty the out queue, and use the poll eater e*Way to remove the Events from the IQ. The *ui_msg_detail* table can grow quite rapidly. If Events are not removed regularly, disk space and performance issues may arise.

Polling e*Way Components

The polling e*Way consists of one Collaboration. You can modify the Event Type or create new Event Types if needed, but doing so may require that the e*Way Collaboration Service be changed from Pass Through to Monk, and that you create additional processing rules. Figure 6-4 on the following page illustrates the components of the polling e*Way.

- **colPOLL_EI_EVENT**
  This Collaboration subscribes to the Event Type **eiEvent** from the database and publishes to the Event Type **eiEvent**, publishing Events to the **iqPOLL_EI_EVENT** IQ, where the appropriate external systems can pick up the Events.

Figure 6-4: Polling e*Way Components


All event types are eiEvent
ewUIPOLL
e*Index Database
to e*Gate or ewUIPOLLEATER
iqPOLL_EI_EVENT IQ
colPOLL_EI_EVENT Collaboration

Configuring the Polling e*Way

The polling e*Way configuration file, **ewUIPOLL.cfg**, requires minimal customizations. Table 6-4 lists the parameters and default settings for the polling e*Way. For more detailed information about these parameters, see chapter 3 of the *e*Index Global Identifier Technical Reference*.

Table 6-4: Polling e*Way Configuration Settings

Window	**Parameter**	**Default Setting**
General Settings	Journal File Name	No default. To create a journal file for the e*Way, enter a file name and path.
	Max Resends Per Message	Default: **5** You can modify the number of times an error message can be resent.
	Max Failed Message	Default: **3** You can modify the maximum number of error messages the e*Way will allow.
	Forward External Errors	Default: **No** Change this value to **Yes** to queue error messages received from the external that start with **DATAERR** to the configured queue.
Communication Setup	Start Exchange Data Schedule	No default. Since the e*Way is event-driven, this can remain blank.
	Stop Exchange Data Schedule	No default. Since the e*Way is event-driven, this can remain blank.
	Exchange Data Interval	Default: **10** Since the e*Way is event-driven, you do not need to modify this value.

Window	Parameter	Default Setting
	Down Timeout	Default: **15** Modify this value to change the number of seconds to wait before attempting to reconnect to the database.
	Up Timeout	Default: **15** Modify this value to change the number of seconds to wait between checks to verify the database connection.
	Resend Timeout	Default: **10** Modify this value to change the number of seconds to wait before resending a message that received an error.
	Zero Wait Between Successful Exchanges	Default: **No** Since the e*Way is event-driven, you do not need to modify this value.
Monk Configuration	Additional Path	Default: **monk_scripts/ui** ***Note**: If you are using the standard e*Index scripts and database connectivity functions, you do not need to modify any of the Monk parameters. For more information about these parameters, see "Monk Configuration" in chapters 3 and "External Monk API Descriptions" in chapter 4 of the* e*Index Global Identifier Technical Reference.
	Auxiliary Library Directories	Default: **monk_library/dart; monk_library/ui**
	Monk Environment Initialization File	Default: **ui-stdver-init** ***Note:** This function is located in the file **ui_stdver_eway_funcs.monk**, along with the following functions. You can modify the functions as needed to meet your processing requirements for the data from the* ui-out-queue *table.*
	Startup Function	Default: **ui-poll-startup**
	Process Outgoing Message Function	Default: **ui-stdver-proc-outgoing**
	Exchange Data with External Function	Default: **ui-poll**
	External Connection Establishment Function	Default: **ui-stdver-conn-estab**
	External Connection Verification Function	Default: **ui-stdver-conn-ver**
	External Connection Shutdown Function	Default: **ui-stdver-conn-shutdown**

Window	Parameter	Default Setting
	Positive Acknowledgement Function	Default: **ui-poll-pos-ack**
	Negative Acknowledgement Function	Default: **ui-poll-neg-ack**
	Shutdown Command Notification Function	Default: **ui-stdver-shutdown**
Database Setup	Database Type	Default: **Oracle8i** Enter any of the following values: ▪ **Oracle9i** for an Oracle 9i database ▪ **Oracle8i** for an Oracle 8i database ▪ **ODBC** for a Microsoft SQL Server database ▪ **Sybase11** or **Sybase12** for a Sybase database
	Database Name	No default. Enter the Oracle SID name of the e*Index database.
	User Name	Default: **UI** Modify this value to the Superuser login ID of the e*Index database.
	Encrypted Password	Default: ** Enter the Superuser password to the e*Index database.

Feeder e*Way

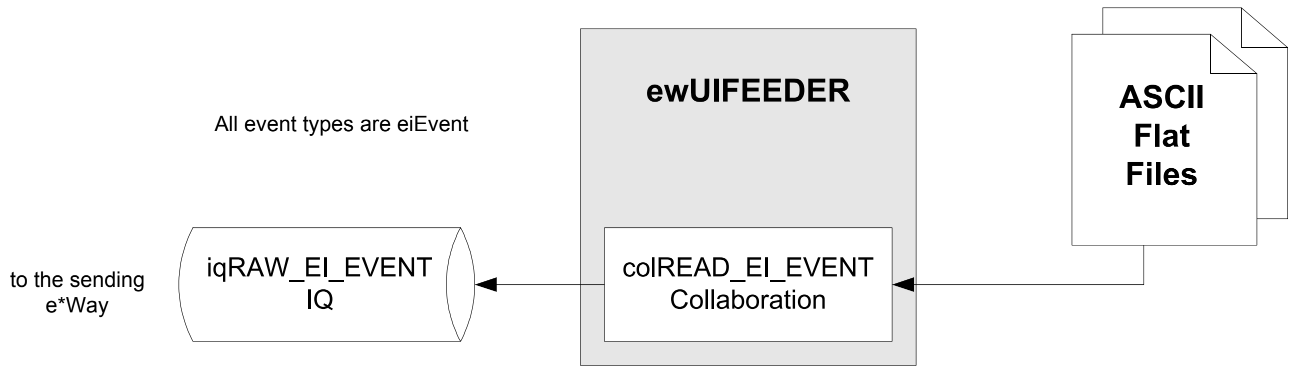
The feeder e*Way is a standard inbound **stcewfile** e*Way that can be used to test how data is processed into the e*Index database or to load batch files into the e*Index database. This e*Way reads data files in the specified directory, and publishes the Events to the **iqRAW_EI_EVENT** IQ. These Events can then be picked up by the sending e*Way and processed into the database.

Feeder e*Way Components

The feeder e*Way consists of one Collaboration. The Event Type should be the same Event Type that is subscribed to by the sending e*Way since no data transformation is performed by this e*Way. Figure 6-5 on the following page illustrates the components of the feeder e*Way.

- **colREAD_EI_EVENT**
  This Collaboration subscribes to the Event Type **eiEvent** from a flat file and publishes to the Event Type **eiEvent**, publishing Events to the **iqRAW_EI_EVENT** IQ, where the sending e*Way can pick them up.

Figure 6-5: Feeder e*Ways Components



Configuring the Feeder e*Way

The feeder e*Way will function properly with the default configuration settings. However, if required, you can change the configuration parameters. You do not need to modify the Outbound (send) Settings since they are ignored in this e*Way. Because the feeder e*Way is a standard e*Way that merely copies Events to the IQ, a detailed discussion of the configuration is not included here. For more information about configuring standard e*Ways, see the *Standard e*Way Intelligent Adapters User's Guide*.

Eater e*Way

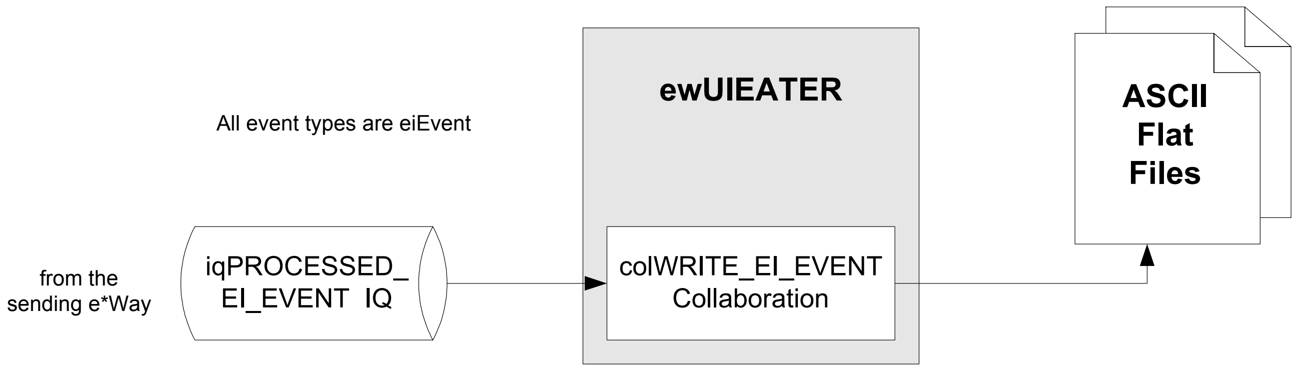
The eater e*Way is a standard outbound **stcewfile** e*Way that can be used to test how data is processed through the sending e*Way. This e*Way reads Events in the **iqPROCESSED_EI_EVENT** IQ, which were processed by the sending e*Way, and appends the Events to a file in the specified directory.

Eater e*Way Components

The eater e*Way consists of one Collaboration. The Event Type should be the same Event Type that is published by the sending e*Way. Figure 6-6 on the following page illustrates the components of the eater e*Way.

- **colWRITE_EI_EVENT**
  This Collaboration subscribes to the Event Type **eiEvent** from the **iqPROCESSED_EI_EVENT** IQ (from the sending e*Way) and publishes the Event Type **eiEvent** to a flat file.

Figure 6-6: Eater e*Way Components



Configuring the Eater e*Way

The eater e*Way will function properly with the default configuration settings. However, if required, you can change the configuration parameters. You do not need to modify the Inbound (send) Settings since they are ignored in this e*Way. Because the eater e*Way is a standard e*Way that merely copies the Events to a flat file, a detailed discussion of the configuration is not included here. For more information about configuring standard e*Ways, see the *Standard e*Way Intelligent Adapters User's Guide*.

Poll Eater e*Way

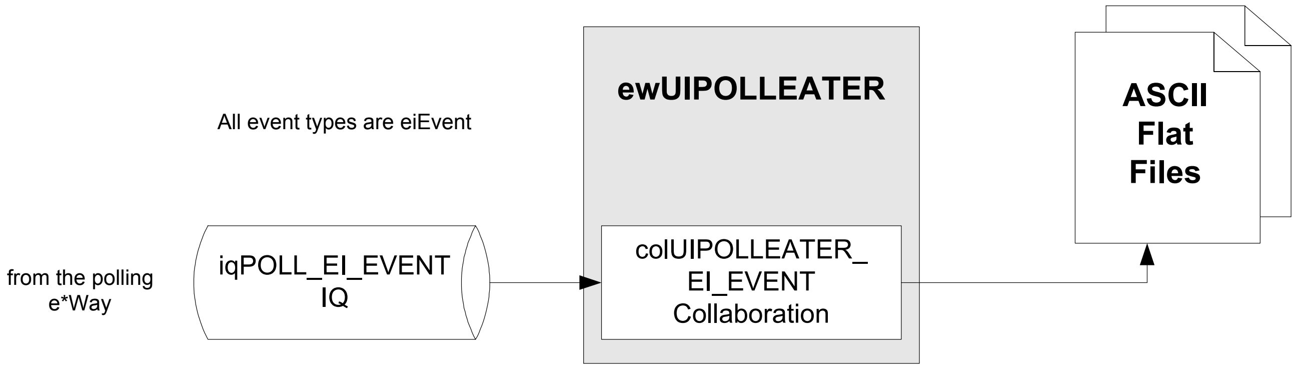
The poll eater e*Way is a standard outbound **stcewfile** e*Way that can be used to test how data is processed through the polling e*Way, and to help empty the out queue from the database if you are not using the polling functionality. This e*Way reads Events in the **iqPOLL_EI_EVENT** IQ, which were processed by the polling e*Way, and appends the Events to a file in the specified directory. If you use this e*Way to empty the out queue, be sure to archive the output file regularly because the file may grow rapidly, depending on the number of transactions being written to the *ui_msg_detail* and *ui_msg_header* tables.

Poll Eater e*Way Components

The poll eater e*Way consists of one Collaboration. The Event Type should be the same Event Type that is published by the polling e*Way. Figure 6-7 on the following page illustrates the components of the poll eater e*Way.

- **colUIPOLLEATER_EI_EVENT**
  This Collaboration subscribes to the Event Type **eiEvent** from the **iqPOLL_EI_EVENT** (from the polling e*Way) and publishes the Event Type **eiEvent** to a flat file in the specified directory.

Figure 6-7: Poll Eater e*Way Components



Configuring the Poll Eater e*Way

The poll eater e*Way will function properly with the default configuration settings. However, if required, you can change the configuration parameters. You do not need to modify the Inbound (send) Settings since they are ignored in this e*Way. Because the poll eater e*Way is a standard e*Way that merely copies the Events to a flat file, a detailed discussion of the configuration is not included here. For more information about configuring standard e*Ways, see the *Standard e*Way Intelligent Adapters User's Guide*.

Developing your e*Index System

Overview

Once you create a basic system design, along with the design specifications, build on that design by developing your complete e*Index system. You will use a combination of tools to create and setup all the necessary system components. These tools include the e*Gate GUIs, e*Index GUIs, and database installation scripts.

System Development Considerations

Before you begin the development phase, make sure that the design specifications have been documented in detail, and the programmers and implementers have been assigned specific development tasks. Also make sure that the development and test environments are installed, configured appropriately, and tested. You can use several SeeBeyond tools to develop and unit test the e*Index system, including the e*Gate GUIs and the e*Index GUIs. Begin the development phase by installing all the required SeeBeyond software in the development and test environments. Once this is in place, you are ready to begin developing the e*Index and e*Gate systems.

Using e*Gate GUIs for Development

Use the e*Gate GUIs to configure the schema components for e*Index and the external systems with which e*Index will share data. e*Gate's system development and setup features are accessed via the Enterprise Manager GUI. In addition to the e*Gate Enterprise Manager, use the following GUIs for setting up, configuring, and editing the e*Gate components. Certain e*Index schema components, such as the Collaboration Rules file (**uidb.dsc**) and the ETD (**eiEvent.ssc**), cannot be modified in the e*Gate editors. These files need to be modified in a text editor and then promoted to runtime using the e*Gate Enterprise Manager.

- Event Type Definition (ETD) Editor
- Collaboration Rules Editor
- Collaboration-ID Rules Editor (for backwards-compatibility with e*Gate Version 3.6 only)
- e*Way Editor

For more information about working with e*Gate GUIs, see the *e*Gate Integrator User's Guide*.

e*Gate Component Development

Developing the e*Gate Components for e*Index requires the following steps. See Chapter 5 of the *e*Xchange eBusiness Integrator Suite Deployment Guide* for detailed information about performing these tasks.

- Creating Schemas
- Creating Event Types and their definitions (ETDs)
- Creating Collaboration Rules and scripts
- Adding e*Ways
- Adding Intelligent Queues
- Adding Collaborations
- Reviewing and testing the system

Note: *A sample e*Index schema can be installed when you install the schema components for e*Index. If you install this schema, you can customize it for your data transfer and processing needs, which means you may not need to perform all of the above tasks for the e*Index schemas. You will need to perform these steps for the schemas that transfer data to and from the external systems linked to e*Index.*

e*Index GUI Development

The e*Index GUIs must be installed with the appropriate hardware and networking configurations, and must be configured according to the design documents. You can install the GUIs as soon as you have the specified hardware, software, networking, and database configuration.

Database Connectivity

Before you install the GUIs, make sure you have the database connectivity configured correctly. For an Oracle database, this consists of modifying **tnsnames.ora**. For SQL Server, you must create an ODBC data source for the database. For a Sybase database, you must modify the **sql.ini** file. Chapters 3, 4, and 5 of the *e*Index Global Identifier Installation Guide* provide procedures for performing these tasks.

GUI Initialization and Setup

Once you install the GUIs, modify the initialization file, **stc_ua.ini**, to make sure it points to the correct database. Once you install the e*Index database, you can use the e*Index Security GUI to set up security and access permissions for e*Index. Use the e*Index Administrator GUI to customize the GUIs, data processing parameters, code table data, configurable query, matching algorithm logic, and address-parsing rules.

Database Development

Before installing the database, make sure a complete sizing and distribution analysis was performed, and that the specifications are thoroughly documented. Installing a new database can be a complicated task, and certain attributes cannot be modified after the database is created. Carefully follow the instructions for installing an e*Index database contained in your *e*Index Global Identifier Installation Guide*.

The tasks you need to complete to install and configure the database include:

- Install the database.
    - Make sure to customize sizing parameters in the database creation scripts according to your sizing and distribution specifications.
    - Modify the paths in **defs.sql** and **initSID.ora** according to your specifications for a distributed database.
- Modify data processing parameters using the Control Key Table Maintenance function of e*Index Administrator.
- Insert or modify code table values using the Common Table Maintenance and the Table Maintenance functions of e*Index Administrator.
- Customize the matching algorithm rule set files and address-parsing rules.
- Modify the configurable query.

Initial Data Load

During the development phase of the e*Index implementation, you should perform test runs of the initial data load on a subset of the legacy data to be loaded. You will likely perform multiple test runs of the data load to ensure that the data is in good condition, and that it is being processed into the e*Index database correctly. Multiple runs will also help you tailor the initial load validations until you achieve the desired results. Use the production reports along with additional data analysis information to determine data issues that need to be addressed and to fine tune the matching thresholds. Throughout the development cycle, you will continue to fine-tune the Vality rule set and the matching weights to obtain the best match results for your data.

Once you have completed the MPI clean up and customized the rule set files and the matching thresholds to the desired configuration, perform final test run of the initial load to verify the matching results, the customized algorithm, and the matching thresholds. This test run will help you with scheduling the move to production because the initial load must be performed before you start to process live data through the e*Index schema.

System Development Checklists

Following is a list of items to verify during the development process. Make sure to complete all tasks before beginning integration and acceptance testing.

e*Index GUIs

- Verify hardware and software configuration for the development and test environments.
- Verify networking components for the development and test environments.
- Install the appropriate database client software and configure the database connection.
- Install the e*Index GUIs according to the instructions in chapter 6 of the *e*Index Global Identifier Installation Guide.*
- Modify the initialization file **stc_ua.ini** so it points to the appropriate databases on the database platform of the e*Index database.

***Note:** The remaining steps must be performed after the e*Index database is installed.*

- Use the Control Key Maintenance function of e*Index Administrator to modify the system parameters that only affect the e*Index GUI.
  - ALLOWNUM
  - ALSRCHLMT
  - ATSRCHLMT
  - AUDITONOFF
  - DATEFRMT
  - DEBUGSQL
  - DEMOSRCLMT
  - DOBYYREQ
  - ENCOUNTER
  - ENDTIME
  - EXTNSVSRCH
  - LNEXCTRSRCH
  - MIXEDCASE
  - PDSRCHLMT
  - SEEDEACTIV
  - SEEMERGED
  - SEEMSGCODE
  - SHORTID

- SHOWMN
- SMARTCARD
- SNDXSRCH
- SRCHDOB
- STARTTIME
- THRESHOLD
- TITLEAVL
- UVAUDITLOG

■ Set up the field display configuration by specifying displayed fields, required fields, and field label names.

■ Customize tabbed page labels and search type labels.

■ Define field formats for the Social Security Number, Phone, and Postal Code fields.

■ Customize the configurable query for phonetic searches (make sure the fields you use in the query appear on the Searchj window in the Demographic section).

■ Set up security using the e*Index Security GUI.

■ Configure security system parameters using the e*Index Security GUI.

- DAYSPASEXP
- MINPWDLEN
- IDLETIMER
- PASSHIST

■ Unit test connections to the database, e*Index GUI transactions, and the security configuration

e*Index Database

Important! *Before you begin the database installation, make sure an analysis was performed on disk space requirements, tablespace sizing, and distribution requirements. You will need to specify certain sizing and distribution parameters before you create the database.*

■ Verify hardware and software configuration for the development and test environments.

■ Verify networking components for the development and test environments.

■ Install database server software.

- Install the e*Index database installation files as described in chapter 3, 4, or 5 of the *e*Index Global Identifier Installation Guide*, depending on the database platform used. Use the sizing, distribution, and disk space analyses to modify the database installation files.
- Create or modify code table data using e*Index Administrator.
    - Address Types
    - Citizenships
    - Countries
    - Departments
    - DORs (districts of residence)
    - Driver License Issuers
    - Ethnicities
    - Events (these codes should not be changed)
    - Event Notifications
    - Genders
    - Languages
    - Marital Statuses
    - Nationalities
    - Non-unique ID Types
    - Application Messages (these codes should not be changed)
    - Person CategoriesPhone Types
    - Predefined Messages
    - Races
    - Regions
    - Religions
    - Source Applications
    - States
    - Statuses (these codes should not be changed)
    - Suffixes
    - Systems
    - Titles
    - Veteran Statuses
    - VIP Statuses (these codes should not be changed)
    - Zip Codes
- Using the Control Key Maintenance function of e*Index Administrator, modify the control keys that affect how data is processed through e*Index.
    - 1XACTMTCH

- CKSUMLEN
- COUNTRY
- DUPTHRES
- MATCHTHRES
- MAXPROB
- SMEFACREP
- ASSMTCH
- BLNKONUPDT
- DEBUGLVL
- DUPCHK
- UIDLENGTH

Note: *If you change the UID length, be sure to change the corresponding value specified for* ui_person *in the* ui_seq_no *table.*

- Make sure an MPI scan was performed, and the data clean up is in process.
- Build analysis reports, and customize the standard e*Index production reports. Create any new reports needed.
- Customize the Vality rule set files and matching thresholds based on the data analysis performed in the analysis and planning phases. Also customize the Vality address-parsing rules if needed.
- Customize the candidate selection query (configurable query).
- Perform an initial data load on a subset of the legacy data to be loaded and review the conversion reports.
- Fine tune the matching thresholds and the Vality rule set files based on the results of the initial load.
- Perform a test run of the initial load on the full set of legacy data to be loaded.
- Perform any required unit testing.

e*Index Schema Components

- Verify hardware and software configuration for the development and test environments.
- Verify networking components for the development and test environments.

- Create the schemas for data being transferred between external systems and e*Gate. Make sure to map the processing codes used by the external systems to those used by e*Index (see "Designing Code Tables" on page 6-17 for more information).
  - Develop each e*Way
  - Configure each e*Way
  - Define Event Types and ETDs
  - Define translations if necessary
  - Map processing codes if necessary
- Install the e*Index sample schema or only the binary schema components (according to the instructions in chapter 2 of the *e*Index Global Identifier Installation Guide*).
- Modify the sample schema, or create a new e*Index schema. Be sure to configure database connectivity information.
  - Sending e*Way
  - Polling e*Way
  - Poll Eater e*Way (optional)
  - Customize the Collaboration Script (**uidb.dsc**).
  - Customize the Event Type Definition (**eiEvent.ssc**).
  - Modify or add Monk functions to **ui-fns.monk**.
  - Customize the Monk lists defined in **ui-custom.monk**. You may need to use functions from **ui-fns.monk** for some customizations.
  - If needed, customize the initialization and error handling functions defined in **ui-stdver-eway-funcs.monk**.
- Unit test the e*Index e*Ways using the standard "eater" and "feeder" e*Ways provided in the e*Index schema.
- Unit test the e*Ways for external systems.

Chapter 7

Testing and Training

About this Chapter

Overview

This chapter provides the information you need to test your e*Index system before moving to production. It also discusses training the appropriate personnel on the e*Index system.

The following diagram illustrates the contents of each major topic in this chapter.

Application Testing	Learn about the different types of tests that must be performed on the e*Index system
Unit Testing	Learn about testing each component individually
Integration Testing	Learn about testing a few components together
Acceptance Testing	Learn about testing the entire e*Index system prior to moving to production
Training	Learn about the training that is required prior to moving to production

About Application Testing

Overview

This section of the chapter provides an overview of testing the e*Index GUIs, database, processing configuration, and schemas. It is crucial to fully test the system prior to the transition from a test environment to a production environment. SeeBeyond recommends the following types of testing.

- Unit testing (testing of individual components)
- Integration testing (testing of groups of components together)
- Acceptance testing (testing of completed system)

The first two types of testing are generally performed in the development phase of the implementation, and acceptance testing is performed as a final check before moving into production. Figure 7-1 on page 7-5 provides a diagram of the steps required during the testing and end-user training phase.

Testing Methodology

The method used for testing an e*Index implementation varies from site to site, depending on the specifications of each system. However, certain methodologies apply to all system testing. In general, you should test the individual parts of the system before testing the entire system. Also, test individual components and blocks of code in isolation before testing them in a broader context.

The testing process involves identifying test conditions and scenarios that cover all possible combinations of transactions that may be processed through e*Index or be initiated by e*Index. Once testing has begun, the results need to be analyzed. Track any problems and report them to the appropriate personnel so they can apply the appropriate fixes. Once a problem is fixed, the affected items must be retested.

Testing Guidelines

Testing the e*Index system should follow very specific guidelines to ensure that all possible scenarios are covered, and that any issues found during testing are fixed and retested. The following is a list of guidelines to use when creating and executing the test plans.

- Verify that the test system is operational, and is identical to the production system.
- Provide sufficient data to perform each test.

- Determine the lead-time to test all conditions.
- Specify the order in which the tester should perform the tests (if applicable).
- Complete incident sheets for each error found during testing. Provide complete information, including screen prints if applicable.
- Once a problem is fixed, perform additional tests as required.
- Once the fix is confirmed through testing, make sure to document that the issue is closed.

The Test Plan

The high-level test plan was developed during the planning phase of the implementation using the information obtained from your analysis. This test plan should specify how to test the system and should also define the requirements the system must meet before moving the system to a production environment. The test plan is further refined in the design phase of the implementation to encompass the specific components created for the implementation. The test requirements specification should outline the procedure to use for testing.

The test plan must include:

- **Data Types**
  The test plan specifies the type of data to use for each test scenario, including unit tests, integration tests, and acceptance tests. The data types should be typical of the data that will be processed through the e*Index system. You may want to use some of your existing data for testing purposes. Use a wide variety of data to cover all possible types of scenarios, and be sure to include some "bad" data to test error-handling capabilities.
- **Expected Output**
  Include specific information about the expected results of the e*Index system in the test plan, including error handling requirements, transaction processing speed, correct routing information, correct data transformation, accurate matching and match weights, accurate address-parsing, GUI appearance, and any other requirements you have identified.
- **Implementation Testers**
  The person responsible for each test depends on the type of test. Typically, the developers perform the unit and integration testing for the specific components they developed. The Project Manager or Technical Lead is responsible for testing the entire system. Perform acceptance testing in conjunction with the personnel for whom the system was created. This group could include records personnel, system administrators, and so on.

Testing Environment

The required e*Index test environment varies depending on the component being tested and the type of testing being performed. The test environment is designed during the planning and design phases of the implementation.

- **GUI testing requirements**
  For unit and integration testing of the GUI, you must have the e*Index database installed and configured, and the GUI must be installed and configured on a client workstation for each tester. The proper network connectivity must exist on each machine, and the database client software (Oracle, Sybase, or SQL Server) must be installed and configured on each client workstation. For Oracle, the **tnsnames.ora** file must have a stanza for the database on which you are testing. For SQL Server, an ODBC data source must be configured for the test database. For Sybase, the **sql.ini** file must be configured for the e*Index database. In addition, the appropriate Database e*Way (Oracle, Sybase, or ODBC) must be installed on the machine running the e*Index schemas.

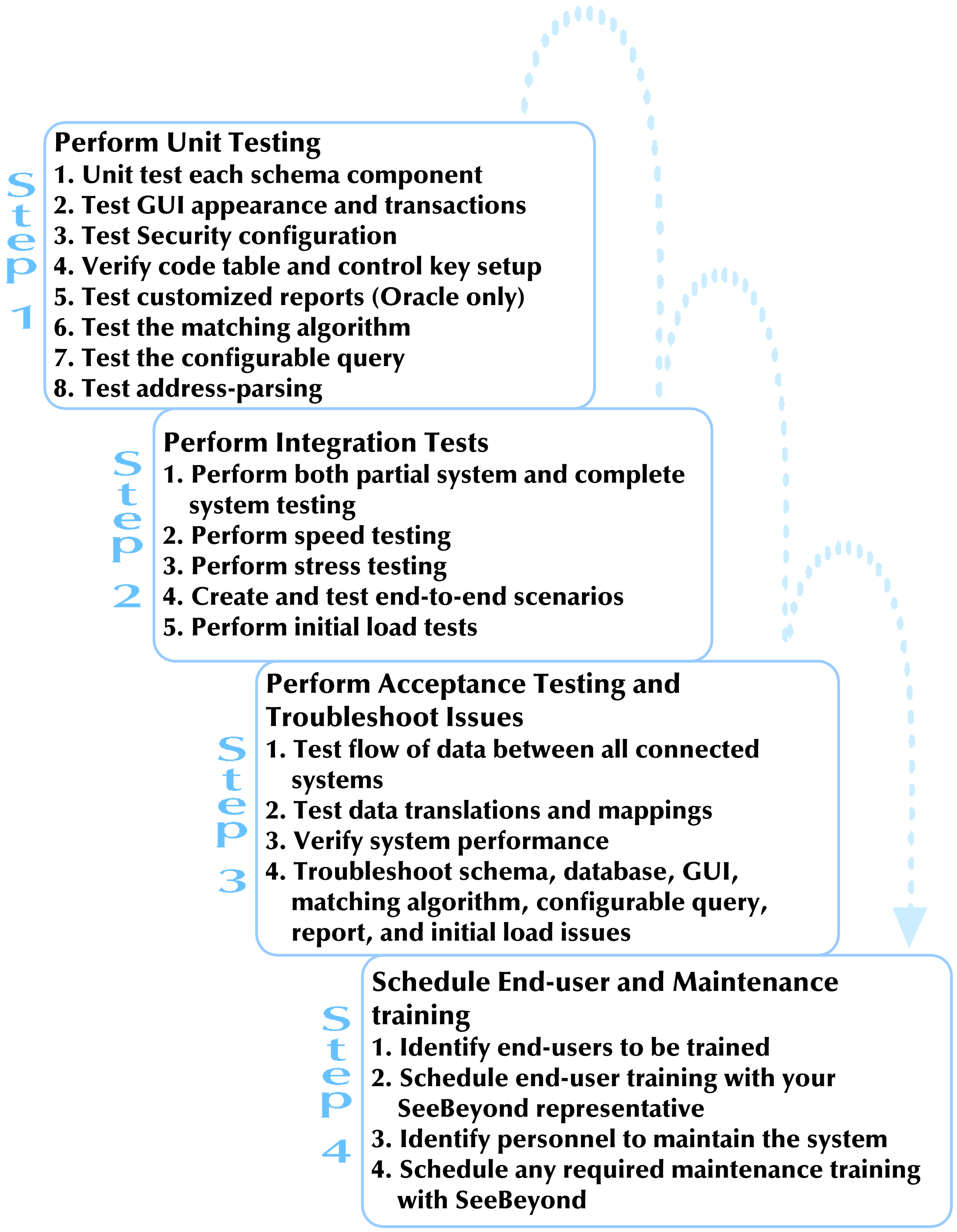
- **Interface testing requirements**
  For unit and integration testing of the interfaces, you must have the e*Index database installed and configured, and you must have an e*Gate server from which you will run the e*Ways. Proper network connectivity must be in place for each machine, and the database client software must be installed on the e*Gate server from which you will run the e*Ways. Oracle, Sybase, or Microsoft SQL Server client must be configured as described above for the GUI testing requirements.

- **Initial load and matching algorithm testing requirements**
  For testing the initial load and matching algorithm, you must have an e*Index database installed and configured according to the production specifications. You can run both the initial load program and the reports from the database server. For testing the matching algorithm, you may also want to run some data into the database through the e*Index e*Way and enter some data through the e*Index GUI. Note that in testing the matching algorithm, you should also test the configurable query to be sure the candidate selection pools contain accurate record sets.

- **Acceptance testing**
  For the final acceptance testing, you must have a complete e*Index system installed, with all of the schema components for each interface in place. You must have a database server with the e*Index database installed and configured, e*Gate server(s) with the schemas for the external systems and e*Index, and client workstations with the e*Index GUIs installed. All machines must have the proper network connectivity configured. The client workstations and the e*Gate server on which the e*Index schemas are located must have the appropriate database client software installed.

  In addition, the Vality rule set files must be configured as designed for the production system and loaded into the database, the matching thresholds must be set at the values specified for the production system, and the configurable query must be customized as defined for the production system. Make sure that all configurable attributes of the system, including control keys, display configuration, and country-specific options, are customized as defined for the production system.

Figure 7-1: Testing and End-User Training Phase Steps


Step 1
Perform Unit Testing
1. Unit test each schema component
2. Test GUI appearance and transactions
3. Test Security configuration
4. Verify code table and control key setup
5. Test customized reports (Oracle only)
6. Test the matching algorithm
7. Test the configurable query
8. Test address-parsing
Step 2
Perform Integration Tests
1. Perform both partial system and complete system testing
2. Perform speed testing
3. Perform stress testing
4. Create and test end-to-end scenarios
5. Perform initial load tests
Step 3
Perform Acceptance Testing and Troubleshoot Issues
1. Test flow of data between all connected systems
2. Test data translations and mappings
3. Verify system performance
4. Troubleshoot schema, database, GUI, matching algorithm, configurable query, report, and initial load issues
Step 4
Schedule End-user and Maintenance training
1. Identify end-users to be trained
2. Schedule end-user training with your SeeBeyond representative
3. Identify personnel to maintain the system
4. Schedule any required maintenance training with SeeBeyond

Unit Testing

Overview

The developers responsible for creating each component perform unit testing as part of the development phase. If the functional, technical, or test plan specifications provide a procedure for testing a particular component, the developers must follow this procedure. If the specifications do not provide a test plan for a specific component, at a minimum the test should verify that the component performs as outlined in the functional and technical specifications.

The tools used to test each component are different depending on its purpose within the system. The methods employed vary, depending on the component being tested. Specific objectives of unit testing include:

- Verify the application meets the functional requirements and business requirements as outlined in the specifications.
- Verify that the correct fields appear on the GUI, with the correct field labels, tab labels, and search type labels.
- Ensure that GUI transactions are processed as required.
- Verify that security is configured correctly.
- Verify that all required code table data is added to the database and that the control keys are configured correctly.
- Verify that the code table elements (such as religion, gender, marital status, and so on) are mapped correctly from incoming Events.
- Ensure the reports are customized for your reporting requirements and any additional required reports have been developed (Oracle only).
- Ensure matching and weighting perform as expected, including the configurable query.
- Verify that addresses are being parsed accurately.
- Ensure all components of the schemas created for both external systems and the e*Index database operate as expected.
- Ensure the operational reliability of the application, interfaces, database, and hardware.

Testing the Schemas

The e*Gate Integrator provides several different testing methods you can use to ensure that the schemas you create process data as expected. You can test

Collaboration Rules scripts and configuration files using e*Gate tools. Before creating sample data to use for the tests, determine a unique naming convention for the data to ensure that your data will not overlap any other test data. This section concentrates on testing e*Index schema components. For general information about testing the e*Ways for external systems, see "Unit Testing" in chapter 6 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

Collaboration Rules Scripts

You can use two different methods to test the Collaboration Rules scripts for the schemas you create.

- **Monk Test Console**
  Use the Monk Test Console in the e*Gate Enterprise Manager to test the operation of each schema component (for all new schemas) for correct data transformation. For more information about the test console, see "Unit Testing" in chapter 6 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

  Note: *You cannot test the sample Collaboration Rules script using this method. To test the sample script, use the following method,* ***stctrans****.*

- **stctrans**
  You can run Monk functions from the command line using the stctrans command. This program functions in the same way as the Monk Test Console, without the graphical user interface (GUI). See the *e*Gate Integrator System Administration and Operations Guide* for information on its usage.

Configuration Files

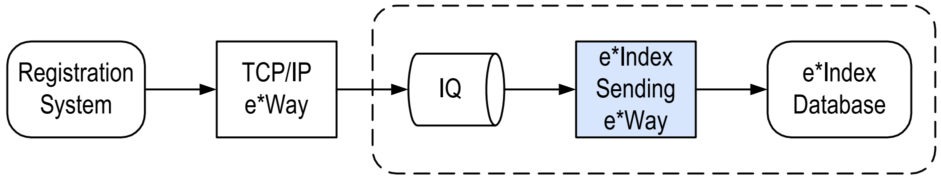
Once the transformations used by an e*Way have been tested, you must run the e*Way to test whether it is properly configured to communicate with the appropriate external system. Since BOBs do not connect to external systems, they do not have configuration (*.cfg) files. Testing an e*Way's configuration requires the use of an additional component, a file e*Way (**stcewfile.exe**), that passes data from a flat file to the e*Way being tested or passes data from the e*Way being tested to a flat file. For the e*Index sample schema, you can use the e*Ways ewUIEATER, ewUIFEEDER, and if necessary ewUIPOLLEATER.

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.*

*Sending e*Ways*

Figure 7-2 below displays a standard end-to-end diagram of the flow of information to be tested for the sending e*Way.

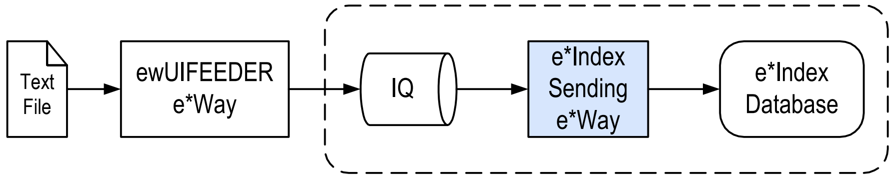
Figure 7-2: Sending e*Way End-to-End Diagram



In this example the TCP/IP e*Way receives member information in HL7 format from a registration system. It converts the data into the format required by the e*Index database and publishes it to the IQ. The e*Index e*Way retrieves the data from the IQ and inserts it into the e*Index database (see Figure 7-2).

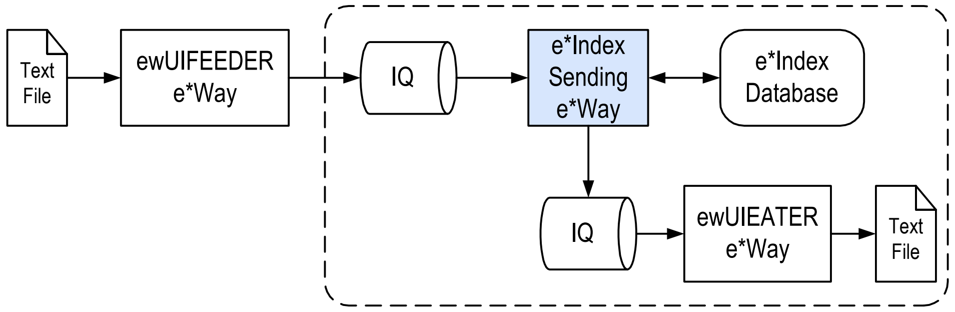
To test only the sending e*Way processing events to the database, we replace the TCP/IP e*Way with the standard file e*Way, ewUIFEEDER, which reads a text file containing HL7 data. You can then view the information in the database to verify the data was processed correctly (you can do this using either the e*Index GUI or any SQL editor). Figure 7-3 illustrates this test scenario.

Figure 7-3: Sending e*Way Test Scenario 1



The advantage of this type of test is that it not only tests the data transformation, but it also tests the information the e*Way uses to communicate to the e*Index database. The e*Index e*Way is bi-directional. You can test both directions in one test scenario as shown below. Figure 7-4 illustrates a bi-directional test scenario.

Figure 7-4: Sending e*Way Test Scenario 2

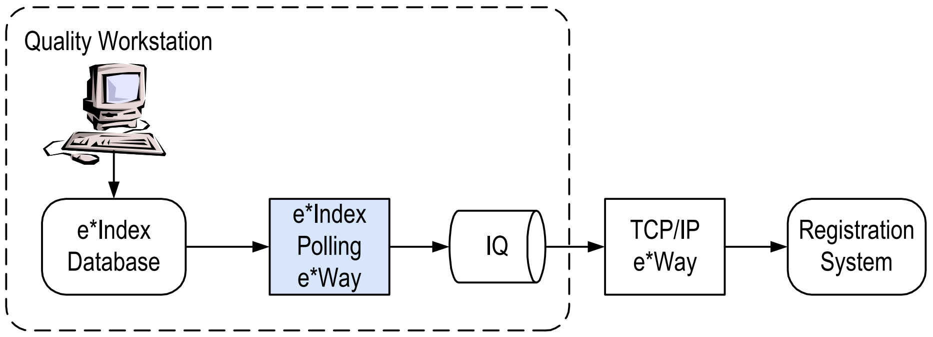


In this case, ewUIFEEDER reads a flat file, and sends the data to the e*Index sending e*Way via an IQ. The sending e*Way sends the information to the database for processing and receives the processed data back from the database. The sending e*Way then sends the processed data to ewUIEATER via an additional IQ. ewUIEATER then writes the information to a flat file.

*Polling e*Ways*

To test the polling e*Way's configuration, use the file e*Way (**stcewfile.exe**) named ewUIPOLLEATER, to pass data from the e*Way being tested to a flat file. Figure 7-5 on the following page displays a standard end-to-end diagram of the flow of information to be tested for the polling e*Way.

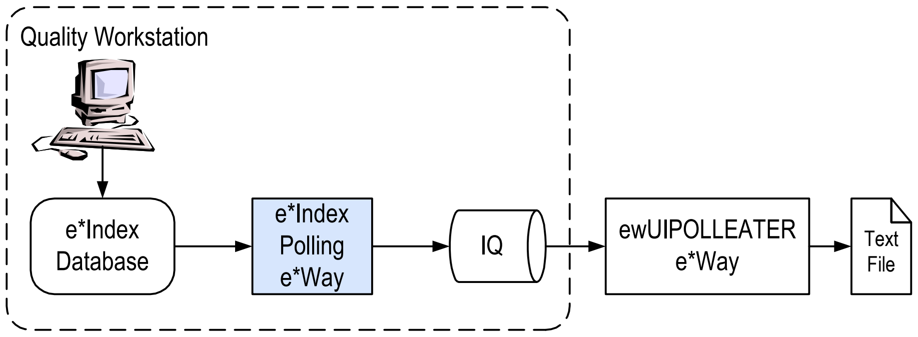
Figure 7-5: Polling e*Way End-to-End Diagram



In this example the database receives member information from the e*Index Quality Workstation, and writes an Event to the *ui_msg_detail* table. The e*Index polling e*Way retrieves the Event from the database and publishes it to the IQ. The TCP/IP e*Way retrieves the data and inserts it into the registration system database (it is important to note here that the receiving system must be set up to receive information from the e*Way).

To test the polling e*Way processing events from the database, replace the TCP/IP e*Way with the standard file e*Way (ewUIPOLLEATER), which reads Events from the IQ and writes the information to a flat file. You can then view the information in the file to verify the data was processed correctly.

Figure 7-6: Polling e*Way Test Scenario



▶ To test e*Index sending e*Ways:

Before you begin:

- ✓ Make sure the database is running and configured as it will be for production
- ✓ Make sure the sending e*Way is properly configured for both inbound and outbound Events
- ✓ Create a flat file containing Events to be processed by the e*Index e*Way in the format described by the e*Way's ETD

Note: *To test the e*Ways for any external systems, see "Testing e*Way Configuration Files" in chapter 6 of the* e*Xchange eBusiness Integration Suite Deployment Guide.

---

1. Configure the **ewUIFEEDER** e*Way to read the flat file you created. To do this, modify the Poller (inbound) Settings of the configuration file by specifying the path to the flat file and the file extension.
2. Make sure the Collaboration (**colREAD_EI_EVENT**) subscribes and publishes to the Event subscribed to by the sending e*Way without doing any data transformation (Pass through Service). It should subscribe to external and publish to the IQ to which the sending e*Way subscribes.
3. Configure the **ewUIEATER** e*Way to write to a flat file in your test path. To do this, modify the Outbound (send) Settings of the configuration file by specifying a pathname and filename for the file.
4. Make sure the Collaboration (**colWRITE_EI_EVENT**) subscribes and publishes to the Event published by the sending e*Way without doing any data transformation (Pass through Service). It should subscribe to the IQ to which the sending e*Way publishes, and it should publish to external.
5. Set the file e*Ways, the e*Way you are testing, and the IQ to automatically start when the Control Broker starts. The other components in the schema must not be set to start automatically.
6. Start the schema.
7. If possible, verify the connectivity between the database and the e*Way you are testing before sending any data. The sample e*Index schema writes an entry into the log file if the connection to the external system is successful.
8. The data file you specified should be read by the "feeder" e*Way, processed by the sending e*Way to the database, and then a UID is attached to the message. The modified records are then processed back

out through the sending e*Way and picked up by the "eater" e*Way to be written to a file.

▶ To test the e*Index polling e*Way:

Whether you are using the polling e*Way to send GUI transaction information to external systems or to simply empty the *ui_msg_detail* table, you should test the e*Way to be sure it is working as expected.

Before you begin:

- ✓ Make sure the database is running and configured as it will be for production
- ✓ Make sure the polling e*Way is properly configured

1. Enter and update records in the e*Index database using the e*Index Global Identifier GUI to place messages in the *ui_msg_detail* table. Document the information you entered or modified.
2. Configure the **ewUIPOLLEATER** e*Way to write to a flat file in your test path. To do this, modify the Outbound (send) Settings of the configuration file by specifying a pathname and filename for the file.
3. Make sure the Collaboration (**colUIPOLLEATER_HL7**) subscribes and publishes to the Event published by the polling e*Way without doing any data transformation (Pass through Service). It should subscribe to the IQ to which the polling e*Way publishes, and it should publish to external.
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 not be set to start automatically.
5. Start the schema.
6. If possible, verify the connectivity between the database and the e*Way you are testing before sending any data. The sample e*Index schema writes an entry into the log file if the connection to the external system is successful.
7. The data you entered into the GUI should be read and processed by the polling e*Way, and then written to the IQ. The Events are then picked up by the poll eater e*Way to be written to a file.

Testing the GUIs

Once you configure security, add or modify code table data as required, and modify the system parameter values, you can perform unit testing on the e*Index GUIs. Be sure to perform the following tests.

- Verify that the correct fields appear on the GUI with the correct field labels and containing the correct data element.
- Verify that any required fields are required when adding a record.
- Verify that the tab labels and search type labels are correct.
- Verify the formats of the SSN, Phone, and Postal Code fields.
- Process transactions through the e*Index GUI to verify all work as required.
- Verify that security is properly configured.
- Verify that code table data has been added or modified accurately.
- Verify that the control key settings are configured correctly.

In testing the GUI, you can also verify that the database is installed correctly, and is storing and processing data as designed.

Testing the Reports

Since production reports are customized for your implementation, they must be tested against specific data sets to ensure that they capture all of the information you require. Be sure to test that the following information appears on the appropriate reports and that the information is accurate.

- All cross-facility potential duplicates according to the criteria specified for each cross-facility report
- All same-facility potential duplicates according to the criteria specified for each same-facility report
- All assumed matches
- All update transactions, merge transactions, and unmerge transactions
- Any custom information to be reported on, as defined in the specifications

Be sure to include the weekly, monthly, and yearly reports in your test plan.

Testing the Matching Algorithm and Configurable Query

The matching algorithm should be customized by a qualified SeeBeyond services member. Once the files are customized, you need to test the configuration to ensure that the matching logic provides the most accurate identification of person records. You can test the matching algorithm configuration by extracting data from a legacy system and processing the data through the feeder and e*Index sending e*Ways, by entering data manually through the e*Index GUI, or by performing an initial load on the extracted data. The most thorough approach is a combination of all three methods.

The goals of this test are to verify that the algorithm is accurately identifying potential duplicate records and that any assumed matches (automatic merges) are made correctly. When testing the algorithm, verify that weights are being assigned as expected based on the match key fields and that the candidate selection pool returned from each search is accurate. Document this test and the results in detail to enable your SeeBeyond representative to continue to fine-tune the algorithm configuration. These tests will also help you determine if the matching, duplicate, and minimum thresholds are configured accurately. To test the matching algorithm, build test cases to verify that the following types of transactions are processed as expected:

- Person searches executed from the GUI, using multiple combinations of input data.
- Record inserts from the e*Index GUI to verify potential duplicate identification and assumed match processing.
- New person inserts from the daily feeds to verify potential duplicate identification and assumed match processing.
- Updates to existing records to verify assumed match processing and potential duplicate processing.

Integration Testing

Overview

Integration testing verifies how well the components created during the implementation work together in the e*Index system. 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. Make sure unit testing is complete before proceeding to integration testing.

Test Methodology

Integration testing includes running end-to-end tests to verify that data is processed correctly from one system to another. For example, one test would be to send data from System A via e*Gate to the e*Index database, and verifying that the correct translations and processing occur to the data. You should perform the same scenario for all interfaces. There are two types of integration testing:

- **Partial Integration Testing**
  Partial integration testing includes testing the data flow and translations between one external system and the e*Index database. This is a complete end-to-end path.
- **Complete System Testing**
  Complete system testing includes testing the data flow between all external systems and e*Index. This type of test requires quite a great deal of coordination, and is preliminary to the acceptance test. Perform this test on a test system that duplicates the production system as closely as possible.

Performance Testing

One part of integration testing is performance testing to test whether the system works fast enough, and if not, to identify actual and potential bottlenecks. Perform this type of test once the components to be tested have been thoroughly unit tested. Performance testing is included within the context of integration testing because many factors in combination will affect the performance of the e*Index system, and changes made to one component may not change the performance of the entire system.

You may want to create a benchmark test plan to define the tests that must be performed in order to verify that the system does meet performance requirements. The test plan must specify the exact requirement or goal in terms of system performance. Finding the areas in the system where processing slows is one of the goals of performance testing. This allows you to allocate additional system resources in order to improve processing. There are two types of performance tests, speed testing and stress testing.

Speed Testing

Speed testing verifies whether the system is processing data fast enough through the e*Index schemas and the schemas for any external systems. During a speed test, make sure that the e*Way logging is set to a normal level. Using higher than normal logging levels can degrade system performance and slow processing speed.

Stress Testing

Stress testing verifies whether the entire e*Index system can handle the expected volume. During a stress test, try to overload the system with data to see if, where, or how the system might fail. This type of testing can reveal network and database bottlenecks, as well as any bottlenecks in schema design.

Testing the GUI

While performing integration testing, don't forget to include Quality Workstation users. While you are processing data through the e*Index system, you can test the effect of having several users logged into the e*Index database at one time making modifications and performing queries against the database. This also gives you an idea of the response time you can expect for the Quality Workstation users.

Initial Load Testing

Initial load testing should occur throughout the development and testing phases to be sure that all valid values are defined in the initial load valid value lists and that any excluded values are defined in the exclusion lists. During these tests, verify that data is being validated and corrected properly during the validation phase of the initial load. You may need to modify the validation rules during the test process as you continue to analyze the initial load process. Be sure to define the validation requirements in the test plan so the tester knows how each field should be validated or modified for loading into the database.

Once the data is loaded into the database, run the customized reports to verify and analyze the process. Also, be sure to review the "reject" file for records that could not be loaded into the database. The contents of this file may suggest that additional validation rules are required.

Acceptance Testing

Overview

Perform acceptance testing prior to moving the system into production. This test is your final check to verify that the system performs according to your specifications, and that all bottlenecks and remaining issues have been addressed. You can perform acceptance testing on a partial system before performing the test on the complete system. If the entire system is not being put into production at the same time, you can perform acceptance testing only on the portion of the system going into production. The test plan you created in the planning phase should include all of the conditions that the system must meet in order to pass the acceptance test and move to production.

Test Methodology

In order to perform reliable acceptance testing, ensure that your test environment matches your production environment as closely as possible. Make sure the hardware configuration you use is the same as the hardware configuration for the production environment. Allow plenty of time to complete the acceptance test, as you will need to run data through the system for several days and then analyze the results of the test. If modifications are required, the test should be rerun when the changes are finalized.

Before beginning the acceptance test, all unit and integration testing must be complete. The test data should consist of actual data from the legacy systems. Acceptance testing should include an initial loading of legacy data into the database using the initial load utility.

Troubleshooting the schemas

The e*Gate Integrator provides the following tools for finding and correcting errors in the schemas of the e*Index system. For more information about how to use these tools, see "Troubleshooting" in chapter 6 of the *e*Xchange eBusiness Integration Suite Deployment Guide*.

- e*Gate Monitor
- Log files
- e*Gate Alert Agent
- e*Gate SNMP Agent

Troubleshooting the Database

Due to the complex nature of large databases, the issues you may run into while working with the e*Index database can be difficult. For information specific to your database platform, you may want to review the documentation provided for that database platform. The following is a short list of general issues and common solutions.

- If the e*Index schema or GUIs are having trouble connecting to an Oracle database, verify your **listener.ora** and **tnsnames.ora** files. For a SQL Server database, verify your ODBC data source, and for Sybase, verify the **sql.ini** file.
- If the e*Index schema is having trouble connecting to the database, verify that the Database Setup parameters are defined correctly. If you change the password, make sure to press **Enter** after typing the new password.
- If the initial load process slows down after processing a large number of records, you may want to consider increasing your sizing definitions for the database.
- When logging on to an Oracle database, the following error appears: **Unable to resolve the TNS Server Name (ORA 12514).**
  - Make sure that the ServerName variable in **stc_ua.ini** is set to the Oracle SID (or SERVICE_NAME) name in **tnsnames.ora**. This should not be the HOST name.
  - If you have more than one instance of **tnsnames.ora**, make sure that each instance contains a stanza pointing to the e*Index database.
- When logging on to an Oracle database, the following error appears: **Oracle not found; Oracle not available.**
  - Make sure your Oracle database is started (use the **startup** command in Oracle's Server Manager).
  - Verify that the domain name in the **sqlnet.ora** file is the same as that used for the database in **tnsnames.ora**.
- When trying to connect to an Oracle database, the following error appears: **ORA-12638: Credential retrieval failed.**
  - Comment out the following line in the client's **sqlnet.ora** file: **SQLNET.AUTHENTICATION_SERVICES=(NTS)**
- For troubleshooting Oracle databases, you can get the underlying SQL statement by logging into the database as a DBA user and running the following statement:

  ```
  alter system set events '1031 trace name errorstack level 10';
  ```

  After executing the statement, reproduce the error. The database writes a log file to the 'user_dump_dest' location (as specified in the database initialization file, **init<SID>.ora**).

Troubleshooting the GUIs

The primary problems you might encounter with the e*Index GUIs are related to database connectivity. If you are having trouble launching the e*Index GUI, or cannot connect to the database, check the following:

- For Oracle, make sure the **tnsnames.ora** file on the client workstation is configured correctly. For SQL Server, make sure the ODBC data source has been defined accurately. For Sybase, make sure the **sql.ini** file on the client workstation is configured correctly.
- For Oracle, make sure that **listener.ora** identifies the e*Index database correctly.
- Verify that the ServiceName and Database name in **stc_ua.ini** is defined correctly. For more information about the values to enter for **stc_ua.ini** variables, see "Step 7: Modify **stc_ua.ini**" in chapter 6 of the *e*Index Global Identifier Installation Guide*.
- Make sure the correct **uiserver.dll** file is registered. To do this, open an MS-DOS window and navigate to the directory of the e*Index client. At the prompt, type **regsvr32 uiserver.dll**.
- If you cannot connect to the database through the e*Index GUIs, verify that you can connect to the database using SQL Plus. If you cannot connect through SQL Plus, your database may not be started or the **tnsnames.ora** or **listener.ora** files may not be configured correctly.
- If you are having trouble logging on, and you have e*Index 1.03 or earlier installed, you need to rename the file **uiserver.dll** for the earlier version. This file is located in **\WINNT\system32** or **\Windows\system32**.

Additional errors may occur when trying to change your e*Index password. If you cannot change your password, check the following:

- If the following error message appears, make sure the new password you enter does not begin with a numeric character: **Error occurred while altering user password. ORA-00988: missing or invalid password(s)**, e*Index does not allow passwords beginning with numbers.
- If the following message appears, close e*Index and log in again using an uppercase password, and enter an uppercase password in the Original Password field: **Original password is not correct. Please try again.** While you can log into the database using either uppercase or lowercase characters, passwords are stored in uppercase.

Training

Overview

Prior to moving the e*Index system to production, make sure that all of the people who will be using and maintaining the e*Index system are fully trained. Training classes are offered through the SeeBeyond training department.

Scheduling End-User Training

Once you determine the personnel who will be using the e*Index GUIs, contact your SeeBeyond representative to schedule their training. Training can be either on-site or at the SeeBeyond headquarters in Monrovia, California. Be sure that the training is scheduled to be completed prior to the move to production.

In addition, if the personnel who will be maintaining the e*Gate components and the database were not part of the implementation team, they may require additional training. Again, make sure their training is scheduled to be completed before the move to production.

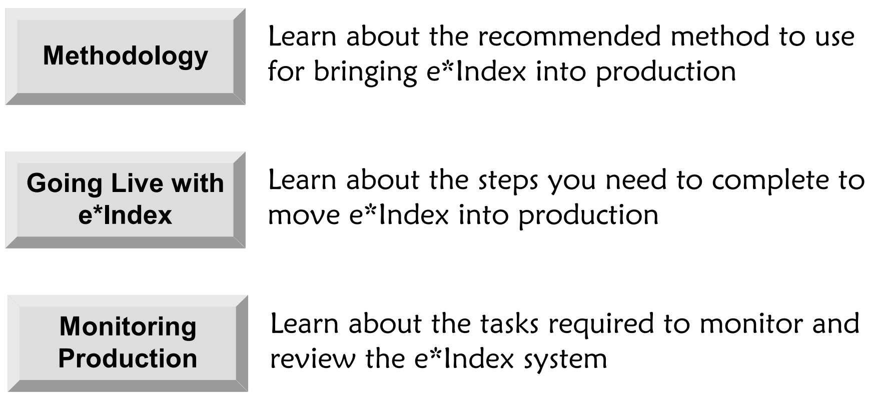
Moving to Production

About this Chapter

Overview

This chapter provides the background information and procedures you need to perform to move your e*Index implementation from testing to a production environment. It also includes information about monitoring and reviewing the production system.

The following diagram illustrates the contents of each major topic in this chapter.



Methodology	Learn about the recommended method to use for bringing e*Index into production
Going Live with e*Index	Learn about the steps you need to complete to move e*Index into production
Monitoring Production	Learn about the tasks required to monitor and review the e*Index system

Go-Live Methodology

Overview

This section provides a summary of SeeBeyond's planned methodology for bringing e*Index into a production environment. This method has been tested and proven, and following the specified order of tasks will help ensure that no data is lost in the transition. This chapter discusses a passive implementation, and does not discuss moving an active implementation into production.

Methodology Description

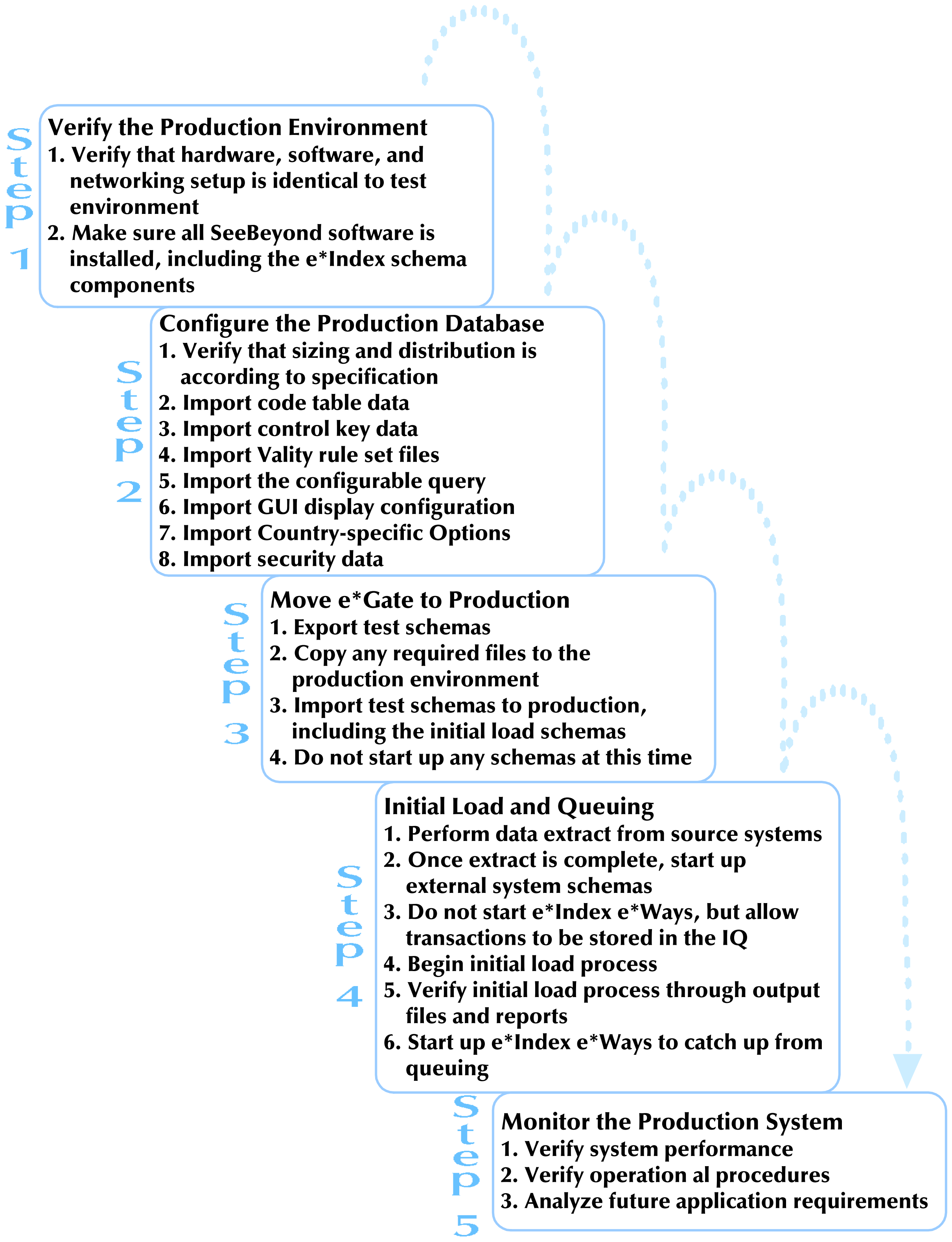
The order in which e*Index and e*Gate system components are migrated from the test to the production environment is designed to ensure that no data is lost during the transition process. The migration includes moving the e*Gate environment, e*Index schemas, initial load schemas, e*Index database, and e*Index client workstations to the production environment. The hardware, software, and networking configuration of the production environment should be verified before you begin the migration.

The first step in the migration is to configure the production database and ensure that all of your customized data has been created. This includes code table data, control key values, country-specific options, GUI display configuration, the matching algorithm, configurable query, and security information. You can export this information from your test database and import it into the production database, or you can enter the information into the production database through the e*Index GUIs. Once your database is in place, you can migrate the e*Gate environment by exporting the schemas from the test environment and importing them into the production environment.

In order to ensure that no data is lost or duplicated during the transition, begin the extract of the initial load data from source systems at the same time you start up the schemas for external systems. Once the data extract begins, start all schema components for the external systems and the IQ to which the e*Index sending e*Ways subscribe. Do not start the e*Index e*Ways at this time. Instead, queue up incoming Events until the initial load process is complete. Once you complete and verify the initial load process, start the e*Index e*Ways to catch up from the queue. Your e*Index system is now live, and should be monitored carefully for performance and processing accuracy.

Figure 8-1 on the following page provides a summary of steps to bringing the e*Index and e*Gate systems to production.

Figure 8-1: Steps to Bringing e*Index to Production



Verify the Production Environment
1. Verify that hardware, software, and networking setup is identical to test environment
2. Make sure all SeeBeyond software is installed, including the e*Index schema components

Configure the Production Database
1. Verify that sizing and distribution is according to specification
2. Import code table data
3. Import control key data
4. Import Vality rule set files
5. Import the configurable query
6. Import GUI display configuration
7. Import Country-specific Options
8. Import security data

Move e*Gate to Production
1. Export test schemas
2. Copy any required files to the production environment
3. Import test schemas to production, including the initial load schemas
4. Do not start up any schemas at this time

Initial Load and Queuing
1. Perform data extract from source systems
2. Once extract is complete, start up external system schemas
3. Do not start e*Index e*Ways, but allow transactions to be stored in the IQ
4. Begin initial load process
5. Verify initial load process through output files and reports
6. Start up e*Index e*Ways to catch up from queuing

Monitor the Production System
1. Verify system performance
2. Verify operation al procedures
3. Analyze future application requirements

Going Live with e*Index

Overview

This section of the chapter outlines the tasks you must complete to move e*Index into a production environment, and the order in which the tasks should be completed. Closely monitor the process of moving to production and the performance of the application so you can make any necessary adjustments to ensure a smooth transition from the test environment. To move e*Index into production, complete the following tasks:

- Step 1: Verify the Production Environment
- Step 2: Configure the Production Database
- Step 3: Move e*Gate from Test to Production
- Step 4: Extract Initial Load Data
- Step 5: Start e*Gate Queuing
- Step 6: Perform the Initial Data Load
- Step 7: Run the Conversion Reports
- Step 8: Catch up from e*Index Queuing

Step 1: Verify the Production Environment

The first step in bringing e*Index live is to make sure that the production environment has been installed and is thoroughly tested. Verify the hardware, software, and network connectivity configuration before beginning the steps to going live. Also, make sure that the e*Gate Registry Host and Participating Hosts, along with any add-ons, are installed on the production servers exactly as they are installed on the test servers. Similarly, any third-party software, such as Oracle, Sybase, or SQL Server, must be installed on the production servers. It is very important to verify the software configuration prior to beginning the import procedures for moving the test schemas into production.

In addition, the database server and client workstations must be configured as they were in the test environment.

Step 2: Configure the Production Database

Before beginning the move to production, you need to make sure that you have an e*Index database installed and verified on the production server. Your test database must be configured with all of the code table elements, system parameter values, and security information that you will use in your production database. The test database should also contain the final matching algorithm configuration, country-specific option information, GUI display configuration, and configurable query. If this is the case, you can export this information from your test database and import the information into the production database. This way, you do not have to re-enter information such as your systems, regions, person categories, security profiles, and so on.

You can export the entire test database and import it into the production environment, or you can only export the configuration data. The tables you need to export and import for each type of data are listed under "Exporting Configuration Data". Before performing the import, make sure you truncate any tables in the production database into which you are importing data from the test database.

Exporting the Entire Test Database

If you choose to export the entire test database and import it into the production environment, the database will contain all of your test data from the GUI, initial load, and daily feeds. To remove this data, you need to truncate certain tables and reset the next UID number. The tables you need to truncate are listed below. Before you truncate tables, disable any foreign keys. They must be enabled again once all tables are truncated. Your SeeBeyond representative can provide SQL scripts that will disable the foreign keys, truncate the appropriate tables for you, set the next UID number, and then enable the foreign keys.

Important! *If you export your entire database, make sure that the information in the Vality tables (*ui_ctrl_file *and* ui_ctrl_rule*) is the most current.*

- ui_comment
- ui_alias
- ui_alias_x_name
- ui_alias_history
- ui_local_id
- ui_local_id_history
- ui_duplic
- ui_assumed_match
- ui_address
- ui_address_history
- ui_phone
- ui_phone_history
- ui_aux_id
- ui_aux_id_history
- ui_audit
- ui_mrg_trans

- ui_person_history
- ui_person
- ui_person_x_name
- ui_transaction
- ui_msg_detail
- ui_msg_header
- ui_login

After you import the test database to the production database, update the next UID number by running the following SQL script on the production database. Replace <next_id> with the beginning UID number for your implementation. Make sure the length of the UID matches the length specified by the control key UIDLENGTH.

```
ALTER TRIGGER "UI"."TUB_UI_SEQ_NO" DISABLE;
update ui_seq_no set seq_no=1;
update ui_seq_no set seq_no=<next_id> where
  table_name='ui_person';
ALTER TRIGGER "UI"."TUB_UI_SEQ_NO" ENABLE;
commit;
```

Exporting Configuration Data

The lists below indicate the tables to export for each type of data. After you complete the export and import, make sure that the information in the Vality tables (*ui_ctrl_file* and *ui_ctrl_rule*) in the production environment is the most current. In addition to the files listed below, you may want to import the *ui_nickname* table if you have added names to the table. Before you import any tables, you should truncate them in the production database.

Exporting Security Data

To export your security configuration from the test database, export the following tables, and then import them into the production database:

- stc_acc_def
- stc_group
- stc_group_acc
- stc_module
- stc_user
- stc_user_acc
- stc_user_region
- stc_user_group
- ui_control_sec
- ui_login
- ui_login_current

- ui_no_passwd
- ui_notify_user
- ui_user_passwd_hist

Note: *If you have not made any modifications to the* ui_no_passwd *table, you do not need to export and import the data. This table can only be modified using a database tool such as SQL Plus.*

Exporting Code Table Data

To export the code table data you have customized in the test database, export the following tables, and then import them into the production database. Some of the tables, such as ui_message and ui_zip, have many records. You do not need to export tables in which no data elements were added or modified.

- stc_common_detail
- stc_common_header
- ui_aux_id_def
- ui_canned_msg
- ui_dept
- ui_facility
- ui_message
- ui_zip

Exporting Country-specific Data

To export the country-specific table data you have customized in the test database, export the following tables, and then import them into the production database. If you did not modify any country-specific options, you do not need to export and import these tables.

- ui_misc_opt_control
- ui_misc_opt_country
- ui_misc_option

Exporting the GUI Display Configuration

To export the GUI display configuration you customized in the test database, export the following tables, and then import them into the production database. If you did not modify any GUI display attributes, you do not need to export and import these tables.

- ui_table
- ui_table_column

Exporting the Configurable Query

To export the configurable query data that you customized in the test database, export the following tables, and then import them into the production database. If you did not change the configurable query, you do not need to export and import these tables.

- ui_cand_from_table
- ui_cand_select_column
- ui_cand_where_column
- ui_cand_sql
- ui_cand_sql_column
- ui_cand_sql_table
- ui_cand_where_clause

Exporting the Vality Rule Set Files

To export the Vality rule set data that you customized in the test database, export the following tables, and then import them into the production database. If you did not change the matching logic, you do not need to export and import these tables.

- ui_ctrl_rule
- ui_ctrl_file
- ui_nickname

Exporting Control Key Data

To export your system parameters configuration, export the *ui_control* table, and then import it into the production database.

Step 3: Move e*Gate from Test to Production

To migrate the e*Gate environment from test to production, you need to move certain e*Gate components from the test e*Gate servers to the production e*Gate servers. You can export all of your test schemas for external systems, e*Index, and the initial load from the test e*Gate environment and then import the schemas into the production environment. Use the e*Gate export and import tools and the SeeBeyond Registry utility to perform the migration.

Exporting e*Gate Schemas

To export an e*Gate schema, type the following at the command prompt:

```
    stcregutil.exe -rh registry_host -rs schema -un user_name
    -up password -rt -usr -e filename
```

where

registry is the name of the Registry Host

schema is the name of the schema you want to export

user-name and *password* are your login ID and password for e*Gate, and

filename is the name of the ASCII text file to which the schema data will be written.

The *–rt* and *–usr* parameters indicate that the resource table and user names will be exported. You can also export a schema using the Export Schema Definitions feature. For more information about using **stcregutil.exe** or the export feature, see the *e*Gate Integrator Administration and Operations Guide*.

Important! *This method for moving an e*Gate schema assumes that at least **one** of the following statements is true:*

- *The Default schemas for the source and target Registry Hosts are identical.*
- *The schema that is being moved does not rely on any files stored in the Default schema. In other words, all of the files required by the schema are stored in the schema directory and not the **\default** directory.*

*The e*Index and initial load schemas do rely on files stored in the **\default** directory, so if you are going to export and import the schemas, the Default schemas for both host machines must be identical. See "Importing e*Gate schemas" below for more information.*

Importing e*Gate Schemas

Before you can import the schemas, you need to copy the export files of the test schemas to the production server. Depending on the size and number of files to import, you can move them from the test environment via floppy disk, tape, CD or LAN connection. Make sure the archiving tool you use can archive and restore files that have file names that are longer than those restricted by "8.3" formatting. Before importing the schemas, complete the following tasks:

- Copy the text files created from the e*Gate schema export.
- Copy all files in **\<eGate>\server\registry\repository\<schema>**, where **<schema>** is the name of the schemas you exported. Be sure that the directory restructure is retained when the Registry files are archived and restored.

- If you are migrating e*Index or initial load schemas from the test to the production environment, you must first install the e*Gate components from the e*Index schema on the production server using the e*Index installation CD-ROM. This will place files in the Default schema that are required for e*Index and for the initial load.

To import an e*Gate schema to your production environment:

1. Before you begin, edit the import file by changing the e*Gate Host name from the test Host to the production Host. Otherwise, the schema will have two Participating Hosts, one that you need to delete and one that you need to rename.

2. Type the following at the command prompt to create the Registry for the schema:

```
stcregutil.exe -rh registry_host -rs schema -un user_name
-up password -i filename
```

where

registry is the name of the Registry Host

schema is the name of the schema you want to export

user-name and ***password*** are your login ID and password for e*Gate, and

filename is the name of the ASCII text file that was created during the export procedure.

3. Create a new directory for the schema files. The new directory must match the name of the imported schema. For example, if you imported a schema named **eIndex**, the path for the schema files should look like this:

   \<eGate>\ server\registry\repository\eIndex

4. Restore all of the files copied from the schema in the test environment to the newly created directory. Make sure you maintain the original directory structure from the test environment.

5. Copy any required files to the default directory. You only need to copy files that you may have customized. For example, in the Default schema directory for the e*Index schema, you may have customized **ui-fns.monk** or **eiEvent.ssc**. If you customized these files, copy them to **\<eGate>\server\registry\repository\default\monk_library\ui** and **...\default\monk_scripts\ui** respectively.

A schema can also be imported using the Import Schema Definitions feature. For more information about using the import feature or **stcregutil.exe**, see the *e*Gate Integrator Administration and Operations Guide*.

Before moving on to the next step, make sure that all of the schemas in the test environment are moved onto the production servers. Make one final check that all required software is installed, and that all of the schema components have been imported. Before activating the production servers, make sure that the e*Index database is up and running, and ready for the initial load procedure.

Step 4: Extract Initial Load Data

Before you begin processing transactions through the e*Index system, you need to perform the data extract from the systems that are sharing data with e*Index. As the same time you begin the extract, begin e*Gate queuing as described in "Step 5: Start e*Gate Queuing". Performing the steps simultaneously begins queuing Events from external systems as the extract is being performed, ensuring that no transactions are missed.

Step 5: Start e*Gate Queuing

As you prepare to move e*Index from a test to production environment, you want to make sure that no transactions are missed in the cross-over. The best approach is to begin queuing Events for the e*Index database as soon as you begin the data extracts for the initial load. At this point, you need to start the e*Ways for the external systems sharing data with e*Index, but the e*Index sending and polling e*Ways should not be started yet. Make sure that they are not configured to start automatically.

To start the schemas for external systems, start the e*Gate Control Broker on the host machines where those schemas are located. Start the Control Broker as a service for Windows. Any schemas that are scheduled to start automatically will begin to run. Monitor the e*Way to be sure they are running and are processing data correctly. To start queuing for the e*Index database, start the **IQ iqRAW_EI_EVENT** in the Control Broker. For detailed instructions about running the Control Broker in a production environment, see the *e*Gate Integrator System Administration and Operations Guide*.

Step 6: Perform the Initial Data Load

Now that the daily transactions are being stored in the e*Index queue, you can begin the initial data load process to populate the e*Index database with legacy data. It is best to schedule this portion of the move to production during a time when the anticipated data volume is lower than usual because all Events that are sent to e*Index during the initial load are written to the e*Gate queue. As the data is loaded into the database, monitor the process closely to ensure that no errors are occurring and that processing speed is not degrading. Make sure to monitor and correct and records processed to the "bad" data file. This process may take several hours to a few days.

Step 7: Run Reports on the Initial Load

After all the required data has been back-loaded, run the reports you customized earlier in the implementation. Review these reports to be sure that Events were identified, matched, and flagged as potential duplicates appropriately. In these initial reports, you may have a larger than normal number of potential duplicate records, but after you analyze the report results you can fine-tune the duplicate threshold control key, DUPTHRESH, and the matching threshold, MATCHTHRES, as needed.

Step 8: Catch up from e*Index Queuing

When the initial load is complete, and you have verified the results of the initial load by running the conversion reports, you can start the e*Index e*Ways to catch up from the queue and start processing live transactions. Before you start the e*Index e*Ways, configure the e*Ways to start automatically (using the e*Gate Enterprise Manager). Then start the e*Gate Control Broker, again starting the Control Broker as a service for Windows. If you have set the e*Index e*Ways to start automatically, they should begin to run. For detailed instructions about running the Control Broker in a production environment, see the *e*Gate Integrator System Administration and Operations Guide*.

Monitoring the Production System

Overview

Once your system is in production, it is important to monitor and review how the system is processing Events, including processing speed, data integrity, accurate identification, error handling, and so on. If you notice any errors, you can make the necessary adjustments to the system to ensure smooth performance. If any changes are required, be sure to thoroughly test the changes before implementing them in the production system. Monitoring the system falls under three categories:

- System Performance
- Operational Procedures
- Future Application Requirements

System Performance

Once your e*Index and e*Gate systems are in production, you need to make sure that each component is functioning in a timely manner. Routine checks of the system can help you determine long-term performance requirements against which to compare actual performance. Monitor the system for any bottlenecks or for any areas that could use improvement. You can use e*Gate tools, such as the e*Gate Monitor, to monitor the e*Gate components such as the external system schemas and e*Index schemas.

On the database server, make sure that there are no bottlenecks in the database, and check database tables for unexpected growth. If you notice that the *ui_audit* table is growing quickly, you may want to archive the table more regularly. If you notice the database is growing more rapidly than you expected, you may want to consider modifying certain sizing specifications.

Operational Procedures

In order to be sure that the e*Index system is working as expected, monitor the data flow to be sure that data is being transferred and translated accurately. Make sure that maintenance procedures are clearly defined and that the procedures are being followed. This includes data maintenance activities, such as reviewing potential duplicates and assumed matches, reviewing daily database reports, reviewing e*Way logs as needed, and so on. Make sure that the process for resolving potential duplicate records and performing manual updates for e*Index data is clearly defined and being followed.

Future Application Requirements

Periodically review the e*Index and e*Gate systems to verify that it is meeting your current business requirements. Keep in mind future processing requirements and how they can be incorporated into the e*Index and e*Gate systems. For example, if you are planning on incorporating additional legacy data into e*Index through an initial load process, make sure to consider the current status of the database and any changes that might be required in order to accommodate the additional transaction volume.

Appendix A

Sample Data Mapping Table

About this Appendix

The following pages contain a sample data mapping table for the demographic data elements stored in the e*Index database. This sample assumes that the messages coming from external systems are in standard HL7 format. For other messaging standards, you can adjust this table as required.

The sample data mapping table includes data elements that are stored in the tables *ui_address*, *ui_alias*, *ui_aux_id*, *ui_person*, *ui_local_id*, and *ui_phone*, and *ui_transaction*. These tables store the primary demographic data for each person record, and are linked together through the transaction_no column. There are 20 customizable fields: CLASS1 through CLASS5, in which you can store any data you require up to 20 characters in each field; STRING1 through STRING10, in which you can store data strings of varying length; and DATE1 through DATE5, in which you can store relevant dates.

e*Index HL7 → e*Index Table Mappings

UI_PERSON

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
U_ID	NOT NULL	STRING(15)	eiEvent.REC.ID		
PERSON_CAT_CODE		STRING(8)	eiEvent.REC.DEMO.person_category		
LAST_NAME		STRING(40)	eiEvent.REC.DEMO.person_name.last_name	PID:5	**LastName**^FirstName^MiddleName^Suffix^Prefix^Degree
FIRST_NAME		STRING(40)	eiEvent.REC.DEMO.person_name.first_name	PID:5	LastName^**FirstName**^MiddleName^Suffix^Prefix^Degree
MIDDLE_NAME		STRING(30)	eiEvent.REC.DEMO.person_name.middle_name	PID:5	LastName^FirstName^**MiddleName**^Suffix^Prefix^Degree
SUFFIX		STRING(10)	eiEvent.REC.DEMO.person_name.suffix	PID:5	LastName^FirstName^MiddleName^**Suffix**^Prefix^Degree
TITLE		STRING(8)	eiEvent.REC.DEMO.person_name.title	PID:5	LastName^FirstName^MiddleName^Suffix^**Prefix**^Degree
DOB		DATE	eiEvent.REC.DEMO.date_of_birth	PID:7	
DEATH		STRING(1)	eiEvent.REC.DEMO.death.DT.flag		
SEX		STRING(8)	eiEvent.REC.DEMO.sex	PID:8	
MSTATUS		STRING(8)	eiEvent.REC.DEMO.marital_status	PID:16	
SSN		STRING(8)	eiEvent.REC.DEMO.SSN_number	PID:19	
RACE		STRING(8)	eiEvent.REC.DEMO.race	PID:10	
ETHNIC		STRING(8)	eiEvent.REC.DEMO.ethnic_group	PID:22	
RELIGION		STRING(8)	eiEvent.REC.DEMO.religion	PID:17	
LANGUAGE		STRING(8)	eiEvent.REC.DEMO.language	PID:15	
SPOUSE_NAME		STRING(100)	eiEvent.REC.DEMO.alt_name.ANM.spouse_name		
MOTHER_NAME		STRING(100)	eiEvent.REC.DEMO.alt_name.ANM.mother_name		
MOTHER_MN		STRING(40)	eiEvent.REC.DEMO.alt_name.ANM.mothers_maiden_name	PID:6	
FATHER_NAME		STRING(100)	eiEvent.REC.DEMO.alt_name.ANM.fathers_name		
MAIDEN		STRING(40)	eiEvent.REC.DEMO.alt_name.ANM.maiden_name	PID:9	
POB_CITY		STRING(30)	eiEvent.REC.DEMO.birth_place.BP.city	PID:23	**POBCity**^POBState

UI_PERSON

Column Name	Null?	Data Type (Length)	eiEvent ETD Field	HL7 Field	HL7 Format
POB_STATE		STRING(10)	eiEvent.REC.DEMO.birth_place.BP.state	PID:23	POBCity^**POBState**
POB_COUNTRY		STRING(20)	eiEvent.REC.DEMO.birth_place.BP.country		
VIP_FLAG		STRING(8)	eiEvent.REC.DEMO.vip		
VET_STATUS		STRING(8)	eiEvent.REC.DEMO.veteran_status	PID:27	
STATUS		STRING(8)			For e*Index use only.
DRIVERS_LICENSE		STRING(20)	eiEvent.REC.DEMO.driver_license.DL.state_country	PID:20	**LicenseNumber**^Issuer
DRIVERS_LICENSE_ST		STRING(10)	eiEvent.REC.DEMO.driver_license.DL.number	PID:20	LicenseNumber^**Issuer**
DOD		DATE	eiEvent.REC.DEMO.death.DT.date_of_death		
DEATH_CERTIFICATE		STRING(10)	eiEvent.REC.DEMO.death.DT.death_certificate_number		
NATIONALITY		STRING(8)	eiEvent.REC.DEMO.nationality		
CITIZENSHIP		STRING(8)	eiEvent.REC.DEMO.citizenship	PID:26	
PENSION_NO		STRING(15)	eiEvent.REC.DEMO.pension.PN.number		
PENSION_EXP_DATE		DATE	eiEvent.REC.DEMO.pension.PN.expiration_date		
REPATRIATION_NO		STRING(16)	eiEvent.REC.DEMO.repatriation_number		
DISTRICT_OF_ RESIDENCE		STRING(8)	eiEvent.REC.DEMO.district_of_residence		
LGA_CODE		STRING(4)	eiEvent.REC.DEMO.LGA_code		
MILITARY_BRANCH		STRING(4)	eiEvent.REC.DEMO.military.ML.branch		
MILITARY_RANK		STRING(4)	eiEvent.REC.DEMO.military.ML.rank_grade		
MILITARY_STATUS		STRING(4)	eiEvent.REC.DEMO.military.ML.status		
DUMMY_DATE		DATE			
CLASS1		STRING(20)	eiEvent.REC.AUX.class		Note: Use this and the following fields to store any information not already included in this table.
CLASS2		STRING(20)	eiEvent.REC.AUX.class		
CLASS3		STRING(20)	eiEvent.REC.AUX.class		
CLASS4		STRING(20)	eiEvent.REC.AUX.class		
CLASS5		STRING(20)	eiEvent.REC.AUX.class		

UI_PERSON

Column Name	Null?	Data Type (Length)	eiEvent ETD Field	HL7 Field	HL7 Format
STRING1		STRING(40)	eiEvent.REC.AUX.string		
STRING2		STRING(40)	eiEvent.REC.AUX.string		
STRING3		STRING(40)	eiEvent.REC.AUX.string		
STRING4		STRING(40)	eiEvent.REC.AUX.string		
STRING5		STRING(40)	eiEvent.REC.AUX.string		
STRING6		STRING(40)	eiEvent.REC.AUX.string		
STRING7		STRING(100)	eiEvent.REC.AUX.string		
STRING8		STRING(100)	eiEvent.REC.AUX.string		
STRING9		STRING(100)	eiEvent.REC.AUX.string		
STRING10		STRING(255)	eiEvent.REC.AUX.string		
DATE1		DATE	eiEvent.REC.AUX.date		
DATE2		DATE	eiEvent.REC.AUX.date		
DATE3		DATE	eiEvent.REC.AUX.date		
DATE4		DATE	eiEvent.REC.AUX.date		
DATE5		DATE	eiEvent.REC.AUX.date		

UI_ADDRESS

Column Name	Null?	Data Type (Length)	eiEvent ETD Field	HL7 Field	HL7 Format
U_ID	NOT NULL	STRING(15)			
ADDRESS_TYPE	NOT NULL	STRING(8)	eiEvent.REC..DEMO.address.AD.type		
ADDRESS1		STRING(40)	eiEvent.REC..DEMO.address.AD.street_1	PID:11	**StreetAddress**^OtherDesignation^City^State^ZipCode^Country^Type^Other
ADDRESS2		STRING(40)	eiEvent.REC..DEMO.address.AD.street_2	PID:11	StreetAddress^**OtherDesignation**^City^State^ZipCode^Country^Type^Other

UI_ADDRESS

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
ADDRESS3		STRING(40)	eiEvent.REC..DEMO.address.AD.street_3		
ADDRESS4		STRING(40)	eiEvent.REC..DEMO.address.AD.street_1		
CITY		STRING(30)	eiEvent.REC..DEMO.address.AD.city	PID:11	StreetAddress^OtherDesignation^**City**^State^ZipCode^Country^Type^Other
STATE_CODE		STRING(10)	eiEvent.REC..DEMO.address.AD.state_or_province	PID:11	StreetAddress^OtherDesignation^City^**State**^ZipCode^Country^Type^Other
POSTAL_CODE		STRING(8)	eiEvent.REC..DEMO.address.AD.zip	PID:11	StreetAddress^OtherDesignation^City^State^**ZipCode**^Country^Type^Other
POSTAL_CODE_EXT		STRING(4)	eiEvent.REC..DEMO.address.AD.zip_ext	PID:11	StreetAddress^OtherDesignation^City^State^**ZipCode**^Country^Type^Other
COUNTY		STRING(20)	eiEvent.REC..DEMO.address.AD.county	PID:12	
COUNTRY		STRING(20)	eiEvent.REC..DEMO.address.AD.country	PID:11	StreetAddress^OtherDesignation^City^State^ZipCode^**Country**^Type^Other

UI_PHONE

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
U_ID	NOT NULL	STRING(15)			
PHONE_TYPE	NOT NULL	STRING(8)	eiEvent.REC..DEMO.phone.PH.type		
PHONE	NOT NULL	STRING(20)	eiEvent.REC..DEMO.phone.PH.phone_number	PID:13 PID:14	
PHONE_EXT		STRING(6)	eiEvent.REC..DEMO.phone.PH.phone_ext	PID:13 PID:14	

UI_ALIAS

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
U_ID	NOT NULL	STRING(15)			
LAST_NAME	NOT NULL	STRING(40)	eiEvent.REC..DEMO.person_alias.NM.last_name	PID:9	**LastName**^FirstName^MiddleName^Suffix^Prefix^Degree
FIRST_NAME	NOT NULL	STRING(40)	eiEvent.REC..DEMO.person_alias.NM.first_name	PID:9	LastName^**FirstName**^MiddleName^Suffix^Prefix^Degree
MIDDLE_NAME		STRING(30)	eiEvent.REC..DEMO.person_alias.NM.middle_name	PID:9	LastName^FirstName^**MiddleName**^Suffix^Prefix^Degree

UI_LOCAL_ID

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
U_ID	NOT NULL	STRING(15)			
FACILITY	NOT NULL	STRING(20)	eiEvent.REC..ID.local_id.LID.system	PID:3	PatientID^CheckDigit^CheckDigitScheme^**Facility**^type
LOCAL_ID	NOT NULL	STRING(25)	eiEvent.REC..ID.local_id.LID.id	PID:3	**PatientID**^CheckDigit^CheckDigitScheme^Facility^type
STATUS		STRING(8)			This field is used in e*Index only.

UI_AUX_ID

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
U_ID	NOT NULL	STRING(15)			
AUX_ID_DEF	NOT NULL	STRING(10)	eiEvent.REC..ID.non_unique_id.NID.type		Note: Use these fields to store any identifiers that are not unique to each individual.
ID	NOT NULL	STRING(40)	eiEvent.REC..ID.non_unique_id.NID.id		

UI_TRANSACTION

Column Name	**Null?**	**Data Type (Length)**	**eiEvent ETD Field**	**HL7 Field**	**HL7 Format**
U_ID	NOT NULL	STRING(15)			
SYSTEM		STRING(8)	eiEvent.EVNT.EVN.source	MSH:4	
DEPT		STRING(8)	eiEvent.EVNT.EVN.dept		
TERMINAL_ID		STRING(25)	eiEvent.EVNT.EVN.ter minal_id		
SESSION_ID		STRING(15)			
UPDATE_FACILITY_ID	NOT NULL	STRING(20)	eiEvent.EVNT.EVN.assigning_system	PID:3	
UPDATE_FUNCTION	NOT NULL	STRING(8)	eiEvent.EVNT.EVN.event_type_code	EVN:1	
UPDATE_DATE	NOT NULL	DATE	eiEvent.EVNT.EVN.date_of_event	EVN:2	
UPDATE_USERID	NOT NULL	STRING(20)	eiEvent.EVNT.EVN.user_id	EVN:5	

Appendix B

Initial Load Processing Log

About this Appendix

The following pages provide a sample checklist for the first phase (validation and standardization) of the initial load procedure. This checklist includes information about fields in the initial load records and the validations and standardizations that are required to each field before a record can be loaded into the database using the initial load process. You can use this sample table for your own initial load checklist, or use it as a guideline to create a customized version.

Make sure you include all data elements that require validation or standardization on your initial load checklist. Also include a description of the results of each validation. For example, if a field should be left justified, but is not, the record may be sent to the "bad" data file during the validation phase of the initial load.

For more information about the validations used in the first phase of the initial load process, see "Validation Functions" in chapter 3 of the *e*Index Initial Load User's Guide*.

Initial Load Processing Log

Step 1 Checklist

Field	Description	Additional Information
Local ID	Allow numeric only	
Last Name	Allow alphabetic and punctuation only Check for nulls Check left-justified	
First Name	Allow alphabetic and punctuation only Check for nulls Check left-justified	
Middle Name	Allow alphabetic and punctuation only Check left-justified	
Suffix	Allow alphabetic and punctuation only Check valid values Check left-justified	See valid value list for allowed values
Address1 and Address2	Allow alphanumeric and punctuation Check left-justified	
City	Allow alphabetic and punctuation Check left-justified	
State	Allow alphabetic only Check left-justified	
Zip Code	Allow numeric only Check left-justified	

Field	Description	Additional Information
Zip Extension	Allow numeric only Check left-justified	
County Code	Allow alphabetic and punctuation Check left-justified	
Country	Allow alphabetic only Check valid values Check left-justified	See code table mappings for values
DOB	Allow numeric and punctuation Check left-justified	
SSN	Allow numeric and punctuation Check left-justified	
Phone	Allow numeric and punctuation Check right-justified	
Phone Extension	Allow numeric and punctuation Check right-justified	
Title	Check valid values Check left-justified	
Death	Check valid values	This does not display on the e*Index GUI, but can be stored for future reference.
Sex	Check valid values Check left-justified	See code table mappings for values
Marital Status	Check valid values Check left-justified	See code table mappings for values
Race	Check valid values Check left-justified	See code table mappings for values
Ethnicity	Check valid values Check left-justified	See code table mappings for values

Field	Description	Additional Information
Religion	Check valid values Check left-justified	See code table mappings for values

Validation descriptions

Validation	Description	Results
CheckNotNull	Checks for null values	If a null value is found, the record is sent to the "bad" data file and information is written to the error file.
CheckLeftJust or CheckRightJust	Checks for left or right justification of the field	If a field does not have the required justification, the record is sent to the "bad" data file and information is written to the error file.
CheckChars	Checks for character types (alphabetic, numeric, or punctuation)	If any non-permitted characters are found, the record is sent to the "bad" data file and information is written to the error file.
CheckVVL	Checks values in the field against a valid value list	If a value is found that is not on the list, the record is sent to the "bad" data file and information is written to the error file.

Appendix C

Sample Implementation Project Plan

About this Appendix

This appendix provides a sample project plan schedule for an e*Index implementation project. Each e*Index implementation is unique, so there is no typical project plan schedule. However, you can use the schedule included here as a basis for your own implementation.

ID	Task Name	Duration	Start	Finish	Resource Names
1	**e*Index Project Timeline**	144.13 days	Mon 10/2/00	Fri 4/20/01	
2	**ANALYSIS AND PLANNING PHASE**	22.25 days	Mon 10/2/00	Wed 11/1/00	
3	**Preliminary Documentation in Hand**	2 days	Mon 10/2/00	Tue 10/3/00	
4	Approved proposal in hand	0.67 days	Mon 10/2/00	Mon 10/2/00	STC-Project Manager,STC-Sales Staff,Cust-Project Manager
5	Approved scope of work in hand	0.67 days	Mon 10/2/00	Tue 10/3/00	STC-Project Manager,STC-Sales Staff,Cust-Project Manager
6	Training complete/scheduled	0.67 days	Tue 10/3/00	Tue 10/3/00	STC-Project Manager,STC-Sales Staff,Cust-Project Manager
7	**Initiate Project**	2.75 days	Wed 10/4/00	Fri 10/6/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,STC-Sales Staff,Cust-Network Administr..
8	Customer Project Kickoff Meeting	1.1 days	Wed 10/4/00	Thu 10/5/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,STC-Sales Staff
9	**Environment Analysis**	2.75 days	Wed 10/4/00	Fri 10/6/00	
10	Production, training test, hardware, software, and network requirements validated	1.1 days	Wed 10/4/00	Thu 10/5/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,STC-Sales Staff

11	Work Area	1.1 days	Thu 10/5/00	Fri 10/6/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,STC-Sales Staff
12	Establish Change Management	0.55 days	Fri 10/6/00	Fri 10/6/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,STC-Sales Staff
13	**Prepare Functional & Technical Requirements Analysis Documents**	9 days	Fri 10/6/00	Thu 10/19/00	
14	**Perform Business Rules Analysis**	9 days	Fri 10/6/00	Thu 10/19/00	
15	Identify System Requirements	0.5 days	Fri 10/6/00	Mon 10/9/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
16	Architecture model	1 day	Mon 10/9/00	Tue 10/10/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
17	Directory structure	1 day	Tue 10/10/00	Wed 10/11/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
18	Exception processing and constraints	1 day	Wed 10/11/00	Thu 10/12/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
19	Search and match specifications	1 day	Thu 10/12/00	Fri 10/13/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer

20	Interface model	1 day	Fri 10/13/00	Mon 10/16/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
21	Hardware/software model (High Level)	1 day	Mon 10/16/00	Tue 10/17/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
22	Build Document Requirements Specification	2 days	Tue 10/17/00	Thu 10/19/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
23	Requirements Specification Document approved	0.5 days	Thu 10/19/00	Thu 10/19/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
24	Review/Modify document	0.5 days	Fri 10/6/00	Mon 10/9/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
25	Finalize document	0.5 days	Fri 10/6/00	Mon 10/9/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
26	**Prepare Technical Requirements Analysis**	3 days	Fri 10/6/00	Wed 10/11/00	
27	**Perform Technical Analysis**	2 days	Fri 10/6/00	Tue 10/10/00	
28	Security requirements	1 day	Fri 10/6/00	Mon 10/9/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
29	System availability	0.5 days	Mon 10/9/00	Tue 10/10/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
30	Technology used	0.5 days	Tue 10/10/00	Tue 10/10/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer

31	Technical Requirements Document approved	0.5 days	Tue 10/10/00	Wed 10/11/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
32	Review/modify document	0.5 days	Wed 10/11/00	Wed 10/11/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
33	Finalize document	0.5 days	Wed 10/11/00	Wed 10/11/00	STC-Project Manager,Cust-Project Manager,STC Sr. System Engineer,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
34	**Identify Testing Requirements**	6.5 days	Thu 10/19/00	Mon 10/30/00	
35	Prepare testing requirements	5 days	Thu 10/19/00	Thu 10/26/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
36	Prepare Testing requirements document	0.5 days	Thu 10/26/00	Fri 10/27/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
37	Review/Modify requirements document	0.5 days	Fri 10/27/00	Fri 10/27/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
38	Finalize document	0.5 days	Fri 10/27/00	Mon 10/30/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer
39	**Technical Document Review**	2 days	Mon 10/30/00	Wed 11/1/00	
40	Review documents	2 days	Mon 10/30/00	Wed 11/1/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
41	Revise documents	1 day	Mon 10/30/00	Tue 10/31/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA

42	Documents approved	0.5 days	Mon 10/30/00	Mon 10/30/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
43	Signoff	0 days	Mon 10/30/00	Mon 10/30/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
44	**TRAINING PHASE**	17.5 days	Wed 11/1/00	Fri 11/24/00	
45	**e*Gate 4.0 Training**	12.5 days	Wed 11/1/00	Fri 11/17/00	
46	e*Gate Business/Analyst and Managers	0.5 days	Wed 11/1/00	Wed 11/1/00	Cust-Project Manager,Cust-System Engineer
47	e*Gate Basic Implementor	3 days	Wed 11/1/00	Mon 11/6/00	Cust-System Engineer
48	e*Gate Advanced Implementor	3 days	Mon 11/6/00	Thu 11/9/00	Cust-System Engineer
49	e*Gate Dart e*Ways	3 days	Thu 11/9/00	Tue 11/14/00	Cust-System Engineer
50	e*Gate Batch/TCPIP e*Ways	3 days	Tue 11/14/00	Fri 11/17/00	Cust-System Engineer
51	**e*Index Training**	5 days	Fri 11/17/00	Fri 11/24/00	
52	GUI Training	3 days	Fri 11/17/00	Wed 11/22/00	Cust-Project Manager,Cust-System Engineer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,Cust-Trainers
53	Backend Training	2 days	Wed 11/22/00	Fri 11/24/00	Cust-Project Manager,Cust-System Engineer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,Cust-Trainers
54	**DESIGN/REVIEW PHASE**	18.25 days	Fri 11/24/00	Wed 12/20/00	
55	**Preliminary Analysis Complete**	1.25 days	Fri 11/24/00	Mon 11/27/00	
56	Project successfully initiated	2 hrs	Fri 11/24/00	Fri 11/24/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA

57	Approved project planning documents available	0.5 days	Mon 11/27/00	Mon 11/27/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
58	Approved Functions/Technical Requirements specifications available	0.5 days	Mon 11/27/00	Mon 11/27/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
59	**Establish Management Control Plan**	0.5 days	Tue 11/28/00	Tue 11/28/00	
60	Configuration Management Plan	2 hrs	Tue 11/28/00	Tue 11/28/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
61	Configuration Management Plan approved	2 hrs	Tue 11/28/00	Tue 11/28/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
62	**Identify Standards**	1.25 days	Tue 11/28/00	Wed 11/29/00	
63	Prepare Standards Document	1 day	Tue 11/28/00	Wed 11/29/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
64	Standards Document approved	2 hrs	Wed 11/29/00	Wed 11/29/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
65	**Design System Architecture**	2.25 days	Wed 11/29/00	Fri 12/1/00	
66	**Design Document**	2 days	Wed 11/29/00	Fri 12/1/00	
67	Prepare Design Document	2 days	Wed 11/29/00	Fri 12/1/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
68	Design Document approved	2 hrs	Wed 11/29/00	Wed 11/29/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA

69	**Infrastructure Plan**	2.25 days	Wed 11/29/00	Fri 12/1/00	
70	Prepare Infrastructure Plan	2 days	Wed 11/29/00	Fri 12/1/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
71	Infrastructure Plan approved	2 hrs	Fri 12/1/00	Fri 12/1/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
72	**Design Programming/Mapping Specifications**	8 days	Mon 12/4/00	Wed 12/13/00	
73	Prepare Program Specification	8 days	Mon 12/4/00	Wed 12/13/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
74	Program Specification approved	0.5 days	Mon 12/4/00	Mon 12/4/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
75	**Design Unit Test Plan for Individual Interfaces**	3 days	Thu 12/14/00	Mon 12/18/00	
76	Prepare Acceptance Test Plan	2.5 days	Thu 12/14/00	Mon 12/18/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
77	Acceptance Test Plan approved	0.5 days	Mon 12/18/00	Mon 12/18/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
78	Prepare Unit Test Plan	2 days	Thu 12/14/00	Fri 12/15/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA

79	Unit Test Plan approved	0.5 days	Mon 12/18/00	Mon 12/18/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA
80	**Update/Refine Project Plan**	2 days	Tue 12/19/00	Wed 12/20/00	
81	Resolve issues	0.5 days	Tue 12/19/00	Tue 12/19/00	STC-Project Manager,Cust-Project Manager
82	Issues communicated and approved	0.5 days	Tue 12/19/00	Tue 12/19/00	STC-Project Manager,Cust-Project Manager
83	Prepare updated Project Plan	2 days	Tue 12/19/00	Wed 12/20/00	STC-Project Manager,Cust-Project Manager
84	Updated Project Plan approved	0.5 days	Tue 12/19/00	Tue 12/19/00	STC-Project Manager,Cust-Project Manager
85	**PROJECT MANAGEMENT**	121.5 days	Wed 11/1/00	Thu 4/19/01	
86	**Prepare Project Plan**	6 days	Wed 11/1/00	Thu 11/9/00	
87	Prepare Project Plan	5 days	Wed 11/1/00	Wed 11/8/00	STC-Project Manager,Cust-Project Manager
88	Issues communicated and approved	0.5 days	Wed 11/8/00	Wed 11/8/00	STC-Project Manager,Cust-Project Manager
89	Project planning documents approved	0.5 days	Wed 11/8/00	Thu 11/9/00	STC-Project Manager,Cust-Project Manager
90	**Update Project Plan/Weekly Meeting/Update Status Reports**	121.5 days	Wed 11/1/00	Thu 4/19/01	
91	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Wed 11/1/00	Wed 11/1/00	STC-Project Manager,Cust-Project Manager
92	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Wed 11/8/00	Thu 11/9/00	STC-Project Manager,Cust-Project Manager
93	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Thu 11/16/00	Thu 11/16/00	STC-Project Manager,Cust-Project Manager
94	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Thu 11/23/00	Fri 11/24/00	STC-Project Manager,Cust-Project Manager

95	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Fri 12/1/00	Fri 12/1/00	STC-Project Manager,Cust-Project Manager
96	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Fri 12/8/00	Mon 12/11/00	STC-Project Manager,Cust-Project Manager
97	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Mon 12/18/00	Mon 12/18/00	STC-Project Manager,Cust-Project Manager
98	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Mon 12/25/00	Tue 12/26/00	STC-Project Manager,Cust-Project Manager
99	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Tue 1/2/01	Tue 1/2/01	STC-Project Manager,Cust-Project Manager
100	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Tue 1/9/01	Wed 1/10/01	STC-Project Manager,Cust-Project Manager
101	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Wed 1/17/01	Wed 1/17/01	STC-Project Manager,Cust-Project Manager
102	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Wed 1/24/01	Thu 1/25/01	STC-Project Manager,Cust-Project Manager
103	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Thu 2/1/01	Thu 2/1/01	STC-Project Manager,Cust-Project Manager
104	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Thu 2/8/01	Fri 2/9/01	STC-Project Manager,Cust-Project Manager
105	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Fri 2/16/01	Fri 2/16/01	STC-Project Manager,Cust-Project Manager
106	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Fri 2/23/01	Mon 2/26/01	STC-Project Manager,Cust-Project Manager
107	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Mon 3/5/01	Mon 3/5/01	STC-Project Manager,Cust-Project Manager

108	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Mon 3/12/01	Tue 3/13/01	STC-Project Manager,Cust-Project Manager
109	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Tue 3/20/01	Tue 3/20/01	STC-Project Manager,Cust-Project Manager
110	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Tue 3/27/01	Wed 3/28/01	STC-Project Manager,Cust-Project Manager
111	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Wed 4/4/01	Wed 4/4/01	STC-Project Manager,Cust-Project Manager
112	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Wed 4/11/01	Thu 4/12/01	STC-Project Manager,Cust-Project Manager
113	Update Project Plan/Weekly Meeting/Update Status Reports	0.5 days	Thu 4/19/01	Thu 4/19/01	STC-Project Manager,Cust-Project Manager
114	**e*GATE DEVELOPMENT**	22 days	Thu 12/21/00	Fri 1/19/01	
115	**Installation Phase**	8 days	Thu 12/21/00	Mon 1/1/01	
116	Install development environment	2 days	Thu 12/21/00	Fri 12/22/00	STC-System Engineer,Cust-System Engineer
117	Development environment tested and approved	0.5 days	Mon 12/25/00	Mon 12/25/00	STC-System Engineer,Cust-System Engineer
118	Install test environment	2 days	Mon 12/25/00	Wed 12/27/00	STC-System Engineer,Cust-System Engineer
119	Configuration	3 days	Wed 12/27/00	Mon 1/1/01	STC-System Engineer,Cust-System Engineer
120	Test environment tested and approved	0.5 days	Mon 1/1/01	Mon 1/1/01	STC-System Engineer,Cust-System Engineer
121	**Development/Configuration Phase**	9 days	Tue 1/2/01	Fri 1/12/01	
122	Source System Communication program	5 days	Tue 1/2/01	Mon 1/8/01	Cust-Source System Engineer/Programmer
123	**e*Ways for External Systems**	9 days	Tue 1/2/01	Fri 1/12/01	

124	Develop inbound and outbound e*Ways	3 days	Tue 1/2/01	Thu 1/4/01	STC-System Engineer,Cust-System Engineer
125	e*Way configuration	2 days	Fri 1/5/01	Mon 1/8/01	STC-System Engineer,Cust-System Engineer
126	Translation from Event Type	4 days	Tue 1/9/01	Fri 1/12/01	STC-System Engineer,Cust-System Engineer
127	**Testing Phase**	5 days	Mon 1/15/01	Fri 1/19/01	
128	e*Gate 4.0 testing	5 days	Mon 1/15/01	Fri 1/19/01	STC-System Engineer,Cust-Source System Engineer/Programmer
129	**e*INDEX DEVELOPMENT**	72.13 days	Thu 12/21/00	Mon 4/2/01	
130	**Analysis Phase**	3.13 days	Thu 12/21/00	Tue 12/26/00	
131	Identify system(s)	2 hrs	Thu 12/21/00	Thu 12/21/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Oracle DBA,Cust-Network Administrator
132	Identify hardware	3 hrs	Thu 12/21/00	Thu 12/21/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Oracle DBA,Cust-Network Administrator
133	Identify network structure	2 hrs	Thu 12/21/00	Thu 12/21/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Oracle DBA,Cust-Network Administrator
134	Disk space analysis	2 hrs	Thu 12/21/00	Fri 12/22/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Oracle DBA,Cust-Network Administrator

135	Dial Up setup	2 days	Fri 12/22/00	Tue 12/26/00	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Oracle DBA,Cust-Network Administrator
136	**Reports**	7 days	Tue 12/26/00	Thu 1/4/01	
137	Review existing reports and identify modifications	2 days	Tue 12/26/00	Thu 12/28/00	STC-Project Manager,Cust-Project Manager,STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA
138	Modify reports	5 days	Thu 12/28/00	Thu 1/4/01	STC-Project Manager,Cust-Project Manager,STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA
139	**Installation**	11 days	Tue 12/26/00	Wed 1/10/01	
140	**Application Installation**	3 days	Tue 12/26/00	Fri 12/29/00	
141	Install database software (Oracle, Sybase, or SQL Server)	1 day	Tue 12/26/00	Wed 12/27/00	STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA
142	Install e*Index database	2 days	Wed 12/27/00	Fri 12/29/00	STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA
143	Workstation installation	1 day	Tue 12/26/00	Wed 12/27/00	STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA,Cust-Network Administrator
144	**e*Index e*Ways**	10 days	Wed 12/27/00	Wed 1/10/01	
145	Install/configure Sending e*Ways	4 days	Wed 12/27/00	Tue 1/2/01	STC-System Engineer,Cust-System Engineer
146	Install/Configure polling e*Way	4 days	Tue 1/2/01	Mon 1/8/01	STC-System Engineer,Cust-System Engineer
147	Program customized translations	2 days	Mon 1/8/01	Wed 1/10/01	STC-System Engineer,Cust-System Engineer
148	**Application Configuration**	1 day	Mon 1/8/01	Tue 1/9/01	
149	Configure system/application values, field display options, country-specific options, and candidate selection query	1 day	Mon 1/8/01	Tue 1/9/01	STC-Project Manager,Cust-Project Manager,Cust-System Administrator

150	**MPI Scan (Source System)**	23 days	Wed 1/10/01	Mon 2/12/01	
151	**Data Cleanup**	5 days	Wed 1/10/01	Wed 1/17/01	
152	Extract source system data	0 days	Wed 1/10/01	Wed 1/10/01	Cust-Source System Engineer/Programmer
153	Cleanup source system data	5 days	Wed 1/10/01	Wed 1/17/01	STC-System Engineer,Cust-System Engineer
154	**After Cleanup**	18 days	Wed 1/17/01	Mon 2/12/01	
155	Build frequency tables (optional)	5 days	Wed 1/17/01	Wed 1/24/01	Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer[50%]
156	Analyze data for building selection/matching rules	3 days	Wed 1/24/01	Mon 1/29/01	Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer[50%]
157	Define data rules	2 days	Mon 1/29/01	Wed 1/31/01	Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer[50%]
158	Build duplicate reports and define "survivorship"	3 days	Wed 1/31/01	Mon 2/5/01	Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer[50%]
159	Build MPI Scan reports	5 days	Mon 2/5/01	Mon 2/12/01	Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer[50%]
160	**Initial Data Load (Source System)**	15 days	Mon 2/12/01	Mon 3/5/01	
161	Initial data load	15 days	Mon 2/12/01	Mon 3/5/01	STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA[20%]
162	**Testing**	20 days	Mon 3/5/01	Mon 4/2/01	
163	Application testing	20 days	Mon 3/5/01	Mon 4/2/01	STC-System Engineer,Cust-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA[50%]

164	**INTEGRATION TESTING**	10 days	Mon 4/2/01	Mon 4/16/01	
165	Testing	10 days	Mon 4/2/01	Mon 4/16/01	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-Oracle DBA,Cust-Network Administrator
166	**MIGRATION TO PRODUCTION**	4 days	Mon 4/16/01	Fri 4/20/01	
167	e*Gate	2 days	Mon 4/16/01	Wed 4/18/01	STC-System Engineer,Cust-System Engineer
168	e*Index	2 days	Wed 4/18/01	Fri 4/20/01	STC-System Engineer,Cust-System Engineer,Cust-Oracle DBA[10%]
169	Go Live	0 days	Mon 4/20/01	Mon 4/20/01	STC-Project Manager,Cust-Project Manager,Cust-System Engineer,STC-System Engineer,Cust-Source System Engineer/Programmer,Cust-System Administrator,Cust-Security Administrator,Cust-Oracle DBA,Cust-Network Administrator

Implementation Survey Questionnaire

About this Appendix

This appendix contains several questions that can help you gather the information you need to plan and design your e*Index implementation. Use questionnaires or surveys to gather information from a variety of people and departments in your organization. This information will help you assess their requirements during the analysis and planning phases of the e*Index implementation (see chapters 4 and 5 of this guide). You can copy these pages and use them, or use them as a guide as you create your own questionnaire.

The questions are divided into the following sections:

- System-specific Information
- Business Rules and Technical Requirements Analysis
- Hardware and Software Analysis
- Data and Database Analysis
- Initial Load Questionnaire
- Personnel and Training
- Business Planning

System-Specific Information

What existing systems do we need to cross-index? Include systems that may be phased in at a later time.

In these systems, which enterprise applications contain demographic data?

Of these departmental applications, which are already on the network? Will any additional applications be networked by the completion of the implementation project?

Are there any systems/applications that will be phased out by the end of the implementation project?

Will any new applications/systems be implemented during the course of this project?

Of the existing applications/systems, which are already connected through e*Gate?

How do we want to connect to the systems/applications that are not already connected to e*Gate?

What person (or department) is in charge of these systems/applications?

Can we bring all of our systems online with e*Index at once, or do we need to roll out in phases? Have these phases been identified?

Does user access to e*Index information need to be restricted by system or groups of systems (called *regions*)? If so, list each system and the regions to which each belongs. Use this list to set up region-specific security.

System Name	Region Name

Business Requirements and Technical Analysis

This section of the survey will help determine your system requirements and any additional or upgraded hardware that may be required for the e*Index implementation.

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?

Is our existing equipment capable of scaling up to meet future demands?

What are our system performance requirements?

What are our error-handling and data validation requirements?

What are our internal security requirements?

How do we need e*Index to process information? The answers to this question will help determine how to set certain system parameters or how to customize the data translations.

Who is responsible for network-related issues in your organization? Are they available for regular consultation on network-related matters over the course of this installation?

Briefly describe your current networking environment. Provide any schematic diagrams available which depict your existing network.

Please note any obstacles that will deter the communication of messages from your systems to the e*Gate Integrator.

Provide any technical or other documentation describing:

- **Existing outbound interfaces.**
- **Application trigger events that generate outbound messages.**

Hardware and Software Analysis

This section of the survey will help you determine your hardware and software requirements. You need to determine these requirements for the e*Index database server, the e*Gate Participating Host (and the e*Gate Host if this is a new installation of e*Gate), and the e*Index Quality Workstations.

What kind of fault tolerance system will we use, and what level of data security is required?

What are our system and hardware limitations and constraints?

What are the processing requirements for the e*Gate server? Include the number of schemas required and the expected number of transactions flowing through each schema.

What is the format of the records transmitted by each system (for example, HL7 ADT)?

System	Message Format

What networking components are required for each server and client workstation in the e*Index/e*Gate system?

What operating systems will be used for e*Index components?

Component	Operating System
e*Gate Host server	
e*Gate Participating Host for e*Index	
e*Index database server	
e*Index client workstations	

What database platform (Oracle, Sybase, or Microsoft SQL Server) and version will the e*Index database be installed on?

How many Quality Workstations must be included in the system?

Data and Database Analysis

The questions in this section will help you analyze your database and data processing requirements. It will also determine the values you need to add to the e*Index code tables (using e*Index Data Dictionary).

Database Analysis

How many records from each system will be initially converted into the e*Index database?

System Name	Number of records
Total	

How many records are expected to be processed into the e*Index database each day from each system?

System Name	Number of records
Total	

How many records from legacy systems do you expect to load into the database at a later date? For example, if you plan to add a new system to the e*Index system next year, how many records will you load from the legacy data?

System Name	Number of records	Expected Load Date
Total		

Determine the sizing requirement for the e*Index database (non-mirrored) based on the numbers above. The <days> variable represents how far into the future you are taking into account for your sizing requirements.

Total Initial Records	
Total New Records Processed Each Day X <days>	
Total Records to be Converted in the Future	
Total	
Each person record requires 4 KB	x 4 KB
Total Database Space Required	

Data Analysis

In each legacy system, are there any known issues with the data that must be addressed prior to running the initial load?

What is the format of the local ID assigned by each system? What types of characters are allowed (alphanumeric, numeric only, alphabetic only, and so on)?

System Name	Local ID Length	Local ID Format

What system codes are used to identify each system?

System Name	System Code

Do you associate regions for your systems (for region-specific security only)? If so, what region codes are used for each region?

Region Name	Region Code

What processing codes are used by each system? Determine the codes used for each of the data element types listed in Chapter 5 under "Designing Code Table Data" (samples provided below).

Data Type	Description	Code
Citizenship	England	ENG
	Canada	CAN
	...	
Language	French	FR
	English	EN
	...	

Are there any non-unique IDs assigned to members in your organization (such as a joint account number, an insurance policy covering several family members, and so on)? If so, list each ID type, an identifying code for the type, and the format of the IDs.

Non-unique ID Type	Non-unique ID Type Code	ID Format

In what country do most of the addresses in your databases originate? Are there any unusual abbreviations or street types in those addresses? Use this information to define the COUNTRY control key and to modify address-parsing attributes if needed.

What data fields should be used for the original record comparison made by e*Index when selecting a pool of possible matches to incoming data?

What data fields are (a) required, or (b) captured but not required, in your current systems? Please use the grid on the following page to document current practices for each system. Modify the data element names as needed and use this list to define your field display configuration.

- **Are any other data elements other than those listed used routinely? Are there other data elements you feel are critical or helpful for consistent patient identification?**
- **Indicate required data elements with "R" and routinely captured data items with "C".**

Note: *You can change the names of the field labels in e*Index, so if you can change the name of the data elements below and store the most pertinent information to your business.*

Data Element	**System 1**	**System 2**	**System 3**	**System 4**
Local identifier (and assigning system)				
Non-unique IDs (and the type of ID)				
Person category				
Last name				
First name				
Middle name / initial				
Suffix				
Title				
Gender				
Date of birth				
Death indicator				
Social security number				
Alias name(s)				
Street address				
City				
State				
Zip/postal code				
Country				
Mothers maiden name				
Day telephone number				
Evening telephone number				
Marital status				
Race				
Ethnicity				
Religion				

Data Element	**System 1**	**System 2**	**System 3**	**System 4**
Language				
Spouse name				
Mother's name				
Father's name				
City of birth				
State of birth				
Country of birth				
VIP flag				
Veterans status				
Maiden name				
Advance directive				
Driver license issuer				
Driver license state				
Date of death				
Death certificate number				
Nationality				
Citizenship				
Pension number				
Pension expiration date				
Repatriation number				
District of Residence				
LGA code				
Military rank				
Military status				
Military branch				
Other________________				
Other________________				
Other________________				

Of the data elements listed above, which elements from each system will be stored in the e*Index database?

Describe the current methods used for searching for person records. Include the data elements used, search parameters, abilities to limit searches by users, and related functions.

Are these current search systems adequate? If lacking, in what ways?

What types of quality assurance procedures are currently in place to ensure accuracy in person identification and demographic data?

What procedures are in place for the identification and resolution of duplicate local identifiers? When duplicate medical record numbers are deactivated, can they be reused? If not, are they physically removed from the database?

How are unidentified persons designated (i.e., John/Jane Doe)? Are unidentified patients entered into the system and allocated local identifiers? What follow-up procedures exist to gather actual person information?

Are there other areas within the organization whereby "pseudo" names and numbers must be assigned for non-identifiable persons?

For medical sites, describe procedures for admitting maternity and neonatal cases. Is the mother's MPI data used to create the newborn record? If yes, itemize those data items that are replicated, and describe procedures used to gather and record information unique to the infant (i.e., social security number).

Which systems are most likely to create duplicate person records? Please describe why, and what, if any, procedures are in place to minimize this problem.

What are the coverage hours for the records or IS management department? If not 7 x 24, what procedures exist for handling duplicate records during off shift hours?

Are reports produced routinely for merged person records? If yes, how is the report used (i.e., by other departments) and how frequently is it produced? What procedures exist to ensure all appropriate departments are informed of record merges?

How are international details recorded? Are there special processing procedures for members with non-U.S. addresses?

Please describe your downtime registration procedures. How are local identifiers identified and assigned during downtime, and how are they input into the system when it is brought back up?

Does your organization perform any institutional billing/processing? If yes, how are persons identified in that workflow?

Are there persons in your MPI with marginal, outdated, or minimal data because of their age, prior systems, or other reasons? If these persons return, is their MPI data updated, or is a new identifier assigned?

____

____

____

____

____

Describe any unusual procedures or practices regarding how local identifiers are issued or how individuals are identified.

____

____

____

____

____

Please list any default values used for the following data elements:

Data element	**Default value(s)**
Last name	
First name	
Date of birth	
Gender	
Social security number	
Address	
Telephone numbers	
Marital status	
Race	
Language	
Zip code	
Other_______________	
Other_______________	
Other_______________	

Initial Data Load Data Questionnaire

Before designing the initial load procedure, you need to determine how certain data issues should be handled, such as the use of default values for dates of birth and SSNs, missing transaction dates, and so on. The information you obtain from the data analysis questions earlier in this chapter will also be helpful for the initial load design.

What systems have legacy data that needs to be loaded into the e*Index database, and how many records will be loaded from each system?

System Name	Number of Initial Load Records
Total Initial Load Records	

What default values are used for the SSN field, and do you want to allow these default values to be loaded into the database or converted to null?

Default Values	Allowed	Converted to Null

What default values are used for the DOB field, and do you want to allow these default values to be loaded into the database or converted to null?

Default Values	Allowed	Converted to Null

Are any of the records to be loaded missing a transaction date? If so, how should updates to the database be handled (records are inserted or updated based on the transaction date)?

Personnel and Training

This section of the survey will help you determine the requirements of the e*Index implementation team and whether you have the personnel required. It will also help pinpoint any training requirements.

Do we have personnel trained and able to deploy the system and what are their names?

Position	Tasks	Name

Do we have personnel trained and able to maintain the system after deployment and what are their names?

Position	Tasks	Name

Do we need to contract or hire any additional personnel to assist with deployment and maintenance of the system?

Position	Tasks	Name

How many employees need to be trained on the Quality Workstation and what are their names?

____

____

____

____

How many employees need to attend SeeBeyond training classes? What are their names and which classes do they need to attend? Include the dates by which these classes must be completed. Recommended classes for e*Index include e*Gate Database e*Ways, e*Gate Basic, and e*Index.

Name	Class	Complete By

Have we scheduled any necessary training classes with outside vendors? Who are the employees scheduled for these classes?

Name	Class	Complete By

Who will serve as the primary technical resource(s) for the e*Index implementation for your organization?

____

____

____

____

____

Do you have in-house resources capable of making slight modifications to your source applications and related products (i.e., extract files), or are these services outsourced to other organizations?

____

____

____

____

____

Do you possess any in-house interface development or support expertise?

____

____

____

____

____

Business Planning

The answers to the following questions will help you determine the scope of documentation required for the e*Index implementation project. It should also give you an idea of monitoring requirements for system maintenance.

What are our record-keeping needs during the implementation? After the implementation?

What are our documentation needs during the implementation? After the implementation?

What steps are required to create an implementation project plan?

What important time/schedule and resource allocation issues do we need to take into account in creating our implementation project plan?

Describe any other known projects or activities occurring within the timeframe of the e*Index implementation that might impact this project.

What business processes do we want e*Index to help us enable? Do we have any documents that describe these 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, or will additional facilities be required?

Are there additional issues that are relevant to this implementation that were not addressed earlier in this questionnaire? Provide information about these issues.

Sample Hardware Worksheets

About this Appendix

The following pages contain sample hardware and transaction worksheets. Use this information to help you determine your hardware requirements during the analysis and planning phases of the e*Index Global Identifier implementation project (see chapters 4 and 5 of this guide for information about these two phases). The transactions worksheet may be helpful when determining tablespace sizing, extent numbers and sizing, and index extents for the e*Index database. You can copy these pages and use them, or you can use the worksheets as guides to help you create your own worksheets.

Hardware Worksheet

General Questions

Number of Users

Maximum number of users that will be working on the system at one time.

Database Size

Include log files in approximation of database size.

Fault Tolerance Operating System

Data security level required for the operating system. Choose from RAID-0, RAID-1, or RAID-5. The standard configuration uses RAID-5.

Fault Tolerance Data Files

Data security level required for the data files. Choose from RAID-0, RAID-1, or RAID-5. The standard configurations uses RAID-5.

Time Window

The span of time during which the transaction loads on the system will be estimated.

Advanced Questions

Expected Cache Hit Rate

The cache hit rate that would be typical of the system and application.

Optimal CPU Utilization

The target CPU utilization in percent of total processing power. Make sure to allow for future growth.

Optimal Disk I/O Utilization

The highest acceptable percentage of disk I/O capacity to be filled by the workload. Make sure to allow for future growth.

Transactions Worksheet

Transaction Number: _________ **Transaction Name:** __________________________

Table Number	Rows Selected	Rows Updated	Rows Inserted	Total Rows	Index Factor	Reads	Writes
					Total Reads and Writes		

Fill out one sheet per transaction. Each transaction can consist of multiple SQL statements. Each table that is touched by the transaction should be posted on a separate line. If a table is touched more than once by different SQL statements, list each occurrence separately.

- Column 1: List table number
- Column 2: Enter the number of rows that will be selected for each table in column 1
- Column 3: Enter the number of rows that will be updated for each table
- Column 4: Enter the number of rows that will be inserted for each table
- Column 5: Enter the total number of rows in each table listed in column 1
- Column 6: To find the index factor, use the following guides:
  - Index factor = 5 when selected rows are accessed randomly according to the index (use 6 for large tables)
  - Index factor = 4 when inserted or updated rows are accessed randomly according to the index (use 5 for large tables)
  - Index factor = 2.5 when selected rows are accessed sequentially according to the index (use 2.8 for less than 10 rows and 2.3 for greater than 100)
  - Index factor = 1.5 when inserted or updated rows are accessed sequentially according to the index
- Column 6: Calculate the reads by multiplying the Rows Selected by the Index Factor
- Column 7: Calculate the writes as the sum of the Rows Updated or 2 times the Rows Inserted

Total the Reads and Writes columns. These values will be used to input the transaction.

Glossary of Terms

API

An acronym for Application Program Interface, which is a set of protocols, routines, and tools for building software applications. The e*Index API consists of a set of Monk functions that can be called from Collaborations to process data into the database.

assumed match

When the matching probability weight between two profiles is above the match threshold and e*Index is configured to merge records with a matching probability above the match threshold, the two profiles are assumed to represent the same person. These records are automatically merged because they are an assumed match.

alphanumeric search

A type of search in e*Index that looks for profiles that precisely match your specified criteria. This type of search does not allow for misspellings or data entry errors.

audit trail

A stored history of a member's profile. This history displays changes made to a member's demographic information as well as merges, unmerges, adds, and so on. Each time a member's information changes, a new audit trail record is written to the database.

batch processes

When updates are performed in batch processes, several updates from external systems are stored in a batch file. At a specified time, the updates stored in the batch file are processed through the e*Index database.

Business Object Broker (BOB)

Like e*Ways, BOBs use Collaborations to route and transform data within the e*Gate system. Unlike e*Ways, BOBs cannot exchange information directly with external systems. BOBs are optional features.

code tables

Tables in the e*Index database that store the data that you use to populate the values in the drop-down lists of the e*Index GUI. They also store processing codes for these data elements. e*Index uses the processing codes to translate the codes from external Events into their descriptions. For example, if the language field in a member record is "FRN", then e*Index translates it to "French" in the e*Index GUI windows.

Collaboration
A component of an e*Way or BOB that receives and processes Events and forwards the output to other e*Gate components. Collaborations subscribe to Events of a known type, apply business rules to Event data, and publish output Events to a specified recipient. Collaborations use Monk translation script files with the extension "**.tsc**" to do the actual data manipulation.

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.

control keys
A set of data elements in e*Index Administrator that allows you to configure the system parameters of your e*Index environment. System parameters include search attributes and limits, display options, debugging options, processing options, matching thresholds, and so on.

drop-down lists
A list of options that you can use to enter information into certain fields in the e*Index GUIs. These fields have a down-arrow button on their right side that, when clicked, display a drop-down list of options for that field. The elements in each drop-down list come from the code table data you enter.

e*Index Administrator
A GUI application within e*Index that allows you to modify system parameter values, insert and update code table values, create predefined comments that users can assign to member profiles, and update zip code information.

e*Index Global Identifier GUI
A GUI application within e*Index that allows you to monitor and maintain member information in the e*Index database. This application provides the ability to monitor and resolve potential duplicate profiles, insert and update profiles, unmerge merged profiles, print reports, view audit trails and profile comparisons, search for profiles, and deactivate profiles. Sometimes called simply the e*Index GUI.

e*Index Security
A GUI application within e*Index that allows you to set up and configure security for e*Index. This includes creating user profiles and user groups, assigning access permissions to user profiles and user groups, and assigning user profiles to user groups.

e*Way Intelligent Adapter (e*Way)

A component that provides a noninvasive point of contact between an e*Gate system and an external application. e*Ways establish connectivity with applications, using the appropriate communication protocol. e*Ways receive unprocessed data from external components, transform it into Events, and forward it to other components within e*Gate via Intelligent Queues (IQs). They also send processed data to external components, which can also include data transformation.

entity

A specific instance of an entity type. For example, information about John Smith is a specific instance (an entity) within a set of members (entity type). Information about local ID record is a specific instance (an entity) within the set of local ID records (entity type).

entity type

An object of interest about which data can be collected and processed. In e*Index, entity types include users, user groups, members, and so on.

Event (message)

Data to be exchanged between e*Index and external systems and that has a defined data structure, such as a known number of fields with known characteristics and delimiters. Events are classified by type using Event Type Definitions.

Event Type Definition (ETD)

An Event Type template that defines Event fields, field sequences, and delimiters. Event Type Definitions enable e*Index systems to identify and transform Event Types. They are Monk script files with an extension of "**.ssc**," short for message structure script file.

Intelligent Queue (IQ)

A standard e*Gate component that manages the exchange of information between components within the e*Gate system, providing nonvolatile storage for data as it passes from one component to another.

local ID

A unique identification code assigned to a member in a specific system. A member may have local IDs at several different systems. e*Index uses the local IDs to cross-reference member records.

master person index

A database that stores information on all of the members of a business enterprise, regardless of the system or location from which the member information originates. e*Index is a master person index.

matching probability weights

An indicator of how closely two member profiles match one another. The weight is generated using matching algorithm logic, and is used to determine whether two profiles represent the same person.

member

Any person who participates within your business enterprise and whose information you store in a computer application. A member could be a customer, employee, patient, and so on.

member profile

A set of information that describes characteristics of an individual member. A profile includes demographic, alias, and identification information about the member.

merge

To join two member profiles that represent the same person into one member profile. You can choose which information from each profile will be saved in the new member profile. By identifying and merging potential duplicate records, you can ensure that your data remains current and accurate.

Monk

SeeBeyond's implementation of the Scheme programming language; the Monk programming language is used to configure various parts of e*Gate.

Participating Host

A client computer that supports an e*Gate system, as opposed to the Registry Host, which acts as a server to the Participating Host.

phonetic search

A search that returns phonetic variations of the entered name, allowing room for misspellings and typographic errors. This type of search can be performed in the Demographic Search section of the Inquiry window in the e*Index GUI. Phonetic searches are also performed when Events are sent to e*Index via the sending e*Way in order to check for potential matches.

potential duplicates

Two different member profiles that have a high probability of representing the same person. The matching algorithm determines the matching probability, and you define the probability weight at which two records are considered potential duplicates.

probability Weight

See "Matching Probability Weight."

predefined comment

A comment that has been previously defined by the System Administrator. To associate a predefined comment with a member, you select a predefined comment from a drop down list of options. Predefined comments are stored in the database table *ui_canned_msg*.

Quality Workstation

A computer workstation on which the e*Index GUIs reside. Use these workstations to monitor and maintain the data in the e*Index database.

real-time

When updates occur in real-time, the e*Index database is modified at the time the updates occur in your external systems.

schema

Includes files and associated stores created by e*Gate that contain the parameters of all the components that control, route, and transform data as it moves through e*Gate in a predefined system configuration.

system

A location within your company at which member information is entered. Systems may include physical store locations, company branch locations, websites, and so on. Systems assign members a unique local ID for identification purposes.

system administrator

The person who maintains the security and database integrity for your e*Index system.

UID

The global identification number assigned to each member added to the e*Index application. This number is used to uniquely identify each member across a business enterprise by cross-referencing the associated local ID assigned by the systems within the organization.

unmerge

To unmerge two previously merged records. When you perform an unmerge, each profile is returned to its status prior to the merge. Any updates made while the records were merged are lost when the records are unmerged.

user group

An e*Index Security entity that allows you to grant the same access permissions to a set of users with similar processing needs without having to specify individual privileges for each user. For example, you may have a set of users who can only view member information, but cannot update or merge member profiles. These users can be assigned to a user group that has only been granted view privileges.

user profile

An e*Index Security entity that describes characteristics of an individual user of e*Index Global Identifier. A profile includes a user's name, login information, user type, and so on. You can assign a user profile to a user group to grant them access permissions, or you can grant access permissions on an individual basis.

Index