

SUN SEEBEYOND
eINDEX™ SINGLE PATIENT VIEW
USER'S GUIDE

Release 5.1.1



Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved. Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries. U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. This distribution may include materials developed by third parties. Sun, Sun Microsystems, the Sun logo, Java, Sun Java Composite Application Platform Suite, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM, eWay, and JMS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

Copyright © 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés. Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays. L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties. Sun, Sun Microsystems, le logo Sun, Java, Sun Java Composite Application Platform Suite, Sun, SeeBeyond, eGate, eInsight, eVision, eTL, eXchange, eView, eIndex, eBAM et eWay sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd. Ce produit est couvert à la législation américaine en matière de contrôle des exportations et peut être soumis à la réglementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

Version 20060619183110

Contents

List of Figures 10

Chapter 1

Introduction	12
About eIndex Single Patient View	12
eIndex Single Patient View Features	13
What's New in This Release	13
About This Document	13
What's in This Document	14
Scope	14
Intended Audience	15
Text Conventions	15
Screenshots	15
Related Documents	15
Sun Microsystems, Inc. Web Site	16
Documentation Feedback	16

Chapter 2

eIndex Single Patient View Overview	17
About eIndex Single Patient View	17
eIndex SPV and the Sun Java Composite Application Platform Suite	18
eGate Integrator	18
eInsight Business Process Manager	18
eIndex SPV Repository Components	18
Editors	18
Project Components	19
Configuration Files	20
Database Scripts	21
Custom Plug-ins	21
Match Engine Configuration Files	22
Outbound Object Type Definition (OTD)	22
Dynamic Java API	22
Connectivity Components	22
Deployment Profile	23

Environment Components	23
About the Runtime Environment	23
Functions of the Runtime Environment	23
Runtime Environment Features	24
Runtime Environment Components	25
Matching Service	27
eIndex Manager Service	27
Query Builder	27
Query Manager	27
Update Manager	27
Object Persistence Service (OPS)	28
Database	28
Enterprise Data Manager	28
Enterprise Records	28
System Records	28
The Single Best Record	29
Objects in an Enterprise Record	29
From the Project to the Runtime Environment	29
Process Overview	29
From XML to the Database	30
From XML to the Enterprise Data Manager	30
From XML to the Connectivity Components	30
From XML to the Runtime Environment	31
Working with Project Components	31
Version Control	31
Saving a Configuration File to the Repository	31
Validating XML Files	32
Copying, Cutting, and Pasting Files	32

Chapter 3

Installation	33
Installation Overview	33
eIndex SPV Installation	33
Database Installation	34
About the Installation	34
Supported Operating Systems	34
System Requirements	35
Database Requirements	35
Software Requirements	35
Before Installing eIndex SPV	36
Installing eIndex SPV	36
Uploading eIndex SPV Components to the Repository	36
Downloading eIndex SPV Components from the Repository	40
Installing the eIndex SPV Monitor Client	40
Downloading eIndex SPV Reports	42

Accessing the eIndex SPV Documentation	42
Installing eIndex SPV in Enterprise Designer	43
Final Steps	49
For the Oracle eWay	49
For the Sun Java System Application Server	49

Chapter 4

Configuring eIndex SPV	50
Configurable Options	50
Object Definition	50
Enterprise Data Manager	51
Query Definitions	51
Standardization and Matching Rules	51
Survivor Calculator	51
Update Policies	52
Field Validations	52
Enterprise-wide Unique Identifiers	52
About the eIndex SPV Configuration Files	52
Object Definition File	53
Candidate Select File	53
Match Field File	53
Threshold File	53
Best Record File	54
Field Validation File	54
Security File	54
Enterprise Data Manager File	54
Modifying the eIndex SPV Configuration Files	54
Match Engine Configuration Files	55

Chapter 5

Creating Custom Plug-ins	56
Custom Processing	56
Update Policies	56
Enterprise Merge Policy	57
Enterprise Unmerge Policy	57
Enterprise Update Policy	57
Enterprise Create Policy	57
System Merge Policy	57
System Unmerge Policy	58
Undo Assumed Match Policy	58
Field Validations	58
Match Processing Logic	58
Execute Match Methods	58
Custom Logic Methods	59
Custom Logic Plug-in Requirements	59

Threshold File Customization	59
Custom Components	60
Survivor Calculator	60
Query Builder	60
Block Picker	61
Pass Controller	61
Match Engine	61
Standardization Engine	62
Phonetic Encoders	62
Custom Plug-in Exception Processing	62
Implementing Custom Plug-ins	62
Creating Custom Plug-ins	62
Building Custom Plug-ins	63

Chapter 6

Building the Application	64
Generated Application Components	64
Generating the Application	65

Chapter 7

Creating the Database	67
Database Scripts	67
Requirements	67
Database Platform Requirements	68
Operating System Requirements	68
Hardware Requirements	68
Database Structure	69
Designing the Database	69
Designing for Performance Optimization	69
Data Analysis	69
Common Table Data	70
User Code Data	70
Considerations	70
Database Sizing	70
Database Distribution	71
Database Indexes	71
Creating the Database	71
Step 1: Analyze the Database Requirements	71
Step 2: Create an Oracle Database and User	72
Step 3: Customize the Database Scripts	72
Defining Indexes	72
Defining Systems	73
Defining Code Lists	75

Defining User Code Lists	76
Creating a Custom Script	77
Step 4: Modify the Database	77
Step 5: Specify a Starting EUID (optional)	79
Deleting Tables and Indexes	79

Chapter 8

Defining Connectivity Components 80

Connectivity Overview 80

Connectivity Components	80
eIndex SPV Server Project Connectivity Components	81
Client Project Connectivity Components	81

Defining Connectivity Components 82

Defining eIndex SPV Application Connectivity Components	83
Configuring the Server Connectivity Map	83
Adding a JMS Topic to the Server Connectivity Map	85
Defining the Collaboration Client Connectivity Components	86
Modifying the Java Collaboration	86
Modifying the Collaboration Client Project Connectivity Map	88
Configuring the Outbound Collaboration	92
Customizing eInsight Client Connectivity Components	93
Modifying the Business Process	94
Modifying the eInsight Client Connectivity Map	95

Chapter 9

Deploying the Project 99

Deployment Overview 99

Creating a Domain 99

Defining Deployment Profiles 102

Deploying the eIndex SPV Server Project	103
Creating the Deployment Profile	103
Mapping Project Components to Environment Components	105
Building and Deploying the Server Project	107
Deploying the Collaboration Client Project	107
Creating a Collaboration Client Deployment Profile	108
Mapping Collaboration Client Project Components	109
Building and Deploying the Collaboration Client Project	110
Deploying the eInsight Client Project	111
Creating an eInsight Client Deployment Profile	111
Mapping eInsight Client Project Components	112
Building and Deploying the eInsight Client Project	113

Defining Security 114

Defining Security (for the Sun SeeBeyond Integration Server)	114
Defining Security (for the Sun Java System Application Server)	117

Chapter 10
Defining the Environment 119
Environment Components 119
Building an Environment 120

Creating the Environment 120

Adding a Logical Host 121

Adding Servers 122

Adding a Sun SeeBeyond Integration Server and JMS IQ Manager 122

Adding Sun Java System Servers 123

Adding an External System 124

Adding an Oracle External System 125

Configuring a Connection Pool Through the Server 126

Step 1: Create a JDBC Connection Pool 127

Step 2: Create a JDBC Resource 127

Chapter 11
Maintenance Tasks 129
Archiving Repository Information 129
Maintaining the Database 130

Backing up the Database 130

Online Backups 130

Offline Backups 130

Database Restoration 130

Archiving 131

Monitoring Day to Day Activity 131

Enterprise Manager Monitor 131

About the Enterprise Manager Monitor 131

Enabling Monitoring for eIndex SPV 132

Log Files 132

Alerts 133

Implementing Changes to the eIndex SPV Project 137

Modifying Configuration Files 137

Updating the Object Structure 138

Updating Normalization and Standardization Structures 138

Updating the Match String 138

Modifying Standard Project Components 138

Modifying the Database 139

Modifying Security 139

Modifying the Local ID Format 139

Appendix A

Field Notations	140
ePath	140
Syntax	140
Example	141
Qualified Field Names	142
Syntax	142
Example	143
Simple Field Names	143
Syntax	143
Example	144

Appendix B

Initial Load Tips	145
Legacy Data	145
Database Indexes	145
Threshold Parameters	146
Reports	146
Glossary	147
Index	153

List of Figures

Figure 1	eIndex SPV Project and Environment Components	20
Figure 2	eIndex SPV Runtime Environment Architecture	26
Figure 3	Java CAPS Login Page	37
Figure 4	Administration Page	38
Figure 5	List of Products to Install	38
Figure 6	Selecting Files to Install	39
Figure 7	Installation Page	40
Figure 8	Downloads Tab	41
Figure 9	Update Center wizard - Select Update Category	44
Figure 10	Update Center Wizard - Select Modules to Install	45
Figure 11	Update Center Wizard - License Agreement	46
Figure 12	Update Center Wizard - Download Modules	47
Figure 13	Update Center Wizard - View Certificates and Install Modules	48
Figure 14	Update Center Wizard - Restart the IDE	48
Figure 15	Generate Context Menu	65
Figure 16	Database Properties Dialog	78
Figure 17	eIndex SPV Server Service Binding Dialog	83
Figure 18	eIndex SPV Server Connectivity Map Connections	84
Figure 19	eIndex SPV Server Connectivity Map with Topic	86
Figure 20	eIndex SPV Methods in Collaboration Editor	87
Figure 21	Collaboration Client Service Binding Dialog	88
Figure 22	Collaboration Client Connectivity Map Connections	89
Figure 23	eIndexClient Connectivity Map	90
Figure 24	eIndexClient Connectivity Map with JMS Topic	91
Figure 25	eIndexClient Connectivity Map with Connections	92
Figure 26	Outbound Java Collaboration	93
Figure 27	eIndex SPV Methods in a Business Process	95
Figure 28	eInsight Client Service Binding Dialog	96
Figure 29	eInsight Client Connectivity Map Connections	97
Figure 30	eIndexClient Connectivity Map	98
Figure 31	Domain Manager	100
Figure 32	Create Domain Dialog Box	101
Figure 33	Create Deployment Profile Dialog Box	104
Figure 34	Deployment Editor - Server Project	105

List of Figures

Figure 35	Mapped eIndex SPV Components in the Deployment Editor	106
Figure 36	Automap Options Dialog	107
Figure 37	Create Deployment Profile Dialog Box	108
Figure 38	Deployment Editor - Client Project	109
Figure 39	Mapped Client Components in the Deployment Editor	110
Figure 40	Create Deployment Profile Dialog Box	111
Figure 41	Deployment Editor Window	112
Figure 42	Mapped Client Components in the Deployment Editor	113
Figure 43	Add/Edit User Window	115
Figure 44	New Environment	121
Figure 45	eIndex SPV Logical Host	122
Figure 46	Integration and JMS Servers	123
Figure 47	Application and JMS Servers	124
Figure 48	File External System	125
Figure 49	Oracle External System	126
Figure 50	Monitoring eIndex SPV Project Components	132
Figure 51	eIndex SPV Database Logging	133
Figure 52	Alert for eIndex SPV Collaboration	133

Introduction

This chapter provides an overview of this guide and the conventions used throughout, as well as a list of supporting documents and information about using this guide.

What's in This Chapter

- [About eIndex Single Patient View](#) on page 12
- [eIndex Single Patient View Features](#) on page 13
- [What's New in This Release](#) on page 13
- [About This Document](#) on page 13
- [Related Documents](#) on page 15
- [Sun Microsystems, Inc. Web Site](#) on page 16
- [Documentation Feedback](#) on page 16

1.1 About eIndex Single Patient View

The Sun SeeBeyond eIndex™ Single Patient View (eIndex SPV) provides a flexible framework to design and configure an enterprise-wide person master index that creates a single view of person information. eIndex SPV maintains the most current information about the people who participate throughout your organization and links information from different locations and computer systems. eIndex SPV provides accurate identification of patients throughout your healthcare enterprise, and cross-references a patient's local IDs using an enterprise-wide unique identification number (EUID). eIndex SPV also ensures accurate patient data by identifying potential duplicate records and providing the ability to merge or resolve duplicate records. All patient information is centralized in one shared index, enabling eIndex SPV to integrate data throughout the enterprise while allowing local systems to continue operating independently.

In eIndex SPV, you define the data structure of the information to be stored and cross-referenced. In addition, you define the logic that determines how data is updated, standardized, weighted, and matched. The structure and logic you define is stored in a group of XML configuration files, which are predefined but can be customized to meet your processing requirements. These files are defined within the context of an eGate Project and can be modified using the XML editor provided in the Enterprise Designer.

1.2 eIndex Single Patient View Features

eIndex SPV provides features and functions that allow you to customize the data structure, database, and logic of the master person index. eIndex SPV provides the following features:

- **Rapid Development** - eIndex SPV allows for rapid and intuitive development of a master person index, providing a predefined template that can easily be configured for your use.
- **Automated Component Generation** - eIndex SPV can regenerate scripts that create the appropriate database schemas and an eGate Object Type Definition (OTD) based on the changes you make to the object structure.
- **Configurable Survivor Calculator** - eIndex SPV provides predefined strategies for determining which field values to populate in the single best record (SBR). You can define different survivor rules for each field and you can create a custom survivor strategy to implement in the master index.
- **Flexible Architecture** - eIndex SPV provides a flexible platform that allows you to customize the object structure, allowing you to design an application that specifically meets your data processing needs.
- **Configurable Matching Algorithm** - eIndex SPV provides standard support for the Sun SeeBeyond Match Engine (SBME) and also provides the ability to plug in a custom matching algorithm of your choice.
- **Custom Java API** - eIndex SPV generates a Java API that is customized to the object structure you define. You can call the methods in this API in the Collaborations and Business Processes that define the transformation rules for data processed by the master index.

1.3 What's New in This Release

This release provides support for Oracle 10g, along with performance enhancements, general maintenance fixes, and enhancements to the Sun SeeBeyond Match Engine. For complete information about the changes included in this release, see the *eIndex Single Patient View Release Notes*.

1.4 About This Document

This guide provides comprehensive information on installing and working with eIndex SPV. This guide explains how to install and customize the components of eIndex SPV, including eGate Project files, the master index database, the runtime environment, and the Enterprise Data Manager (EDM). It also includes information about the architecture and components of eIndex SPV. This guide is intended to be used with the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*, *Sun SeeBeyond eIndex Single*

Patient View Reference Guide, and Implementing the Sun SeeBeyond Match Engine with eIndex SPV.

1.4.1 What's in This Document

This guide is divided into the chapters and appendixes that cover the topics shown below. The chapters are presented in the order in which you would normally perform the steps to create an eIndex SPV application.

- **Chapter 1 “Introduction”** gives a general preview of this document—its purpose, scope, and organization—and provides sources of additional information.
- **Chapter 2 “eIndex Single Patient View Overview”** provides information about the architecture of eIndex SPV and the runtime environment, and describes how the runtime environment is created.
- **Chapter 3 “Installation”** gives instructions for installing the eIndex SPV files and setting up the environment for eIndex SPV and the runtime environment.
- **Chapter 4 “Configuring eIndex SPV”** gives a summary of the configuration files in the eIndex SPV Project, and provides information about the XML Editor.
- **Chapter 5 “Creating Custom Plug-ins”** describes how to implement custom processing code in the master index.
- **Chapter 6 “Building the Application”** explains how to generate the eIndex SPV application to update the application files with your customizations.
- **Chapter 7 “Creating the Database”** describes how to design, install, and configure the Oracle database for the master index.
- **Chapter 8 “Defining Connectivity Components”** describes how to work with the connectivity components of the eIndex SPV Project to share and process data through the eIndex SPV system.
- **Chapter 10 “Defining the Environment”** describes how to set up the physical environment of the master index.
- **Chapter 9 “Deploying the Project”** explains how to create and deploy the Deployment Profile for the master index.
- **Chapter 11 “Maintenance Tasks”** provides information on backing up the master index and using the monitor to view logs and alerts.
- **Appendix A “Field Notations”** describes the different notations used in the configuration files to define different fields in the messages being processed and stored.
- **Appendix B “Initial Load Tips”** provides guidelines to help the initial process of loading date run smoothly.

1.4.2 Scope

This guide provides step-by-step instructions for installing eIndex SPV and customizing the components. It includes navigational information, functional instructions, and background information where required. This guide does not include

information or instructions on using the EDM or eGate Integrator. These topics are covered in the appropriate user guide (for more information, see “[Related Documents](#)” on page 15).

1.4.3 Intended Audience

Any user who installs any component of eIndex SPV, or customizes any of the components, should read this guide. A thorough knowledge of eIndex SPV is not needed to understand this guide. It is presumed that the reader of this guide is familiar with the eGate environment and GUIs, eGate projects, Oracle database administration, and the operating system(s) on which eGate and the index database run. Readers who will configure the master index should also be familiar with XML documents, the SQL scripting language, and Java. The intended reader must have a good working knowledge of his or her company’s current business processes and information system (IS) setup.

1.4.4 Text Conventions

The following conventions are observed throughout this document.

Table 1 Text Conventions

Text Convention	Used For	Examples
Bold	Names of buttons, files, icons, parameters, variables, methods, menus, and objects	<ul style="list-style-type: none"> ▪ Click OK. ▪ On the File menu, click Exit. ▪ Select the eGate.sar file.
Monospaced	Command line arguments, code samples; variables are shown in <i>bold italic</i>	<code>java -jar <i>filename</i>.jar</code>
Blue bold	Hypertext links within document	See Text Conventions on page 15
<u>Blue underlined</u>	Hypertext links for Web addresses (URLs) or email addresses	http://www.sun.com

1.4.5 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document might differ from what you see on your system.

1.5 Related Documents

Sun has developed a suite of user’s guides and related publications that are distributed in an electronic library. The following documents might provide information useful in

creating your customized index. In addition, complete documentation of the eIndex SPV Java API is provided in Javadoc format.

- *Sun SeeBeyond eIndex SPV Enterprise Data Manager User's Guide*
- *Sun SeeBeyond eIndex Single Patient View Configuration Guide*
- *Sun SeeBeyond eIndex Single Patient View Reference Guide*
- *Implementing the Sun SeeBeyond Match Engine with eIndex SPV*
- *Sun SeeBeyond eGate Integrator User's Guide*
- *Sun SeeBeyond eGate Integrator System Administration Guide*
- *Sun SeeBeyond eInsight Business Process Manager User's Guide*

1.6 Sun Microsystems, Inc. Web Site

The Sun Microsystems web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

<http://www.sun.com>

1.7 Documentation Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

CAPS_docsfeedback@sun.com

eIndex Single Patient View Overview

This chapter provides information about eIndex SPV, how the Project defines a master patient index, and the master patient index itself. It includes descriptions of the files stored in the eGate Repository, the configuration files that define the structure and configuration of eIndex SPV, and the runtime components.

What's in This Chapter

- [About eIndex Single Patient View](#) on page 17
- [eIndex SPV and the Sun Java Composite Application Platform Suite](#) on page 18
- [eIndex SPV Repository Components](#) on page 18
- [About the Runtime Environment](#) on page 23
- [Runtime Environment Components](#) on page 25
- [Enterprise Records](#) on page 28
- [From the Project to the Runtime Environment](#) on page 29
- [Working with Project Components](#) on page 31

2.1 About eIndex Single Patient View

In today's business environment, important information about the patients in your organization might exist in many disparate information systems. It is vital that this information flow seamlessly and rapidly between departments and systems throughout the entire business network. As organizations grow, merge, and form affiliations, sharing data between different information systems becomes a complicated task. eIndex SPV can help you manage this data and ensure that the data you have is the most current and accurate information available.

eIndex SPV allows you to design, configure, and create a custom matching and indexing application known as a master patient index. The master index uniquely identifies and cross-references the patient records throughout your organization to ensure that the most current and accurate data is available. eIndex SPV uses the local identifiers generated by your internal systems to create an index of patient records. With eIndex SPV, you define the data structure of the information to be stored and cross-referenced as well as the logic that determines how data is updated, standardized, weighted, and matched.

2.2 eIndex SPV and the Sun Java Composite Application Platform Suite

The eIndex SPV application is created within the structure of an eGate Project in Enterprise Designer. eIndex SPV is tightly integrated within the Sun Java Composite Application Platform Suite (Java CAPS), and can leverage the features of other Java CAPS components.

eGate Integrator

eIndex SPV leverages the eGate™ Integrator by providing identification and cross-referencing capabilities for the data shared throughout the eGate system. It also uses the eGate Integrator to transform and route data between the eIndex SPV database and external systems by adding the eIndex SPV method OTD to the external system Collaborations or Business Processes.

eInsight Business Process Manager

eInsight™ Business Process Manager (BPM) is the component within Java CAPS that facilitates the automation of the flow of business activities. eInsight BPM functions include business process model design, monitoring, and execution as well as the ability to analyze how data messages flow from activity to activity and from page to page. You can include custom Java methods from the eIndex SPV Project in Business Processes for eInsight BPM, enabling access to the eIndex SPV database through eInsight BPM. eIndex SPV includes a sample Project that processes data in the same manner as the Java Collaboration, but uses an eInsight Business Process instead.

2.3 eIndex SPV Repository Components

eIndex SPV has two types of components: Repository and runtime. The Repository components are designed to work within the eGate Enterprise Designer to create and configure eIndex SPV and to define connectivity between external systems and eIndex SPV. This section describes the Repository components; the runtime components are described in [“Runtime Environment Components” on page 25](#).

The primary Repository components fall into three categories.

- [Editors](#) on page 18
- [Project Components](#) on page 19
- [Environment Components](#) on page 23

2.3.1 Editors

eIndex SPV provides the following editors to help you customize the files in the eIndex SPV Project.

- **XML Editor** - Allows you to review and customize the XML configuration files. Enterprise Designer provides schema validation services, while the XML Editor provides verification for XML syntax. The XML editor is automatically launched when you open an eIndex SPV configuration file.
- **Text Editor** - Allows you to review and customize the database scripts and match engine configuration files. This editor is very similar to the XML editor but without the validation and verification services. The text editor is automatically launched when you open an eIndex SPV database script or configuration file.
- **Java Source Editor** - Allows you to create and customize custom plug-in classes for eIndex SPV. This editor is a simple text editor, similar to the Java Source Editor in the Collaboration Editor. The Java source editor is automatically launched when you open a custom plug-in file.

2.3.2 Project Components

eIndex SPV is implemented within a Project in Enterprise Designer. The eIndex SPV Project includes a set of configuration files, a set of database files, and a set of custom plug-ins that you can customize. The Project includes additional components that are automatically updated when you generate the Project, including a method OTD, an outbound OTD, eInsight Business Process methods, database scripts, and the necessary **.jar** files. To complete the Project, you create a Connectivity Map and Deployment Profile.

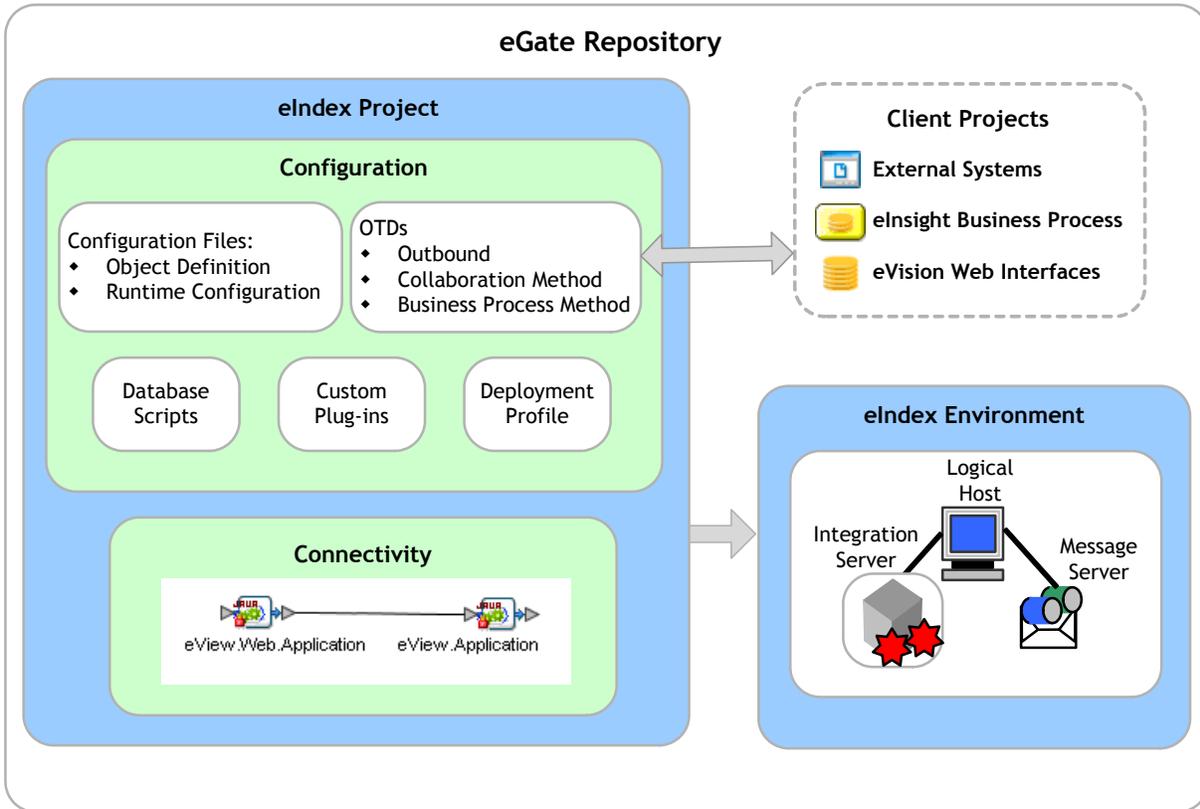
Additional eGate components can be added to the client Projects that share data with eIndex SPV, including Services, Collaborations, OTDs, Web Connectors, eWays, JMS Queues, JMS Topics, Business Processes, and so on. You can use the standard Enterprise Designer editors, such as the OTD or Collaboration editors, to create these components.

Following is a list of eIndex SPV Project components.

- Configuration Files
- Database Scripts
- Custom Plug-ins
- Match Engine Configuration Files
- Object Type Definitions
- Dynamic Java Methods
- Connectivity Components
- Deployment Profile

Figure 1 on page 20 illustrates the Project and Environment components of eIndex SPV.

Figure 1 eIndex SPV Project and Environment Components



Configuration Files

Several XML files together determine certain characteristics of eIndex SPV, such as how data is processed, queried, and matched. These files configure the runtime components of eIndex SPV, which are listed in **“Runtime Environment Components” on page 25**.

- **Object Definition** - Defines the data structure of eIndex SPV.
- **Enterprise Data Manager** - Configures the search functions and appearance of the EDM along with debug information and security information.
- **Candidate Select** - Configures the Query Builder component of eIndex SPV and defines the available queries.
- **Match Field** - Configures the Matching Service, and defines the fields to be standardized and used for matching. It also specifies the match and standardization engines to use.
- **Threshold** - Configures the eIndex Manager Service and defines certain system parameters, such as match thresholds, EUID attributes, and update modes. It also specifies which query from the Query Builder to use for matching queries.
- **Best Record** - Configures the Update Manager and defines the strategies used by the survivor calculator to determine the field values for the single best record (SBR).

- **Field Validation** - This file defines rules for validating field values. Rules are predefined for validating the local ID field and you can create custom validation rules to plug in to this file.
- **Security** - This file is a placeholder to be used in future versions.

Database Scripts

Several database scripts are included in the eIndex SPV Project to allow you to create the table structure, indexes, and custom start-up data. Several additional scripts are provided for testing purposes that drop the tables and indexes you create.

- **Systems** - Contains a sample SQL insert statement to add information about external systems to the database. You can modify this script to define your own systems.
- **Code List** - Contains the SQL statements to insert processing codes and drop-down list values into the database. You might need to customize this file to match the processing codes used by your external systems.
- **Create Person database** - Defines the structure of the eIndex SPV database based on the Object Definition file and defines indexes against standard database tables. You can customize this script, and then run it against an Oracle database to create a customized database.
- **Create User Indexes** - Defines indexes against the fields that are defined for the blocking query in the Candidate Select file. You can define additional indexes if needed.
- **Create User Code Data** - Provides a sample script for adding data to the `sbyn_user_code` table.
- **Drop Person database** - Used primarily in testing, when you need to drop existing database tables and create new ones. The delete script removes all tables related to eIndex SPV so you can recreate a fresh database for your Project.
- **Drop User Indexes** - Used primarily in testing, when you need to drop existing indexes, or for loading large batches of data, when indexes can slow down the process. This script removes all indexes defined in the **Create User Indexes** script.

You can also create custom scripts to store in the eIndex SPV Project and run against the database. The default database structure is described in the *eIndex Single Patient View Reference Guide*.

Custom Plug-ins

eIndex SPV provides a method by which you can create custom processing logic for the master index. To do this, you need to define and name a custom plug-in, which is a Java class that performs the required functions. Once you create a custom plug-in, you incorporate it into the master index by adding its name to the appropriate configuration file. You can create custom update procedures, which must be referenced in the update policies of the Best Record file, and field validations, which must be referenced in the Field Validation file. Custom plug-ins can also be used to create custom eIndex SPV components, such as a custom query builder or block picker.

Match Engine Configuration Files

Several configuration files for the Sun SeeBeyond Match Engine (SBME) are stored in the eIndex SPV Project. The configuration files under the Match Engine node define certain weighting characteristics and constants for the match engine. The configuration files under the Standardization Engine node define how to standardize names, business names, and address fields. You can customize any of these files as necessary. For more information, refer to *Implementing the Sun SeeBeyond Match Engine with eIndex SPV*.

Outbound Object Type Definition (OTD)

eIndex SPV includes an outbound OTD based on the object structure defined in the Object Definition file. This OTD is used for distributing information that has been added or updated in eIndex SPV to external systems. It includes the objects and fields defined in the Object Definition file plus additional SBR information (such as the create date and create user) and additional system object information (such as the local ID and system code). If you plan to use this OTD to make eIndex SPV data available to external systems, you must define a JMS topic in the Connectivity Map to which the master index can publish transactions.

Dynamic Java API

Due to the flexibility of the object structure, eIndex SPV includes several dynamic Java methods for use in Collaborations and Business Processes. One set is provided in a method OTD for use in Collaborations and one set is provided for Business Processes. The names, parameter types, and return types of these methods vary based on whether you modify the object structure in the Object Definition file. These methods are described in the *eIndex Single Patient View Reference Guide*.

Method OTD

The method OTD contains Java functions you can use to define data processing rules in Collaborations for external systems. These functions allow you to define how messages received from external systems are processed by the Service. You can define rules for inserting new records, retrieving record information, updating existing records, performing match processing, and so on.

Business Process Java Methods

In addition to the method OTD, which can be used in Collaborations, eIndex SPV includes a set of Java methods that can be incorporated into an eInsight Business Process and into eVision Web Services. These methods are a subset of those defined for the method OTD, providing the ability to view, retrieve, and match information in the database from a Business Process.

Connectivity Components

The default server Project Connectivity Map only consists of three components: the web application service, the application service, and an Oracle eWay for defining the database connection pool (the Oracle eWay is required if eIndex SPV runs on the Sun SeeBeyond Integration Server). You can also include a JMS Topic for broadcasting messages. In client Projects you can use any of the standard Project components to

define connectivity and data flow for eIndex SPV. Client Projects include those created for the external systems sharing data with the index using Collaborations or Business Processes.

For client Projects, you can use connectivity components from the eIndex SPV server Project and any standard eGate connectivity components, such as OTDs, Services, Collaborations, Business Processes, queues, topics, and eWays. Client Project components transform and route incoming data into the database according to the rules contained in the Collaborations or Business Processes. They can also route the processed data back to the appropriate local systems through eWays.

Deployment Profile

The Deployment Profile defines information about the production environment. It contains information about the assignment of Services and message destinations to application or integration servers and JMS IQ Managers within the eIndex SPV system. Each eIndex SPV Project must have at least one Deployment Profile, and can have several, depending on the Project requirements and the number of Environments used. You must deploy the Project before you can use the eIndex SPV runtime environment.

2.3.3 Environment Components

The eIndex SPV Environments define the deployment environment of the runtime components, including the Logical Host and application or integration server. For client Projects referencing the eIndex SPV Project, the Environment might also include a JMS IQ Manager, constants, Web Connectors, and External Systems. Each Environment represents a unit of software that implements eIndex SPV. You must define and configure at least one Environment for eIndex SPV before you can deploy the application. The application or integration server hosting eIndex SPV is configured within the Environment in Enterprise Designer.

For more information about Environments, see the *Sun SeeBeyond eGate Integrator User's Guide*.

2.4 About the Runtime Environment

Regardless of how you define the data structure and configure the runtime environment for eIndex SPV, the final product provides a cross-reference of centralized information that is kept current by the logic you define for unique identification, matching, and update transactions.

2.4.1 Functions of the Runtime Environment

In the runtime environment, eIndex SPV provides the following functions to help you monitor and maintain the data shared throughout the index system.

- **Transaction History** - The system provides a complete history of each patient by recording all changes to each patient's data. This history is maintained for both the local system records and the SBR.
- **Data Maintenance** - The web-based user interface supports all the necessary features for maintaining data records. It allows you to add new records; view, update, deactivate, or reactivate existing records; and compare records for similarities and differences. You can perform these functions against each local system record or SBR associated with a patient record.
- **Search** - The information contained in each SBR or system record can be obtained from the database using a variety of search criteria. You can perform searches against the database for a specific patient or a set of patients. For certain searches, the results are assigned a matching weight that indicates the probability of a match.
- **Potential Duplicate Detection and Handling** - One of the most important features of the eIndex SPV system is its ability to match records and identify possible duplicates. Using matching algorithm logic, eIndex SPV identifies potential duplicate records and provides the functionality to correct the duplication. Potential duplicate records are easily corrected by either merging the records in question or marking the records as “resolved”.
- **Merge and Unmerge** - You can compare potential duplicate records and then merge the records (at either the EUID or system-record level) if you find them to be actual duplicates of one another. At the EUID level, you can determine which record to retain as the active record. At the system level, you can determine which record to retain and which information from each record to preserve in the resulting record.
- **Reports** - You can generate reports that provide information about the current state of the data in the master index, helping you monitor stored data and determine how that data needs to be updated. Report information also helps verify that the matching logic and weight thresholds are defined correctly.

2.4.2 Runtime Environment Features

The components of the eIndex SPV runtime environment are designed to uniquely identify, match, and maintain information throughout a business enterprise. These components are highly configurable, allowing you to create a custom master patient index suited to your specific data processing needs. Primary features of runtime environment include:

- **Centralized Information** - eIndex SPV maintains a centralized database, enabling the integration of data records throughout the enterprise while allowing local systems to continue operating independently. eIndex SPV stores copies of local system records and of SBRs, which represent the most accurate and complete data for each patient. This database is the central location of information and identifiers, and is accessible throughout the enterprise.
- **Configurability** - Before deploying eIndex SPV, you define the components and processing capabilities of the system to suit your organization's processing requirements. You can configure the object structure, matching and standardization

rules, survivorship rules, queries, Enterprise Data Manager (EDM) appearance, and field validation rules.

- **Cross-referencing** - eIndex SPV is a global cross-referencing application that automates record matching across disparate source systems, simplifying the process of sharing data between systems. eIndex SPV uses the local identifiers assigned by your existing systems as a reference, allowing you to maintain your current systems and practices while maintaining the most current information.
- **Data Cleansing** - eIndex SPV uses configurable matching algorithm logic to uniquely identify object records and to identify duplicate and potential duplicate records. eIndex SPV provides the functionality to easily merge or resolve duplicates and can be configured to automatically match records that are found to be duplicates of one another. Data updates from external systems can occur in real time or as batch processes.
- **Data Updates** - eIndex SPV provides the ability to add, update, deactivate, and delete data in the database tables through messages received from external systems. Records received from external systems are checked for potential duplicates during processing. Merges can also be performed through external system messages.
- **Identification** - eIndex SPV employs configurable probabilistic matching technology, which uses a matching algorithm to formulate an effective statistical measure of how closely records match. Using a state-of-the-art algorithm in real-time mode and establishing a common method of locating records, the index consistently and precisely identifies patients within an enterprise.
- **Integration** - Relying on the eGate Integrator, eIndex SPV provides the power and flexibility to identify, route, and transform data to and from any system or application throughout your business enterprise. It can accept incoming transactions and distribute updates to external systems, providing seamless integration with the systems in your enterprise.
- **Matching Algorithm** - eIndex SPV is designed to use the SBME or a custom matching algorithm to provide a matching probability weight between records. The SBME provides the flexibility to create user-defined matching thresholds, which control how potential duplicates and automatic merges are determined.
- **Shared Information** - Each time a record is updated, added, merged, or unmerged from the EDM, eIndex SPV generates a message that can be transmitted to external systems. It also receives, processes, and broadcasts messages containing information about the patients in the database.
- **Unique Identifier** - Records from various systems are cross-referenced using an enterprise-wide unique identifier, known as an EUID, that eIndex SPV assigns to each patient record. eIndex SPV uses the EUID to cross-reference the local IDs assigned to each patient by the various computer systems throughout the enterprise.

2.5 Runtime Environment Components

The eIndex SPV runtime environment is made up of several components that work together to form a complete indexing system. The runtime environment includes the following primary components.

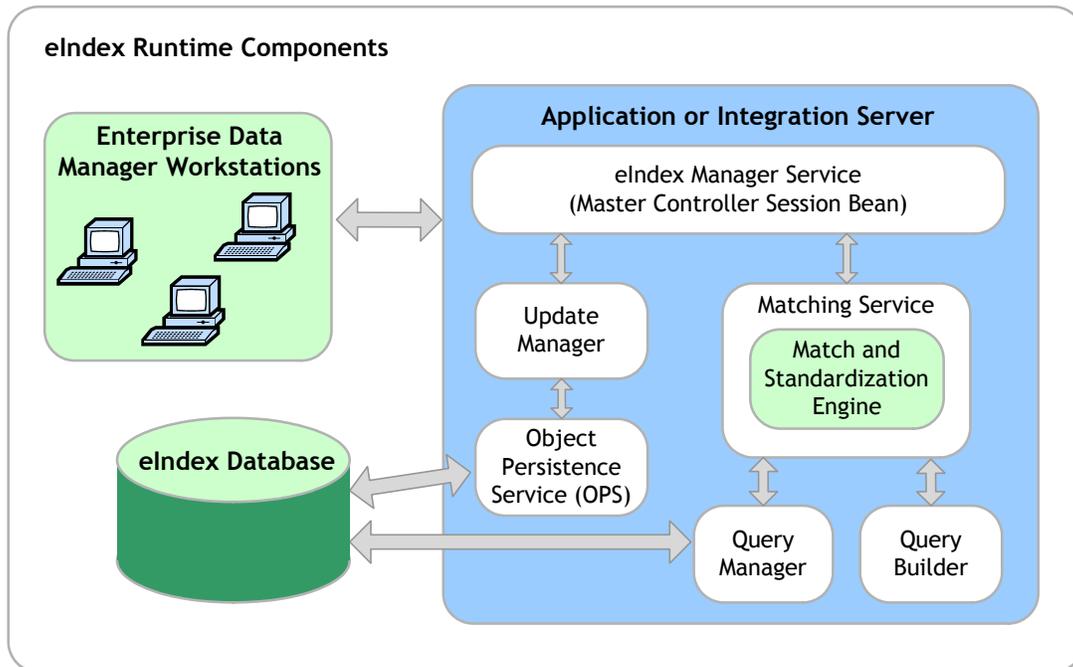
- **Matching Service** on page 26
- **eIndex Manager Service** on page 27
- **Query Builder** on page 27
- **Query Manager** on page 27
- **Update Manager** on page 27
- **Object Persistence Service (OPS)** on page 27
- **Database** on page 27
- **Enterprise Data Manager** on page 28

In addition, eIndex SPV uses the connectivity components defined in the eIndex SPV server and client Projects to route data between external systems and the eIndex SPV database.

The eGate Repository stores information about the configuration and structure of the runtime environment. Because eIndex SPV is deployed within eGate, it can be implemented in a distributed environment. The eIndex SPV system requires the Sun SeeBeyond Integration Server (IS) or the Sun Java System Application Server to enable web service connectivity.

The components of the eIndex SPV runtime environment are illustrated in Figure 2.

Figure 2 eIndex SPV Runtime Environment Architecture



2.5.1 Matching Service

The Matching Service stores the logic for standardization (which includes data parsing and normalization), phonetic encoding, and matching. It includes the specified standardization and match engines, along with the configuration you defined for each. The Matching Service also contains the data standardization tables and configuration files for the match engine. The configuration of the Matching Service is defined in the Match Field file.

2.5.2 eIndex Manager Service

The eIndex Manager Service provides a session bean to all components of the runtime environment, such as the Enterprise Data Manager, Query Builder, and Update Manager. The service also manages connectivity to the database. The configuration of the eIndex Manager Service specifies the query to use for matching and defines system parameters that control EUID generation, matching thresholds, and update modes. The configuration of the eIndex SPV Manager Service is defined in the Threshold file.

2.5.3 Query Builder

The Query Builder defines all queries available to eIndex SPV, including the queries performed automatically by eIndex SPV when searching for possible matches to an incoming record. It also includes the queries performed manually through the Enterprise Data Manager (EDM). The EDM queries can be either alphanumeric or

phonetic and have the option of using wildcard characters. The configuration of the Query Builder is defined in the Candidate Select file.

2.5.4 Query Manager

The Query Manager is a service that performs queries against the database and returns a list of objects that match or closely match the query criteria. The Query Manager uses classes specified in the Match Field file to determine how to perform a query for match processing. All queries performed in the eIndex SPV system are executed through the Query Manager.

2.5.5 Update Manager

The Update Manager controls how updates are made to a patient's SBR by defining a survivor strategy for each field. The survivor calculator in the Update Manager uses these strategies to determine the relative reliability of the data from external systems and to determine which value for each field to populate into the SBR. The Update Manager also manages certain update policies, allowing you to define additional processing to be performed against incoming data. The configuration of the Update Manager is defined in the Best Record file.

2.5.6 Object Persistence Service (OPS)

OPS is a database service that translates high-level and descriptive object requests into actual JDBC calls. The service provides mapping from the Java object to the database and from the database to the Java object.

2.5.7 Database

eIndex SPV uses an Oracle database to store the information you specify for patient records. The database stores local system records, the SBR for each patient record, and certain administrative information, such as drop-down menu lists, processing codes, and information about the systems from which data originates. The script used to create the database structure is based on the object structure defined in the Object Definition file.

2.5.8 Enterprise Data Manager

The Enterprise Data Manager (EDM) is a web-based interface that allows you to monitor and maintain the data in the eIndex SPV database. The configurable attributes of the EDM are defined in the Enterprise Data Manager file. The EDM provides the ability to manually search for records; update, add, deactivate, and reactivate records; merge and unmerge records; view potential duplicates; view comparisons of patient records; and run reports for specific types of records and transactions.

2.6 Enterprise Records

An *enterprise record* is identified by an EUID assigned by eIndex SPV and includes all components of a record that represents one patient. eIndex SPV stores two different types of records in each enterprise record: system records and a single best record (SBR). A system record contains a patient's information as it appears in an incoming message from an external system. An enterprise record's SBR stores data from a combination of external systems and it represents the most reliable and current information contained in all system records for a patient. An enterprise record consists of one or more system records and one SBR.

2.6.1 System Records

The structure of a system record is different from the SBR in that each system record contains a system and local ID pair. The remaining information contained in the system records of an enterprise record is used to determine the best data for the corresponding SBR. If an enterprise record only contains one system record, the SBR is identical to that system record. However, if it contains multiple system records, the SBR might be identical to one system record but will more likely include a combination of information from all system records.

2.6.2 The Single Best Record

The SBR for a patient is created from the most reliable information contained in each system record for a particular enterprise record. The information used from each external system to populate the SBR is determined by the survivor calculator, which is configured in the Best Record file. This data is determined to be the most reliable information from all system records in the enterprise record. The survivor calculator can consider factors such as the relative reliability of an external system, how recent the data is, and whether the SBR contains any "locked" field values. You define the rules that select a field to be persisted in the SBR.

2.6.3 Objects in an Enterprise Record

In eIndex SPV, each system record and SBR in an enterprise record typically contain a set of objects that store different types of information about a patient. A record contains one parent object and typically contains several child objects, but it can have no child objects. A record can have only one instance of the parent object, but can have multiple instances of each type of child object. For example, in the default configuration, the parent object contains demographic data. A record can only contain one patient name and social security number (stored in the parent object), but the record could have multiple addresses, telephone numbers, and aliases, which are defined in different child objects (*address*, *phone*, and *alias* objects respectively). A record can have multiple instances of each child object, such as a home and a billing address.

2.7 From the Project to the Runtime Environment

The process of creating the runtime environment begins with a thorough analysis of the data you plan to store in the database and to share among the systems connected to eIndex SPV. Once your analysis is complete, you can customize the object structure and the configuration files for your processing requirements.

2.7.1 Process Overview

The following steps outline the procedures you need to follow to build the runtime environment using the eIndex SPV Project components.

- 1 Perform a thorough analysis of the data you plan to store in the database.
- 2 Customize the object structure, operating environment, and certain runtime characteristics.

Note: *If you customize Object Definition, you might also need to make corresponding changes to the other configuration files. For example, if you add a new field to Object Definition that you want to include in queries and matching, you need to make the corresponding changes to the Candidate Select, Match Field, and Best Record files.*

- 3 Define and build custom plug-ins, and specify the plug-ins in the appropriate configuration file. This step is optional.
- 4 Generate the master index.
- 5 Create the database.
 - ♦ Customize the scripts by defining system information, processing codes, and drop-down menu values.
 - ♦ Create the database and any necessary indexes.
- 6 Create Connectivity Maps.
 - ♦ Create and define the components in the Connectivity Maps, such as Services, Collaborations, and External Applications.
 - ♦ Configure the Connectivity Maps.
- 7 Define the Environment and configure the server and security.
- 8 Create the Deployment Profile and activate the Project.

2.7.2 From XML to the Database

The database is created using a standard Oracle database and a database script generated from the Object Definition file. Running the database script against a standard Oracle database creates the tables necessary for eIndex SPV. You must define the sizing and distribution of your database before running the database script and create any necessary indexes against the primary object tables after running the file.

2.7.3 From XML to the Enterprise Data Manager

The Enterprise Data Manager is created based on information in the Enterprise Data Manager configuration file. This file defines the fields and appearance of the EDM, and also specifies the searches used by the EDM. The available search types are defined in the Candidate Select file. You can customize many features of the EDM, including the following.

- The fields that appear on the windows
- Field attributes, such as a display name, display order, maximum field length, the field type and format, and so on
- The types of searches that can be performed and the fields available for each type
- The appearance of search results lists
- The location of the fields on all windows

2.7.4 From XML to the Connectivity Components

The eIndex SPV Project contains several default connectivity components, including a method OTD, Business Process methods, and an outbound OTD. These components contain Java methods whose names are based on the default Object Definition. The method OTD contains certain Java methods for use in Collaborations; the Business Process methods are used for eInsight BPM integration; and the outbound OTD is used when publishing messages processed by eIndex SPV for broadcasting to external systems. Generating a Project also creates application files that you can drag into the Connectivity Map.

2.7.5 From XML to the Runtime Environment

The information you specify in the eIndex SPV configuration files is stored in the eGate Repository and is read at runtime when the domain is started. The only exception is the Object Definition file, which is stored only as a record of the object structure. You can modify many of the parameters of the configuration files after moving to production. For the changes to take effect, you must regenerate the application and redeploy the Project to apply the changes to the Logical Host. You also need to restart the EDM and any eWays connected to the application for the changes to take effect. Use caution when modifying these files; changing certain elements after moving to production might result in loss of data integrity or unexpected weighting and matching results.

2.8 Working with Project Components

eIndex SPV supports standard Sun Java Composite Application Platform Suite functions for version control and for copy, cut, and paste functions. In addition, Enterprise Designer provides validation capability for the XML files (this is described in the *eIndex Single Patient View Configuration Guide*).

Version Control

eIndex SPV supports the version control functionality provided by Enterprise Designer. You can check files in and out, retrieve older versions to a workspace, view a version history, and so on. In addition, eIndex SPV supports recursive check-ins and check-outs. When you select **Recurse Project**, you can check in or out all components below the selected node or a subset of those components.

For more information about version control and checking files in and out, see the *Sun SeeBeyond eGate Integrator User's Guide*.

Saving a Configuration File to the Repository

After you modify a configuration file in an eIndex SPV Project, you must save the file to the Repository using the XML Editor's Save command.

Note: For more information about the XML Editor, see the *eIndex Single Patient View Configuration Guide*.

To save a file to the Repository

- 1 With the file open in the XML editor, right-click in the XML editor to display the XML editor context menu.
- 2 On the context menu, click **Save**.
- 3 Validate the file (described in the following procedure) and check it back in to the Repository.

Note: If you did not check the file out before making changes and attempting to save it, a warning dialog appears. Click **Yes** on this dialog to automatically check out the file, save the changes, and check it back in.

Validating XML Files

eIndex SPV includes one XML schema definition (XSD) file for each configuration file. Before saving changes to a file, be sure to validate it against the XSD file to make sure no dependencies have been broken during modification.

To validate XML syntax

- 1 After you save any changes to a configuration file to the Repository, keep the file open and right-click in the text of the file.
- 2 On the context menu that appears, click **Check XML**.
- 3 A message appears indicating the status of the validation and, if there were errors, includes a list of errors.
- 4 Fix any errors found in the file and revalidate.

To validate against the schema

- 1 In the Project Explorer, right-click the file to validate.

- 2 On the context menu that appears, click **Validate**.
- 3 A message appears indicating the status of the validation and, if there were errors, includes a list of errors.
- 4 Fix any errors found in the file and revalidate.

Copying, Cutting, and Pasting Files

You can use standard cut, copy, and paste commands to copy or move files between Projects. This functionality is described in the *Sun SeeBeyond eGate Integrator User's Guide*. eIndex SPV follows the standard functionality, with the exception that you can only copy or move a component from one Project into the same node of another Project. For example, you can only paste a copied configuration file into the Configuration node of another Project. In addition, you cannot cut or delete components that are essential to a Project, such as the configuration files, match and standardization files, and so on.

Installation

eIndex Single Patient View is uploaded to the eGate Repository and is then installed in Enterprise Designer. This chapter provides instructions on installing eIndex SPV once the eGate environment is in place.

What's in This Chapter

- [Installation Overview](#) on page 33
- [About the Installation](#) on page 34
- [Supported Operating Systems](#) on page 34
- [System Requirements](#) on page 35
- [Software Requirements](#) on page 35
- [Installing eIndex SPV](#) on page 36

3.1 Installation Overview

In order to work with eIndex SPV, you only need to perform the eIndex SPV installation described in this chapter. To work with the eIndex SPV master index, a second component, an Oracle database, must be installed.

3.1.1 eIndex SPV Installation

eIndex SPV is installed by uploading the eIndex SPV files to the eGate Repository using Enterprise Manager, and then installing the software in Enterprise Designer. eIndex SPV must be uploaded via an active eGate Repository, and the software must be installed using a computer with an existing Enterprise Designer. You must have access to a web browser for the initial upload. When you install the eIndex SPV application, you can also install the reports, documentation, and Javadocs.

Before installing eIndex SPV, make sure you have installed your eGate environment, including the Repository, monitors, Logical Host, Enterprise Designer, and integration or application server. eIndex SPV can run on the Sun SeeBeyond Integration Server 5.1.0 or Sun Java System Application Server 8.1.

3.1.2 Database Installation

All of the eIndex SPV components are stored in the eGate Repository; the only external component is the master index database. The database does not need to be installed in order to use the eIndex SPV tools, but it must be installed as a part of the master index implementation. For optimal performance, the database should be installed on its own server. The master index database can be installed on Oracle 9i or 10g and can run on any operating system platform supported by Oracle 9i or 10g.

If you are running the Sun Java System Application Server, the database connection pool can be configured either through the application server or through the Oracle eWay. If you are running the Sun SeeBeyond Integration Server, the connection pool must be configured through an Oracle eWay. The Oracle eWay supports both thin client connectivity and OCI client connectivity.

To install the database, create a standard Oracle instance, using the sizing and distribution requirements obtained from the data analysis. Several SQL scripts are included in the eIndex SPV Project. Running these scripts against the database creates the tables and inserts startup data. For complete instructions on installing the database, see [Chapter 7 “Creating the Database”](#).

3.2 About the Installation

The eIndex SPV installation is a multi-stage process that includes the following:

- 1 Uploading eIndex SPV into the Repository.
- 2 Downloading reports and the Enterprise Manager Monitor plug-in.
- 3 Uploading the documentation and sample files.
- 4 Installing eIndex SPV in Enterprise Designer.

3.3 Supported Operating Systems

This section lists the supported operating system requirements for each platform. The eIndex SPV **Readme.txt** file (located in the **eIndexDocs.sar** file) contains the most up-to-date information for the supported platforms. eIndex Single Patient View is available on the following operating systems.

- Sun Solaris 8 (SPARC), 9 (SPARC), and 10 (SPARC and AMD Opteron)
- HP Tru64 V5.1A with patch 5 and V5.1B with required patches
- HP-UX 11.0 (PA-RISC), 11i (PA-RISC), and 11i v2.0 (11.23) (Itanium) with required patches and parameter changes
- IBM AIX 5L, versions 5.2 and 5.3 with required maintenance level patches
- Red Hat Enterprise Linux AS v.2.1 (Intel x86) and v.3.0 (AMD Opteron and Intel x86)

- SUSE Linux Enterprise Server 8 (Intel x86) and 9 (Intel x86)
- Windows 2000 SP3 and SP4, Windows XP SP1a and SP2, and Windows Server 2003 SP1

3.4 System Requirements

eIndex SPV is installed within the Java Composite Application Platform Suite environment. For system requirement information for the suite environment, see the *Java Composite Application Platform Suite Installation Guide*, which also contains information about resource considerations. The Java CAPS **Readme.txt** file contains the most up-to-date operating system requirements for the supported platforms. Both files are located in the Root directory of the Repository installation CD-ROM.

In addition to the operating system requirements listed above, you must have the Java 2 Platform, Standard Edition (J2SE™ Platform) installed on the machine from which you will run the eIndex SPV reports. The eIndex SPV **Readme.txt** file (located in the **eIndexDocs.sar** file) contains the most up-to-date information for the system requirements.

3.4.1 Database Requirements

Requirements for the master index database are included in **Chapter 7 “Creating the Database”**. The client workstations accessing the EDM require Internet Explorer 6.0 with SP1. If you will connect to the database using the Oracle eWay, you must install the Oracle eWay. To use the OCI driver with the Oracle eWay, you need to install the Oracle client on the Logical Host. Because the OCI driver supports both the most current Oracle database and any previous versions, Sun recommends that you use the most current version of the OCI driver. This assures compatibility between the various versions of any Oracle databases you have.

3.5 Software Requirements

eIndex SPV can be installed after you have done the following:

- Installed the Repository.
- Installed Enterprise Manager, including these files:
 - A **eGate.sar**
 - B **eView.sar**
 - C **FileeWay.sar**
 - D **OracleeWay.sar**
 - E **eInsight.sar** (only if you want to install the eInsight sample Project for eIndex SPV or will use eIndex SPV functions in an eInsight Business Process)

3.5.1 Before Installing eIndex SPV

Before you install eIndex SPV, you must do the following:

- 1 Select the Window(s) computers that will host eIndex SPV. This must be a computer running Enterprise Designer, which only runs on Windows systems.
- 2 Determine the add-on applications, if any, you need.
- 3 Make sure eView Studio is installed; otherwise the eIndex SPV projects will not appear.
- 4 Make sure the File eWay is installed. eIndex SPV relies on this eWay in order to create the start-up Project.
- 5 Determine whether you want to install the eIndex SPV client Project for eInsight BPM. If so, you must have eInsight BPM installed before installing eIndex SPV.
- 6 Determine whether to connect to the database using the application server or the Oracle eWay and whether to connect through a thin or thick client.
- 7 Make sure you have the appropriate administrator permissions to install eIndex SPV in Enterprise Designer and to update Enterprise Designer through the Update Center.
- 8 Before you begin the installation, exit all Windows applications.
- 9 If you are reinstalling eIndex SPV, determine whether you want to reinstall the eIndex SPV Projects or retain the Projects you already have installed. To install the updated Projects, you must delete the existing eIndex SPV Projects. If you have customized your existing Projects and want to use them in the new environment, back them up by exporting them before installing eIndex SPV and then reimport them. Make sure to regenerate the application and then rebuild and redeploy all related Projects.

3.6 Installing eIndex SPV

To install eIndex SPV, you must complete the following tasks:

- [Uploading eIndex SPV Components to the Repository](#) on page 36
- [Downloading eIndex SPV Components from the Repository](#) on page 40
- [Installing eIndex SPV in Enterprise Designer](#) on page 43

3.6.1 Uploading eIndex SPV Components to the Repository

The first step to install eIndex SPV is uploading the files into the eGate Repository. These files are in the form of an archive file that contains all the actual components of the eIndex SPV package. Make sure you have installed all the necessary Java Composite Application Platform Suite components before beginning.

The following steps are performed using the Java Composite Application Platform Suite Installer, which serves as an update and management center. Additionally, system administrators use the installer to upload components to the Repository server.

To upload eIndex SPV components

- 1 Make sure the eGate Repository is started. (Run **startserver.bat** in the Repository home directory if it is not started.)
- 2 Start your web browser.
- 3 In the **Address** line, type **http://<hostname>:<port_number>** where:

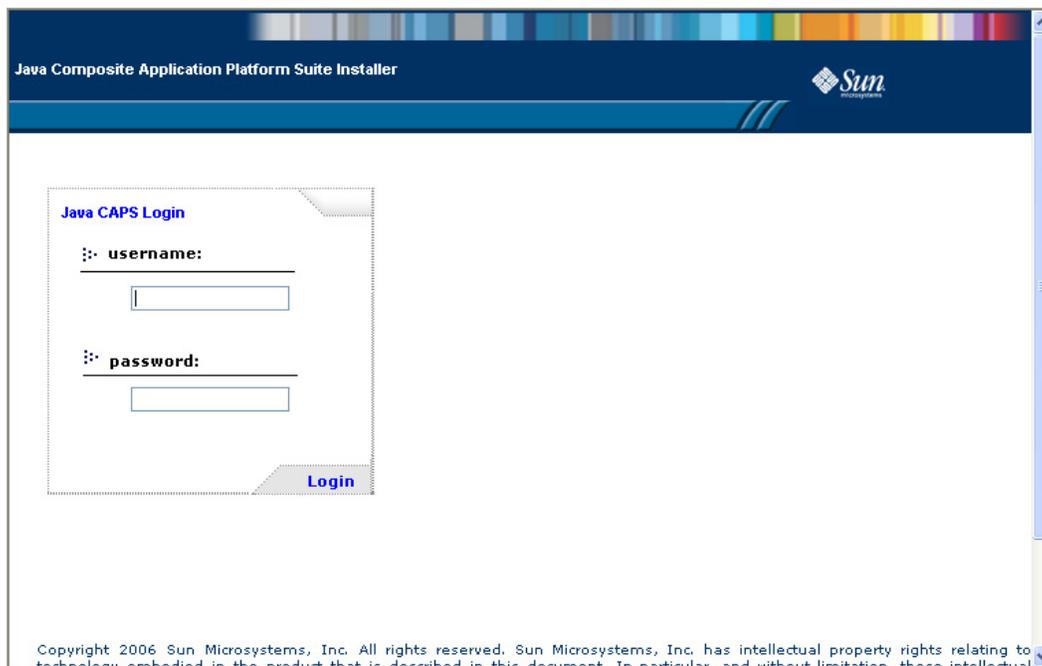
<hostname> is the TCP/IP host name of the server where you installed the Repository—not the name of the Repository itself.

<port_number> is the port number you gave during the installation of the Repository.

When ready, press **Enter**.

The **Login** window of the Suite Installer appears (see Figure 3).

Figure 3 Java CAPS Login Page



- 4 Enter your **username** and **password**.
- 5 When ready, click **Login**.

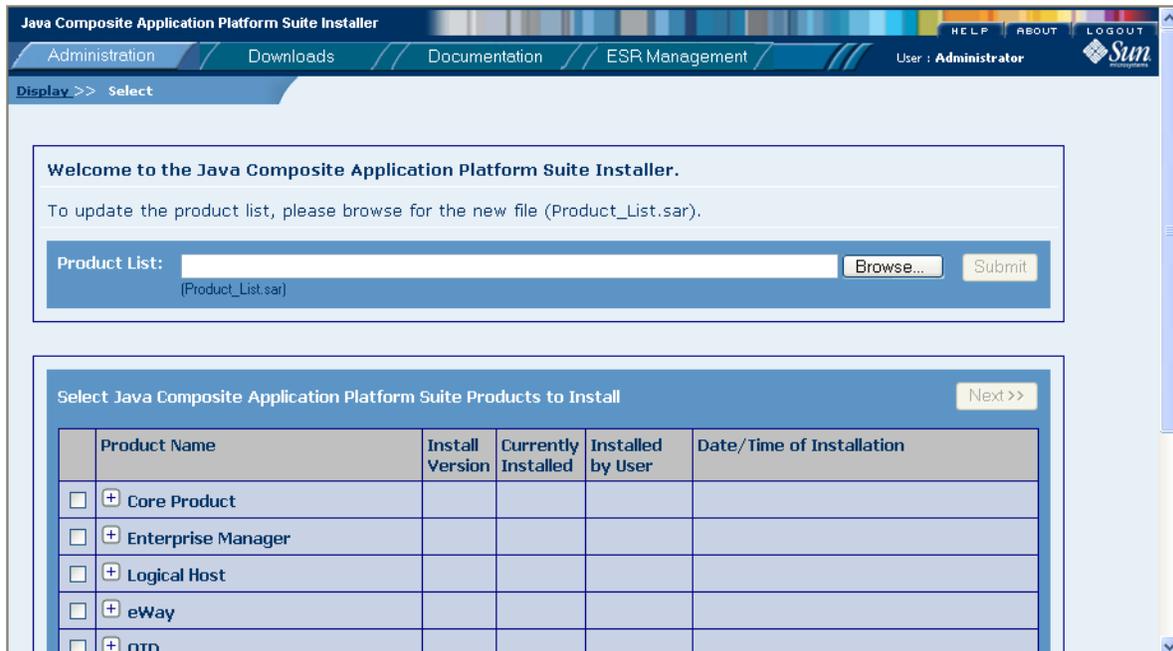
The **Administration** page of Enterprise Manager appears (see Figure 4).

Figure 4 Administration Page



- On the **Administration** page, click the link labeled **Click to install more products**. A list of products you can install appears (see Figure 5).

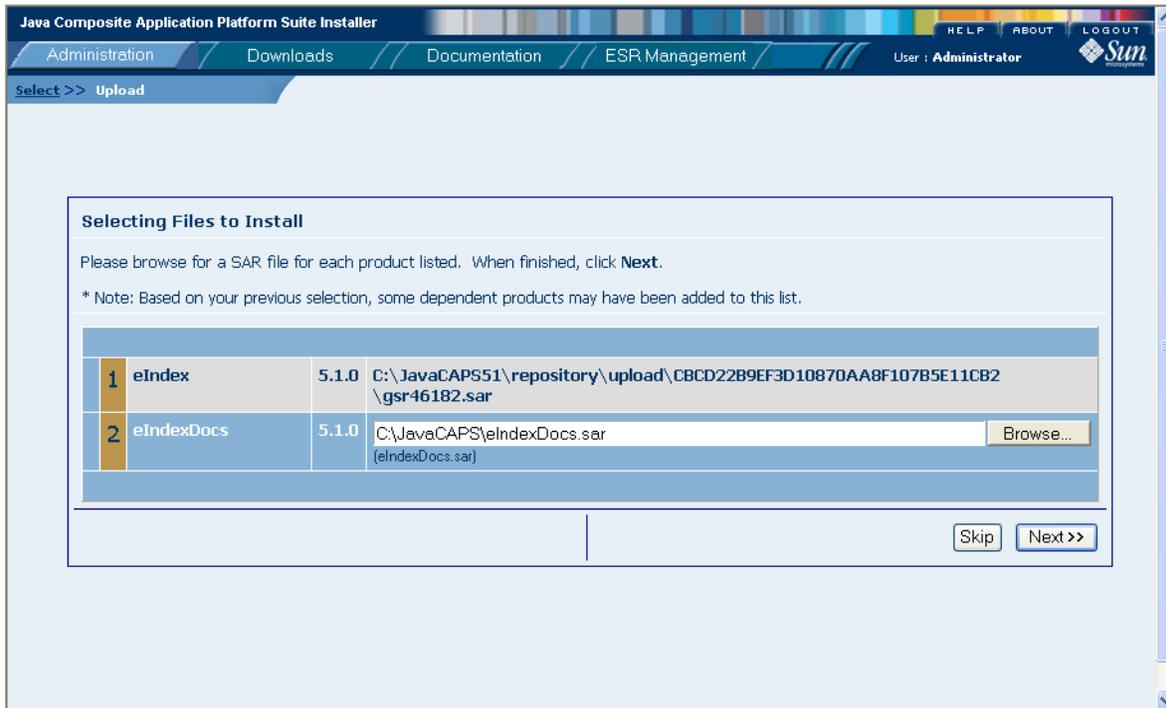
Figure 5 List of Products to Install



- In the product list, expand Core Products, and then select the check box next to eIndex.

- 8 In the product list, expand Documentation, and then select the check box next to eIndexDocs.
- 9 Click **Next** (in the bottom right section of the page).
The **Upload** page appears.

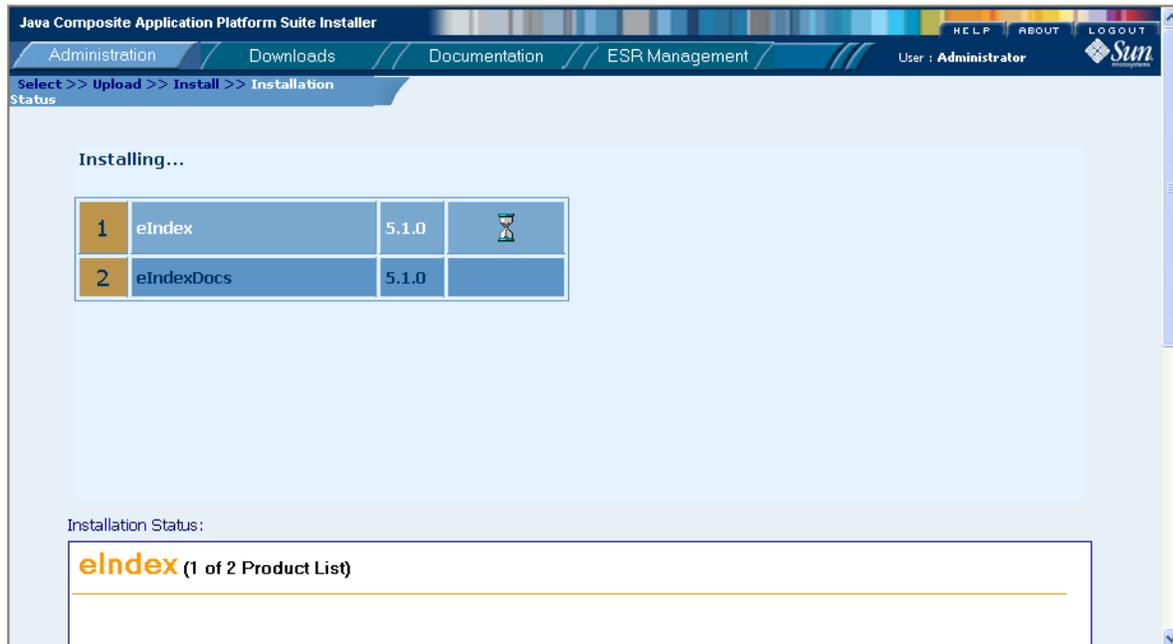
Figure 6 Selecting Files to Install



- 10 On the **Upload** page, browse for the requested file (either **eIndex SPVDocs.sar** or **eIndex SPV.sar**), and then click **Next**.
- 11 Repeat step 11 for the next requested file, and then click **Next**.

After the last file is uploaded, the **Installation** page appears and the installer begins to install the selected products (see Figure 7).

Figure 7 Installation Page



12 Do one of the following:

To download additional components of eIndex SPV using the installer, continue to the following set of procedures, **“Downloading eIndex SPV Components from the Repository”** on page 40.

To complete the eIndex SPV installation in Enterprise Designer, skip to **“Installing eIndex SPV in Enterprise Designer”** on page 43.

3.6.2 Downloading eIndex SPV Components from the Repository

Once eIndex SPV is installed to the Repository, you can download and access eIndex SPV components, such as the Enterprise Manager Monitor plug-in for and eIndex SPV, reports, and documentation. The following steps are performed using the Suite Installer.

- **Installing the eIndex SPV Monitor Client** on page 40
- **Downloading eIndex SPV Reports** on page 42
- **Accessing the eIndex SPV Documentation** on page 42

Before performing any of these steps, make sure eIndex SPV is installed as described in **“Uploading eIndex SPV Components to the Repository”** on page 36.

Installing the eIndex SPV Monitor Client

The Enterprise Manager Monitor is a web-based interface you use to manage applications. The Monitor requires the plug-in specific to eIndex SPV to monitor and manage eIndex SPV applications (eIndex SPV uses the same plug-in as eView Studio).

The plug-in enables the Monitor to target specific alert codes for eIndex SPV and is available from the list of components to download on the **Downloads** tab of the installer.

To download the eIndex SPV Monitor Client

- 1 On the **Downloads** tab of the Suite Installer, select **eView Monitor Client**.

Figure 8 Downloads Tab



- 2 Save the file to the Repository home directory.
- 3 Launch and log on to the Enterprise Manager Monitor.
- 4 From the toolbar of the Enterprise Manager Monitor, select **Configuration**.
The User Preferences page appears.
- 5 Click the **Web Applications Manager** tab in the upper right section of the page.
The Deploy New Management Application page appears.
- 6 Click the **Auto-Install from Repository** tab.
- 7 Enter the Repository logon information, and then click **Connect**.
- 8 In the **Management Applications available for installation from the Repository** list, select the check box next to eView Monitor Client.
- 9 Click **Install**.

In the Results list, verify that the eView Monitor Client was installed properly.

For more information about configuring and using the Enterprise Manager Monitor, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Downloading eIndex SPV Reports

Once eIndex SPV is uploaded to the Repository, you can download command line reports to run against the database. This is optional, and all reports available from the command line are also available through eIndex SPV's web-based GUI, the Enterprise Data Manager (EDM).

To download eIndex SPV reports

- 1 On the Suite Installer, click the **Downloads** tab.
- 2 Click **eView Reports**.
- 3 A dialog appears prompting you to open the file to disk or save it to your computer. Extract the files to the directory where you want to store the reports (the machine on which you store the reports must have network access to the application or integration server).

Accessing the eIndex SPV Documentation

When you uploaded the **eIndex SPVDocs.sar** file in previous steps, links to the documentation components of eIndex SPV were created on the **Documentation** tab of the installer. Perform any of the following steps to access the documentation files.

- ♦ [To access the documentation files](#) on page 42
- ♦ [To download the Javadoc files](#) on page 42

To access the documentation files

- 1 On the Suite Installer, click the **Documentation** tab.
- 2 In the frame on the left side of the page, select the Composite Applications tab.
- 3 Click **eIndex Single Patient View**.
- 4 To view a document in Acrobat Reader, select any document title from the frame on the right side of the page.
- 5 To view documents in HTML format:
 - ♦ Select the icon next to **HTML Help** in the frame on the right side of the page. The eIndex SPV Help window appears.
 - ♦ Select a book or topic to view from the contents in the left side of the page.

Note: You can view a summary of a document by clicking the information symbol to the left of the document name.

To download the Javadoc files

- 1 On the Suite Installer, click the **Documentation** tab.
- 2 Click **eIndex Single Patient View**.
- 3 In the frame on the right side of the page, scroll to, and then click, **Download Javadoc**.

- 4 A dialog appears prompting you to open the file or save it to your computer. Extract the files to the directory where you want to store the files.
- 5 To access the documents, navigate to the directory where you extracted to files and then to `\eIndex SPV_Javadoc\html`.
- 6 Double-click `index.html`. This page provides links to all Javadoc pages.

3.6.3 Installing eIndex SPV in Enterprise Designer

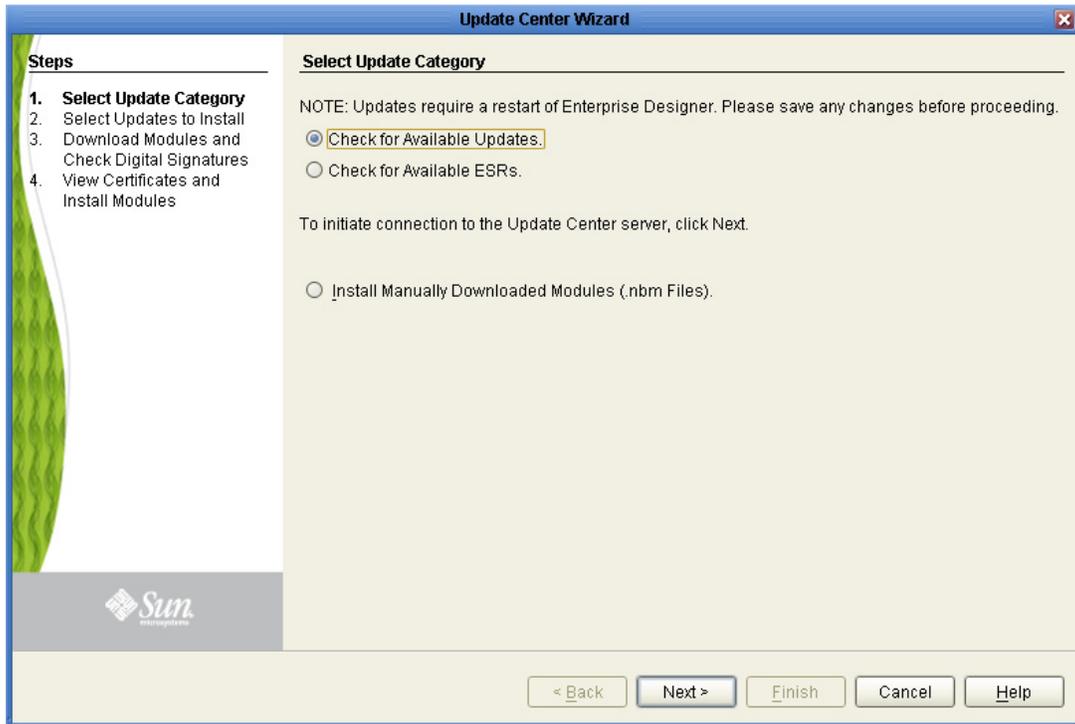
The final step in installing eIndex SPV is updating Enterprise Designer with the eIndex SPV files. Enterprise Designer must be installed on the machine on which you are installing eIndex SPV. This step is performed using the Update Center, which is a tool in Enterprise Designer that allows you to install add-on modules into Enterprise Designer.

***Note:** The eIndex SPV Projects might be visible on the Enterprise Designer before you perform these steps. If they are not, you can click **Refresh All from Repository** to view the Projects. However, you cannot work with the Project files until you perform the following steps.*

To install eIndex SPV in Enterprise Designer

- 1 Navigate to `<c:\JavaCAPS51>\edesigner\bin` and double-click `runed.bat`. The Enterprise Designer GUI opens.
- 2 Select the **Tools** menu and then click **Update Center**.
The **Update Center Wizard** appears (see Figure 9).

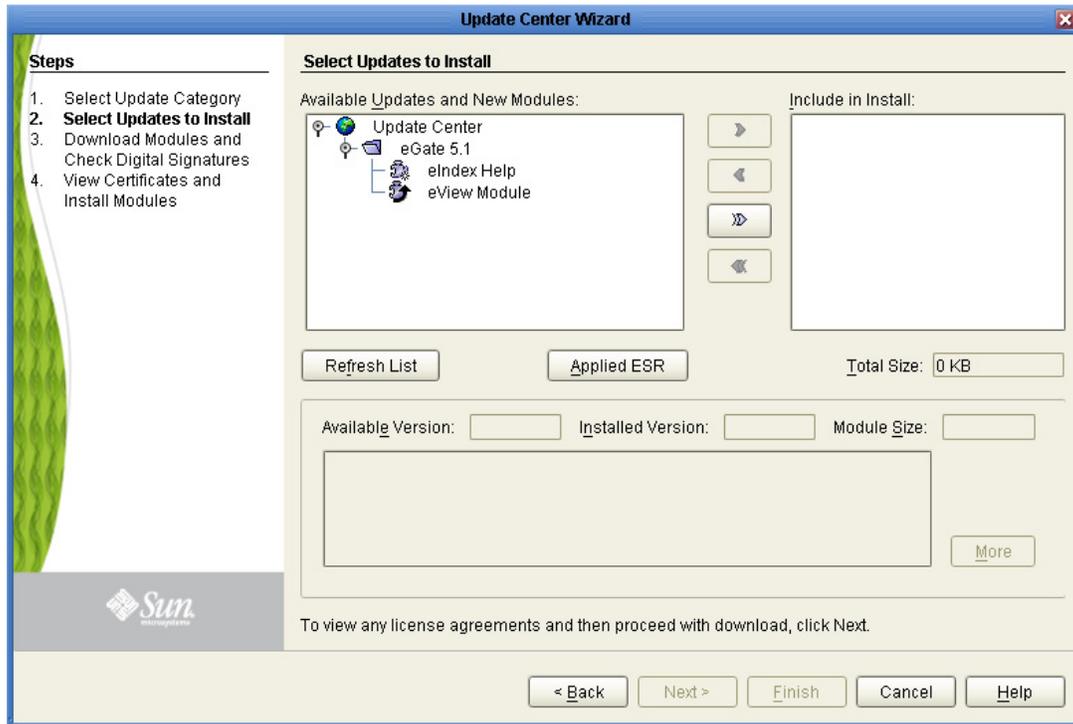
Figure 9 Update Center wizard - Select Update Category



- 3 On the Select Update Category window, select **Check for Available Updates**, and then click **Next**.

The **Select Modules to Install** window appears (see Figure 10).

Figure 10 Update Center Wizard - Select Modules to Install



4 Do one of the following:

- ◆ Move all items in the **Available Updates and New Modules** box at once by clicking the **Add All** button (the double-arrow button between the two panes).
- ◆ In the **Available Updates and New Modules** box, select **eIndex Help**, and then click the **Add** button (single-arrow button at the top).

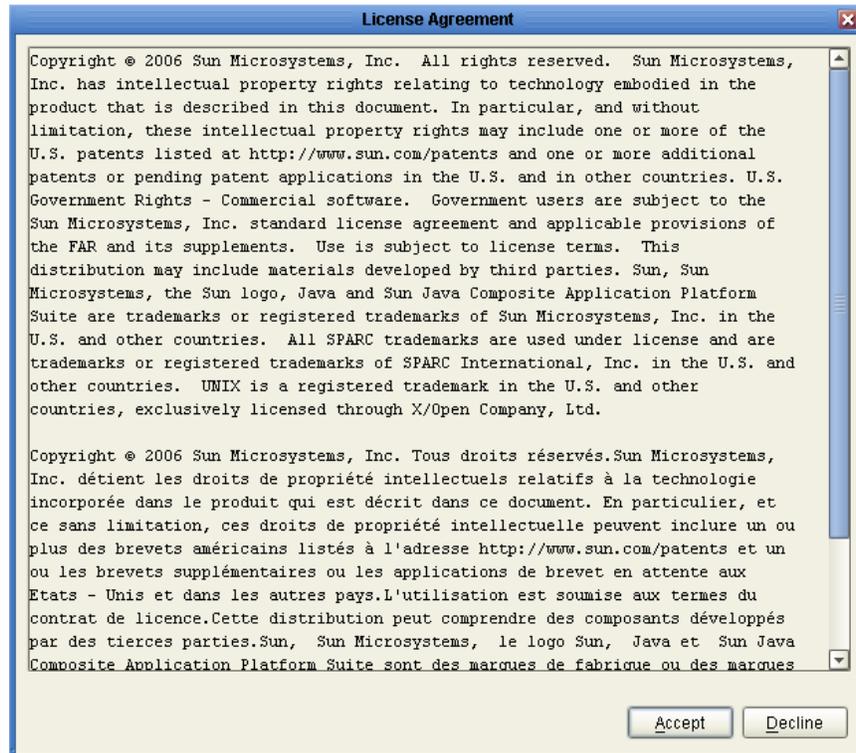
Note: To select each eIndex SPV module, hold down the **Control** key and click each module.

The eIndex SPV files move to the **Include in Install** list.

5 Click **Next**.

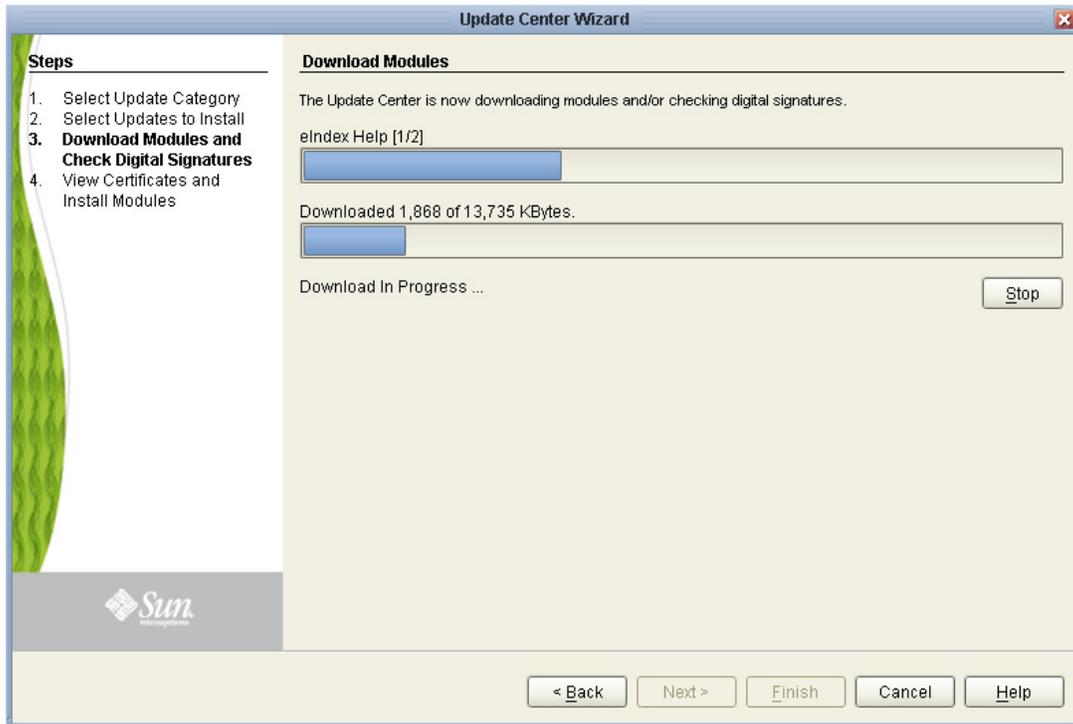
The License Agreement window appears (see Figure 11).

Figure 11 Update Center Wizard - License Agreement



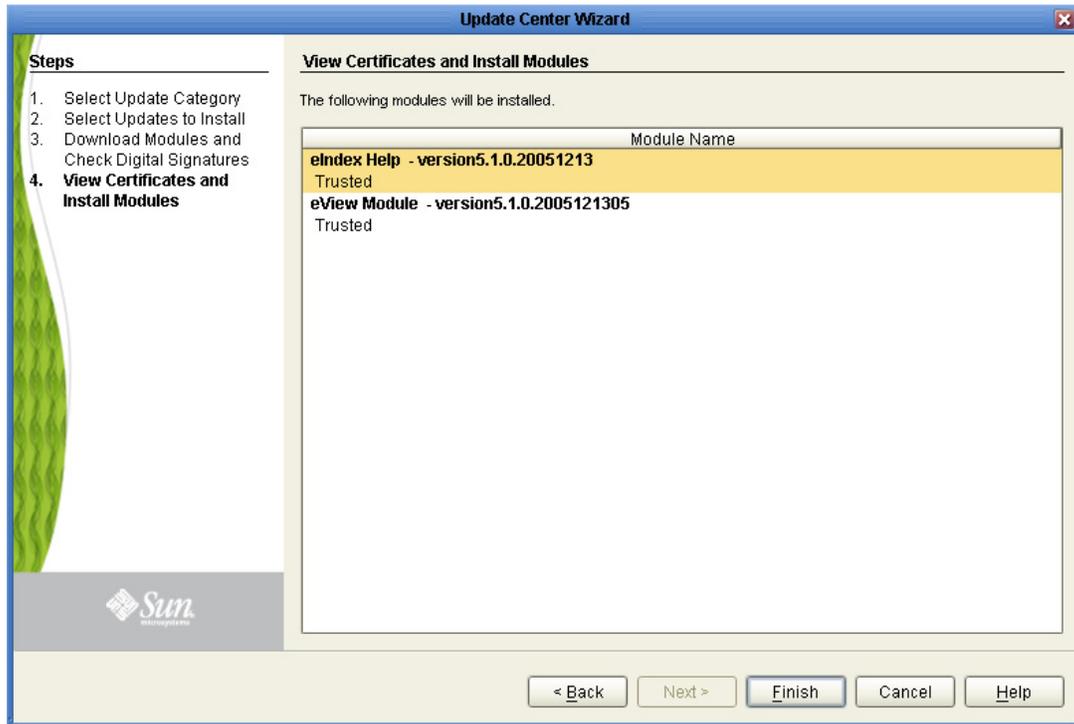
- 6 On the **License Agreement** window, click **Accept**. The **Download Modules** page appears (see Figure 12).

Figure 12 Update Center Wizard - Download Modules



- 7 After the progress bar reaches 100 percent, click **Next**.
The **View Certificates and Install Modules** page appears (see Figure 13).

Figure 13 Update Center Wizard - View Certificates and Install Modules

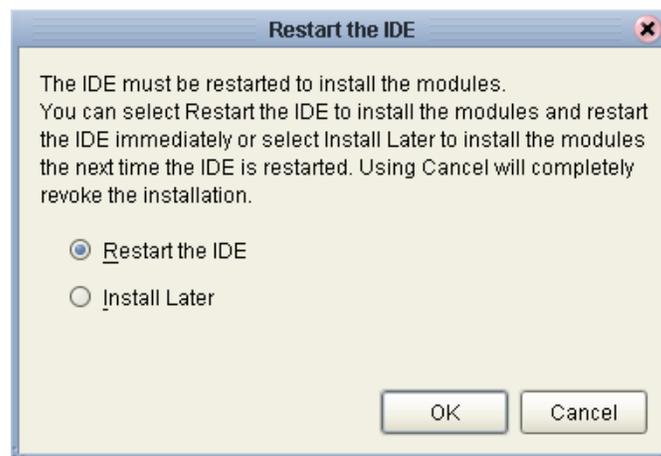


8 Click **Finish**.

The **Restart the IDE** dialog box appears (see Figure 14).

9 The modules that were installed must be reloaded for eIndex SPV to function properly. To restart the IDE and install the module, click **OK**.

Figure 14 Update Center Wizard - Restart the IDE



10 When Enterprise Designer is restarted, the eIndex SPV client and server Projects appear in the Project Explorer.

Important: Before you can use any pre-existing eIndex SPV Projects, you must regenerate the application and then rebuild and redeploy both the server and client Projects.

3.7 Final Steps

If you are using the Oracle eWay to connect to the database with a thin client, you do not need to perform any additional steps. If you are using the OCI driver with the Oracle eWay or if you are connecting to the database through the Sun Java System Application Server, you must make the Oracle drivers accessible.

For the Oracle eWay

If you are using the OCI driver with the Oracle eWay, you need to copy the OCI driver from the Oracle client to the Logical Host. The Oracle eWay will not recognize the database if these steps are not performed.

Important: A domain must be created for eIndex SPV before performing these steps.

To provide access to the Oracle thick client for IS implementations

- 1 Install the Oracle client on the Logical Host machine.
- 2 Copy **ojdbc14.jar** from <Oracle>\jdbc\lib to <logicalhost>\is\domains\<domain>\lib, where <Oracle> is the Oracle home directory, <logicalhost> is the Logical Host home directory, and <domain> is the name of the domain hosting the eIndex SPV application.
- 3 Start the domain (or restart it if it is already running).
- 4 Repeat steps 2 and 3 for each domain that requires access to the Oracle thick client.

For the Sun Java System Application Server

If you are defining database connectivity through the Sun Java System Application Server, you must install the Oracle client on the application server and then copy the **ojdbc14.jar** file to the **\lib** directory in the application server home directory.

See the *Sun Java System Application Server Enterprise Edition 8.1 2005Q2 Developer's Guide* for information on installing and implementing Oracle drivers and setting up the environment.

Configuring eIndex SPV

Several files are included in the eIndex SPV Project that configure the structure and processing logic of the master index. You can customize these files as needed for your implementation. This chapter provides an overview of the configuration files. For detailed information about the structure of these files and how to modify them, see the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*.

What's in This Chapter

- [Configurable Options](#) on page 50
- [About the eIndex SPV Configuration Files](#) on page 52
- [Modifying the eIndex SPV Configuration Files](#) on page 54
- [Match Engine Configuration Files](#) on page 55

4.1 Configurable Options

eIndex SPV provides a very flexible framework for creating a master index that is customized for your requirements. This section describes the configurable components of the master index, which includes the following.

- [Object Definition](#) on page 50
- [Enterprise Data Manager](#) on page 51
- [Query Definitions](#) on page 51
- [Standardization and Matching Rules](#) on page 51
- [Survivor Calculator](#) on page 51
- [Update Policies](#) on page 52
- [Field Validations](#) on page 52
- [Enterprise-wide Unique Identifiers](#) on page 52

4.1.1 Object Definition

By customizing the Object Definition file, you can configure the structure of the data stored in the master index. This file contains the configuration of all objects in the master index and their relationships to one another. It also defines the fields contained

in each object as well as certain attributes of each field, such as length, data type, whether it is required, whether it is a unique key, and so on. This file contains one parent object; all other objects must be child objects to that parent object. The object structure you define in the Object Definition file determines the structure of the database tables that will store object data and the structure of the method OTD in the eIndex SPV Project.

4.1.2 Enterprise Data Manager

The appearance of the EDM is defined in the Enterprise Data Manager file. You can customize this file by defining each field that appears on the EDM along with the properties of each field, such as the field type and length, field labels, format masks, and so on. You can also define the order in which objects and fields appear on the EDM pages, available search types and criteria, search results fields, reports, and so on.

4.1.3 Query Definitions

The queries used in the master index are all defined in the Candidate Select file. You can customize this file by configuring the types of queries used by the master index, including those that are performed from the EDM and those that are used during the match process. You can define both phonetic and alphanumeric searches for the EDM. By default, these are called basic searches. You can also define blocking queries, which define groups of criteria fields, for the match process (this type of search can be used in place of the phonetic search in the EDM as well).

4.1.4 Standardization and Matching Rules

The fields that are standardized or used for matching are configured in two files: Threshold and Match Field. In the Match Field file, you specify the match and standardization engines to use and configure information about the standardization and match process. This includes defining fields to be reformatted, normalized, or converted to their phonetic version; defining the data string to be passed to the match engine; and specifying a blocking query for matching. In the Threshold file, you define certain match parameters that define weight thresholds, how assumed matches are processed, and how potential duplicates are processed.

4.1.5 Survivor Calculator

The logic that determines the data to include in each object's SBR is defined in the Best Record file. You can configure this file by defining the formulas used by the survivor calculator to determine which fields from each system contain the most reliable data. The survivor calculator uses these formulas to generate the SBR for a given object. Survivor logic is defined in the **SurvivorHelperConfig** and **WeightedCalculator** sections of the Best Record file.

The SBR is defined by a mapping of fields from external system objects. Since there might be many external systems, you can optionally specify a strategy to select the SBR field from the list of external values. You can also specify any additional fields that

might be required by the selection strategy to determine which external system contains the best data, such as the object's update date and time.

4.1.6 Update Policies

You can create Java classes that define special processing to perform against a record when the record is created, updated, merged, or unmerged. These classes must be created in the Custom Plug-ins module and can be specified for each transaction type in the Best Record file.

4.1.7 Field Validations

By default, the Field Validation file defines validations for the local identifiers assigned by each external system. You can create custom rules to validate field values before they are saved to the master index database.

4.1.8 Enterprise-wide Unique Identifiers

The configuration of the EUIDs assigned by the master index is defined in the Threshold file. You can specify an EUID length, whether a checksum value is used for additional verification, and a "chunk size" (the number of EUIDs to be sent to the EUID generator at one time). Specifying a chunk size allows the EUID generator to obtain a block of EUIDs from the `sbyn_seq_table` database table so it does not need to query the table each time it generates a new EUID.

4.2 About the eIndex SPV Configuration Files

The files that configure the components of the master index define characteristics of the application, such as how data is processed, queried, and matched. These files configure the runtime components of the master index.

The configuration files include the following:

- **Object Definition File** on page 53
- **Candidate Select File** on page 53
- **Match Field File** on page 53
- **Threshold File** on page 53
- **Best Record File** on page 54
- **Field Validation File** on page 54
- **Security File** on page 54
- **Enterprise Data Manager File** on page 54

These files can only be modified in the XML Editor provided with eIndex SPV. This editor is described in the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*.

4.2.1 Object Definition File

The Object Definition file defines each object in the master index and their relationships to one another. It also defines the fields contained in each object, as well as certain properties of each field, such as length, data type, whether it is required, whether it is a unique key, and so on. This file contains one parent object; all other objects must be child objects to that parent object. The object structure defined in the Object Definition file determines the structure of the database tables that will store object data, the structure of the Java API, and the structure of the OTD generated for the Project.

4.2.2 Candidate Select File

This file configures the Query Builder component of the master index and defines the queries available for the master index. In this file, you define the types of queries that can be performed from the EDM and the queries that are used during the match process. You can define both phonetic and alphanumeric searches for the EDM. By default, these are called basic queries. You can also define blocking queries, which define blocks of criteria fields for the match process. When performing a blocking query, the master index queries the database using the criteria defined in each block, one at a time. After completing a query on the criteria defined in one block, it performs another pass using the next block of defined criteria. In the default configuration, a blocking query is used for the phonetic search in the EDM.

4.2.3 Match Field File

This file configures the Matching Service by defining standardization and matching fields for the master index. It also specifies the match and standardization engines to use and the query process for matching. Standardization includes defining fields to be reformatted, normalized, or converted to their phonetic version. For matching, you must also define the data string to be passed to the match engine. The rules you define for standardization and matching are dependent on the match engine in use. *Implementing the Sun SeeBeyond Match Engine with eIndex SPV* describes the rules for the SBME.

4.2.4 Threshold File

This file configures the eIndex Manager Service and defines attributes of the match process. This file specifies the match and duplicate thresholds and defines certain system parameters, such as how to process records above the match threshold, how to manage same system matches, update modes, and whether merged records can be updated. This file also specifies which query in the Query Builder to use for matching queries.

The Threshold file configures the EUIDs assigned by the master index by defining an EUID length, whether a checksum value is used for additional verification, and a “chunk size” (the number of EUIDs to be sent to the EUID generator at one time). Specifying a chunk size allows the EUID generator to obtain a block of EUIDs from the

sbyn_seq_table database table, so it does not need to query the table each time it generates a new EUID.

4.2.5 Best Record File

This file defines the logic that determines what data to include in each object's single best record (SBR). The Best Record file allows you to define formulas for determining which data should be considered the most reliable and how updates to the SBR will be handled. The survivor calculator uses these formulas to generate the SBR for each enterprise object. This logic is defined in the **SurvivorHelperConfig** and **WeightedCalculator** sections of the Best Record file. This file also allows you to define custom update procedures.

4.2.6 Field Validation File

By default, the Field Validation file specifies a Java class that defines certain validations for the local identifiers assigned by each external system. You can create Java classes that define custom rules to validate field values before they are saved to the database and then specify those classes in this file.

4.2.7 Security File

This file is a placeholder to be used in future versions.

4.2.8 Enterprise Data Manager File

The appearance of the EDM is defined in the Enterprise Data Manager file. This file defines each field that appears on the EDM along with the properties of each field, such as the field type and length, field labels, format masks, and so on. It also defines the order in which objects and fields appear on the EDM pages.

This file defines several additional attributes of the EDM, such as the types of searches available, whether wildcard characters can be used, the criteria for the searches, the results fields that appear, and whether an audit log is maintained of each instance data is accessed through the EDM.

Finally, this file defines certain implementation information, such as the report generator, code lookup classes, debugging options, security details, and the JNDI names for the report generator, master controller, and user code validation services.

4.3 Modifying the eIndex SPV Configuration Files

You might need to modify the configuration files after you review them. Make sure that when you modify the configuration files, you use the **Check Out** and **Check In** commands to maintain version control. If you open and modify a file without first checking the file out, a warning appears when you try to save the file. This warning lets you save and check out the file in one step. Also, be sure to verify that the modifications

are valid by verifying the XML syntax and using Enterprise Designer's validation function to validate the file against the schema. After modifying each file, save the changes to the Repository.

There are a few restraints on modifying these files. In addition to the general rules listed below, the match engine you choose might place other requirements on customizations. Be sure to review *Implementing the Sun SeeBeyond Match Engine with eIndex SPV* before modifying the Match Field file.

- All fields specified in any of the configuration files must be included in the Object Definition file.
- If you add fields to the object structure, make sure you add them to the survivor calculator in the Best Record file.
- If you define additional fields for normalization, parsing, or phonetic encoding, make sure to add the normalized, parsed, and phonetic fields to the Object Definition file and, optionally, the blocking query.
- After modifying any of the configuration files, you must regenerate the eIndex SPV application and redeploy the eIndex SPV Project. You must also refresh any Collaborations in client Projects that reference the eIndex SPV server Project.

4.4 Match Engine Configuration Files

If you implement the Sun SeeBeyond Match Engine, several match engine configuration files are added to the Project tree. You can customize matching logic and standardization information for the match engine by modifying these files. eIndex SPV provides a text editor for this purpose. For information about the structure of these files and how they can be modified, see *Implementing the Sun SeeBeyond Match Engine with eIndex SPV*.

Creating Custom Plug-ins

eIndex SPV provides the ability to create custom update procedures to be performed on a record during and after standard eIndex SPV processing. These procedures are defined as Java classes in the Custom Plug-ins module. This chapter describes how to create custom procedures using this module and how to configure eIndex SPV to use the custom procedures.

What's in This Chapter

- [Custom Processing](#) on page 56
- [Custom Components](#) on page 60
- [Match Processing Logic](#) on page 58
- [Custom Plug-in Exception Processing](#) on page 62
- [Implementing Custom Plug-ins](#) on page 62

5.1 Custom Processing

You can add custom processing to the master index using the Custom Plug-ins module of an eIndex SPV Project. Plug-ins can be used to customize field validations, update policies, match processing logic, and record retrieval, as well as create custom components for the master index, such as a custom phonetic encoders, block pickers, or query builders. The following sections describe custom plug-ins that define custom processing. These are explained more fully in the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*.

- [Update Policies](#) on page 56
- [Field Validations](#) on page 58
- [Match Processing Logic](#) on page 58

5.1.1 Update Policies

For the primary transactions performed by the master index you can define additional custom processing to perform against the record that results from a transaction. The policies you define are invoked by the Update Manager and are applied to the resulting records after they are processed by the survivor calculator. The modifications made to a record by an update policy determine how the record is stored in the database. Using

the Custom Plug-ins function, you can create additional Java classes to support the update policies you define.

eIndex SPV provides default custom plug-ins for each update policy. You can view and edit the Java code for each custom plug-in by expanding the Custom Plug-ins folder of the eIndex SPV Project, right-clicking the file to view, and then selecting **Open**.

Additional alias plug-ins are provided and are called by the custom update policies to process alias names after a transaction occurs.

Update policies are specified in the **UpdatePolicy** section of the Best Record file, and there are several different types. Each policy modifies an enterprise object (class **com.stc.eindex.objects.EnterpriseObject**) and must implement **com.stc.eindex.update.UpdatePolicy**, which contains one method, **applyUpdatePolicy**. The syntax is as follows:

```
public EnterpriseObject applyUpdatePolicy(EnterpriseObject before,  
EnterpriseObject after)
```

This method throws two exceptions: **com.stc.eindex.master.UserException** and **com.stc.eindex.objects.exception.ObjectException**.

Enterprise Merge Policy

The enterprise merge policy defines additional processing to perform after two enterprise objects are merged. The processing defined in this policy acts against the surviving record of the merge. In the **EnterpriseMergePolicy** element in the Best Record file, enter the fully qualified name of this custom plug-in.

Enterprise Unmerge Policy

The enterprise unmerge policy defines additional processing to perform after an unmerge transaction occurs. The processing defined in this policy acts against the surviving record of the merge transaction that was unmerged. In the **EnterpriseUnmergePolicy** element of the Best Record file, enter the fully qualified name of this custom plug-in.

Enterprise Update Policy

The enterprise update policy defines additional processing to perform after a record is updated. In the **EnterpriseUpdatePolicy** element of the Best Record file, enter the fully qualified name of this custom plug-in.

Enterprise Create Policy

The enterprise create policy defines additional processing to perform after a new record is inserted into the master index database. In the **EnterpriseCreatePolicy** element of the Best Record file, enter the fully qualified name of this custom plug-in.

System Merge Policy

The system merge policy defines additional processing to perform after two system objects are merged. The processing defined in this file acts against the surviving

enterprise record of the merge (and not the system record). In the **SystemMergePolicy** element of the Best Record file, enter the fully qualified name of this custom plug-in.

System Unmerge Policy

The system unmerge policy defines additional processing to perform after system objects are unmerged. The processing defined in this file acts against the surviving enterprise record of the system merge transaction that was unmerged. In the **SystemUnmergePolicy** element in the Best Record file, enter the fully qualified name of this custom plug-in.

Undo Assumed Match Policy

The undo assumed match policy defines additional processing to perform after an assumed match transaction is reversed. In the **UndoAssumeMatchPolicy** element in the Best Record file, enter the fully qualified name of this custom plug-in.

5.1.2 Field Validations

You can define validations to be performed against certain fields before information is entered into the master index database. Once you create the custom plug-ins containing the validation logic, enter the name of the plug-in in the Field Validation file. Follow these guidelines when implementing custom field validators.

- The custom validation classes must implement **com.stc.eindex.objects.validation.ObjectValidator**.
- The exception thrown is **com.stc.eindex.objects.validation.exception.ValidationException**.

One default field validator, **validate-local-id**, is provided to validate system and local ID fields before processing data into the database. This is described in the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*.

5.1.3 Match Processing Logic

You can implement custom plug-ins that customize the way the execute match methods process data into the master index.

Execute Match Methods

Match processing is performed by calling one of the following “execute match” functions from the MasterController class.

- `executeMatch`
- `executeMatchUpdate`
- `executeMatchDupRecalc`
- `executeMatchUpdateDupRecalc`
- `executeMatchGui` (this method is only called by the EDM)

These classes contain standard logic for processing records through the master index database, weighting incoming records against existing records, and then using those weights to determine whether to insert a new record or update an existing record. In addition to configuring the match processing logic in the Threshold file, you can customize certain aspects of the processing logic using custom plug-ins that contain functions found in the `ExecuteMatchLogics` class.

Custom Logic Methods

There are five decision branches where custom logic can be inserted. At specific points in match processing, the execute match method looks for the value of the methods listed below. For more information about the methods, see the Javadocs provided with eIndex SPV. (The methods are contained in the `ExecuteMatchLogics` class in the package `com.stc.eindex.master`.) For information about where the decision points are reached in the processing logic and how to change the logic, see Appendix A of the *Sun SeeBeyond eIndex Single Patient View Reference Guide*. The following methods specify the custom logic.

- **bypassMatching** - indicates whether to perform the match process on incoming records or to bypass the match process
- **disallowAdd** - indicates whether an incoming message can be inserted as a new record
- **disallowUpdate** - indicates whether an incoming record can update an existing record
- **rejectAssumedMatch** - indicates whether to accept or reject an assumed match of two records
- **rejectUpdate** - indicates whether to accept or reject an update to an existing record

Custom Logic Plug-in Requirements

The custom plug-ins you create to define custom execute match logic must inherit the `ExecuteProcessingLogic` class. In addition, the following classes must be imported into the custom plug-in.

- `com.stc.eindex.objects.SystemObject;`
- `com.stc.eindex.objects.EnterpriseObject;`
- `com.stc.eindex.objects.exception.ObjectException;`
- `com.stc.eindex.master.ExecuteMatchLogics;`
- `com.stc.eindex.master.CustomizationException;`

Threshold File Customization

If you create a custom plug-in that defines custom processing logic for the execute match methods, you must specify those custom plug-ins in the Threshold file of the eIndex SPV Project. If you create a plug-in for customizing logic in the execute match methods used by Collaborations or Business Processes, specify the name of that class in

the logic-class element. If you create a plug-in for the EDM, specify the name of that class in the logic-class-gui element. For example:

```
<logic-class>com.stc.eindex.user.CustomCollaboration</logic-class>  
<logic-class-gui>com.stc.eindex.user.CustomEDM</logic-class-gui>
```

For more information about the Threshold file, see the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*.

5.2 Custom Components

eIndex SPV provides a flexible framework, allowing you to create custom Java classes to plug in to most eIndex SPV components. This section provides a brief list and descriptions of some components for which you can create custom classes.

- **Survivor Calculator** on page 60
- **Query Builder** on page 60
- **Block Picker** on page 61
- **Pass Controller** on page 61
- **Match Engine** on page 61
- **Standardization Engine** on page 61
- **Phonetic Encoders** on page 62

5.2.1 Survivor Calculator

The survivor calculator determines which field values from the various system records will populate the SBR for the enterprise record. You can create a custom survivor calculator class that selects the surviving field values. Your custom class must implement the survivor calculator interface. Call **selectField** in **com.stc.eindex.survivor.SurvivorStrategyInterface** to return the SBR value for each field. For more information about the classes and methods to use, see the Javadocs provided with eIndex SPV. The primary classes are contained in the **com.stc.eindex.survivor** package.

5.2.2 Query Builder

The query builder defines the different types of queries that can be used in the master index. You can implement custom queries using custom plug-ins. To create a new query builder, you must define a class that extends the base abstract **com.stc.eindex.querybuilder.QueryBuilder** and then specify that class in a **query-builder** element in the Candidate Select file. The exception thrown is **com.stc.eindex.querybuilder.QueryBuilderException**. The following methods must be implemented. For more information about query-related Java classes, see the Javadocs provided with eIndex SPV.

- **init** - This method receives the XML elements after the **config** element so the query builder can read its custom configuration.
- **getApplicableQueryIds** - This method returns an array of string IDs indicating the query objects that can be generated given the available criteria. For example, in the blocking configuration, the unique ID of each block definition is the string that is returned by **getApplicableQueryIds**.
- **buildQueryObject** - This method constructs the query object based on one of the applicable query IDs provided as an input argument.

5.2.3 Block Picker

The block picker chooses the block definition to use for the next matching pass. You can create a custom block picker class to select query blocks in a customized manner. Specify the fully qualified name of this custom plug-in for the **block-picker** element of the Match Field file. Follow these guidelines when implementing a custom block picker.

- Implement the **com.stc.eindex.matching.BlockPicker** interface to select the blocks in the desired order.
- If none of the remaining blocks should be executed, throw a **NoBlockApplicableException** from the **pickBlock** method.

5.2.4 Pass Controller

The matching process can be executed in multiple stages. After a block is evaluated, the pass controller determines whether the results found are sufficient or if matching should continue by performing another match pass. Specify the name of the custom Pass Controller in the **pass-controller** element of the Match Field file. Follow these guidelines when implementing a custom pass controller.

- Implement the **com.stc.eindex.matching.PassController** interface to evaluate whether to do another pass or not.
- Return **true** from **evalAnotherPass** to specify that an additional pass be performed; return **false** to specify that no additional passes are performed.

5.2.5 Match Engine

You can define classes to connect to a custom match engine instead of the SBME. Specify the names of the custom classes you create in the **matcher-api** and **matcher-config** elements of the Match Field file. Follow these guidelines when implementing custom match engine classes.

- Implement the **com.stc.eindex.matching.MatcherAPI** interface to communicate with the match engine.
- Implement the **com.stc.eindex.matching.MatchEngineConfiguration** interface to retrieve any configuration values the match engine requires for initialization.

5.2.6 Standardization Engine

You can define classes to connect to a custom standardization engine instead of the SBME. Specify the names of the custom classes you create in the **standardizer-api** and **standardizer-config** elements of the Match Field file. Follow these guidelines when implementing custom standardization engine classes.

- Implement the **com.stc.eindex.matching.StandardizerAPI** interface to communicate with the standardization engine.
- Implement the **com.stc.eindex.matching.StandardizerEngineConfiguration** interface to retrieve any configuration values the standardization engine requires for initialization.

5.2.7 Phonetic Encoders

The master index supports several phonetic encoders, and you can define custom classes to implement additional phonetic encoders if needed. Specify the names of the custom classes you create in the **encoder-implementation-class** element of the Match Field file. When creating a custom phonetic encoder class, implement the **com.stc.eindex.phonetic.PhoneticEncoder** interface.

5.3 Custom Plug-in Exception Processing

If a custom plug-in throws an exception of the class **ObjectException** or **SystemObjectException**, multiple stack traces are logged in the server log file, which can make operational management tasks more difficult. For cases where you do not want stack traces to be logged, configure your custom plug-ins to throw exceptions of the class **UserException** or one of its derived classes (**DataModifiedException** or **ValidationException**). This is useful for user errors on the Enterprise Data Manager (EDM). When one of these exceptions is thrown, no stack trace is entered in the log file but an error message still appears on the EDM.

For more information about these exception classes, see the Javadoc provided with eIndex SPV.

5.4 Implementing Custom Plug-ins

eIndex SPV provides a simple method of incorporating custom Java code into an eIndex SPV application via the **Custom Plug-ins** module.

Creating Custom Plug-ins

Custom plug-ins contain Java code that you create to tailor how messages are processed in the eIndex SPV system. You can create as many plug-ins as you need to carry out the custom processes.

To create custom plug-ins

- 1 In the eIndex SPV Project, click the **Custom Plug-ins** folder and then right-click.
- 2 Select **New** from the context menu that appears.
- 3 Enter the name of the custom plug-in and then click **OK**.
The custom plug-in file appears in the Java Source Editor with the first line already entered ("**package com.stc.eindex.user;**").
- 4 Create the custom processing rules using Java code.
- 5 Close and save the file.
- 6 Repeat these steps for each plug-in you need to create.
- 7 Build the custom plug-ins, as described under "**Building Custom Plug-ins**".
- 8 Specify the name of the custom plug-in in the appropriate eIndex SPV configuration file.

Note: Custom plug-ins are created in the **com.stc.eindex.user** package, and the name you specify for the plug-in is the name of the Java class created for the plug-in. When you specify the custom plug-in in the configuration files, use the fully qualified class name. For example, if you create a custom plug-in named "MergePolicy", the value to enter for the class in the Best Record file is "**com.stc.eindex.user.MergePolicy**".

Building Custom Plug-ins

In order for the custom plug-ins you create to become a part of the eIndex SPV application, you must build the plug-ins. This compiles the Java code and incorporates it into the application files.

Important: Compiling errors for custom plug-ins are not written to a log. An error message trace appears on a console window alerting you to the errors that occurred.

To build custom plug-ins

- 1 In the eIndex SPV Project, click the **Custom Plug-ins** folder and then right-click.
- 2 Select **Build** from the context menu that appears.

Important: If you modify a custom plug-in file after it has been checked in, be sure to check the file out before making any changes; otherwise, any changes will be lost when you try to save the file. Be sure to rebuild the plug-in after you save the changes and regenerate the application to incorporate the changes. Regenerating also rebuilds all custom plug-ins.

Building the Application

If you make any changes to the default configuration files for eIndex SPV, you must generate the application to customize the remaining runtime components. This chapter gives instructions for generating the eIndex SPV application and describes the Project components that are updated when you generate.

What's in This Chapter

- [Generated Application Components](#) on page 64
- [Generating the Application](#) on page 65

6.1 Generated Application Components

Generating the master index application is the process that actually creates the indexing application. Several custom components are updated in the Project along with the executable files for the application. eIndex SPV uses the Object Definition to update these components based on the configuration you defined. The generated components include the following.

- **Database scripts** - The scripts for creating and dropping tables and indexes are updated based on the Object Definition file.
- **Custom Plug-ins** - Generating the application rebuilds custom plug-ins and incorporates any customized processing rules defined by custom plug-ins into the application files.
- **Outbound OTD** - This component defines the outbound data structure and includes general OTD methods. The data structure is based on the Object Definition file.
- **Method OTD** - The method OTD includes the Java methods you need to process data through the master index. These are customized for your application based on the Object Definition file.
- **Business Process methods** - Business Process methods can be used when processing data through eInsight BPM rather than a Collaboration. They include a subset of the method OTDs and are based on the Object Definition file. Use these methods for eVision web pages as well.
- **Application JAR files** - These files are used by the web-based interface (EDM) and any client Projects that access the master index.

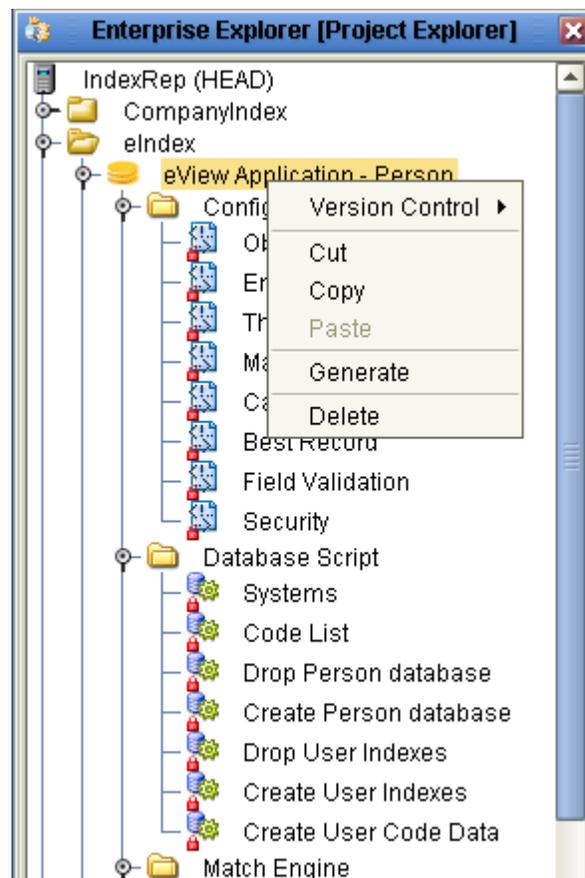
6.2 Generating the Application

Once all modifications to the configuration files are complete and any custom plug-ins are built, generate eIndex SPV to update the components listed above. If you modify any of the configuration files, match and standardization engine files, or custom plug-ins after you generate the application, you must regenerate the application to update the custom components.

To generate the application

- 1 Save all configuration changes to the Repository.
- 2 Right-click the eIndex SPV application in the **Project Explorer** pane to display the application context menu (shown in Figure 15).

Figure 15 Generate Context Menu



- 3 Select **Generate**. eIndex SPV updates the Project components. This might take a few minutes.

Note: When you regenerate an application, a warning dialog appears stating that the application already exists. Click **Yes** on this dialog to recreate the generated components.

- 4 Save the new components to the Repository.
- 5 If there are any client Projects with Collaborations that reference the eIndex SPV server Project, refresh those Collaborations.
- 6 If there were any changes to how incoming data is processed (such as changes to the output OTD), re-import the .jar files.
 - A Open the Collaboration in the Collaboration Editor (Java) and click **Import JAR Files**. The Add/Remove Jar Files dialog appears.
 - B For each eIndex SPV .jar file, highlight the filename on the Add/Remove Jar Files dialog, and then click **Remove**.
 - C For each file to re-import, click **Add**, double-click the eIndex SPV server Project, select the .jar file, and then click **Import**.

For more information about the Add/Remove Jar Files dialog, see the *Sun SeeBeyond eGate Integrator User's Guide*.

- 7 If you are using command line reports, do the following:
 - A Export the regenerated **Person_stc_eindex_client.jar** and **Person_stc_eindex_util.jar** files to the **lib** subdirectory in the reports home directory.
 - B In the **lib** subdirectory, rename the file **stc_eindex_client.jar** to **stc_eindex_client.jar** and rename the file **stc_eindex_util.jar** to **stc_eindex_util.jar**.

Note: *If any errors occur while compiling the application, an error warning appears and the errors are logged in the IDE log file located at <edesigner_home>\usrdir\system\ide.log.*

Creating the Database

The eIndex SPV Project includes several database scripts to create the master index tables, indexes, and startup data for the new database. Additional scripts are created for testing purposes. This chapter provides information about designing the database, modifying the scripts, and using the scripts to create the index-specific database tables and startup data.

What's in This Chapter

- [Database Scripts](#) on page 67
- [Requirements](#) on page 67
- [Database Structure](#) on page 69
- [Designing the Database](#) on page 69
- [Creating the Database](#) on page 71
- [Deleting Tables and Indexes](#) on page 79

7.1 Database Scripts

The database scripts include scripts for defining code lists and external systems, for creating tables and indexes, and scripts for dropping tables and indexes. These scripts appear under the **Database Script** node of the eIndex SPV Project, and are named **Systems**, **Code List**, **Create User Indexes**, **Drop User Indexes**, **Create User Code Data**, **Create Person database**, and **Drop Person database**. You can modify these scripts as needed to customize the tables, indexes, startup data, and database distribution. You can also create new database scripts if needed.

7.2 Requirements

When configuring the master index database, there are several factors to consider, including basic software requirements, operating systems, disk space, and so on. This section provides a summary of requirements for the database. For more detailed information about designing and implementing the database, refer to the appropriate Oracle documentation. The person responsible for the database configuration should be

an Oracle database administrator familiar with the master index database and with your data processing requirements.

7.2.1 Database Platform Requirements

The master index database can be run on Oracle 9i or 10g. You must have this software installed before beginning the database installation. Make sure you also install the latest Oracle patches for the version you are using.

7.2.2 Operating System Requirements

The database can be installed on any operating system platform supported by the version of Oracle you are using. See the Oracle documentation for more information.

7.2.3 Hardware Requirements

This section describes the minimum recommended hardware configuration for a 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 Windows database server, the following configuration is recommended as a *minimal* installation:
 - ♦ Windows 2000 SP3 or later or Windows XP SP2
 - ♦ Pentium 266 or later
 - ♦ 512 MB RAM (increase this based on the number of users, connections to the database, and volume)
 - ♦ 3 GB disk space plus an additional 2 KB for each system record to be stored in the database (note that this is a conservative estimate per system record, assuming that most records do not contain complete data). This depends on the Oracle environment you install. Enterprise Edition can take up to 5 GB.
 - ♦ 256-color video
- 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
 - ♦ 2 GB disk space plus an additional 2 KB for each system record to be stored in the database (note that this is a conservative estimate per system record, assuming that most records do not contain complete data)

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 2.5 GB of disk space.

7.3 Database Structure

The master index database contains some common tables that are created for all implementations and some that are customized for each implementation. The common tables include standard Oracle tables and supporting tables, such as `sbyn_seq_table`, `sbyn_common_header`, and `sbyn_common_detail`. These tables do not store information about the enterprise object structure you defined. The names of the tables that store information about the enterprise object are customized based on the object structure.

Two tables store information about the primary, or parent, object you defined: `sbyn_<parent_object>` and `sbyn_<parent_object>sbr`, where `<parent_object>` is the name you specified for the parent object in Object Definition. The `sbyn_<parent_object>` table stores parent object data from each local system and the `sbyn_<parent_object>sbr` table stores the parent object data contained in the SBRs. Similar tables are created for each child object you defined in the object structure.

For a complete description of the database tables, see Chapter 3, “The Database Structure”, in the *Sun SeeBeyond eIndex Single Patient View Reference Guide*.

7.4 Designing the Database

In designing the database, there are several factors to consider, such as the volume of data stored in the database and the number of transactions processed by the database daily. The eIndex SPV database should be created in its own tablespaces. The following sections describe some of the analyses to perform along with things to consider when designing the database.

7.4.1 Designing for Performance Optimization

The Oracle installation guides provide detailed information about installing the database software for optimal performance. The Oracle administrator’s guides include information about monitoring and fine-tuning your database, including tuning memory, swap space, I/O, CPU usage, block and file size, and so on.

7.4.2 Data Analysis

Before beginning the master index implementation, you performed an analysis of the legacy data to help you define the object structure and the attributes of each field. You can use this data analysis to determine the amount of data that will be stored in the database, which will help you size the master index database and decide how to best distribute the database. Knowing the volume of existing data plus the expected daily

transaction volume will help you plan the requirements of the database server, such as networking needs, disk space, memory, swap space, and so on.

The data analysis will help you define the processing codes and the descriptions to define in the common tables, and should help you determine any default values that have been entered into certain fields that could skew the matching probability weights.

7.4.3 Common Table Data

Common table data analysis involves gathering information about the abbreviations used for specific data elements in each sending system, such as system codes and codes for certain attributes about the patients in your database, such as language, race, and marital status codes. The processing codes and their descriptions are stored in a set of database tables known as common maintenance tables. The eIndex SPV Project includes a script to help you load the processing codes into the database.

When an enterprise object appears on the EDM, the master index translates the processing codes defined in the common tables into their descriptions so the user is not required to decipher each code. The data elements stored in the common maintenance tables are also used to populate the drop-down lists that appear for certain fields in the EDM. Users can select from these options to populate the associated fields.

7.4.4 User Code Data

User code data analysis involves gathering information about the abbreviations used for specific data elements in each sending system for a field whose format or possible values are constrained by a separate field. For example, if you store credit card information, you might have a drop-down list in the Credit Card field for each credit card type. The format of the field that stores the credit card number is dependent on the type of credit card you select. You could also use user code data to validate cities with postal codes. The abbreviations and related constraint information are stored in the `sbyn_user_code` table. A sample script, **Create User Code Data**, is provided to help you insert data into this table.

7.4.5 Considerations

When you create the master index database, you need to consider several factors, such as sizing, distribution, indexes, and extents. By default, all of the master index database tables are installed in the Oracle “system” tablespace. You should install these tables into different tablespaces, depending on the original size and expected volume of the database.

Database Sizing

To begin the database installation, you first create an Oracle database instance using Oracle configuration tools. Use these tools to define the tablespace and extent sizing for the database.

Database Distribution

The Oracle configuration tools allow you to define the distribution of your system tables, data tables, rollback logs, dump files, control files, and so on. Use internal policies regarding relational database distribution to determine how to best distribute your eIndex SPV database.

Database Indexes

By default, indexes are defined for the following tables: `sbyn_appl`, `sbyn_common_header`, `sbyn_common_detail`, `sbyn_enterprise`, `sbyn_transaction`, `sbyn_assumedmatch`, `sbyn_potentialduplicates`, `sbyn_audit`, and `sbyn_merge`. Index scripts are also created for indexing the fields included in the default blocking query in the Candidate Select file. You can create additional indexes against the database to optimize the searching and matching processes. At a minimum, it is recommended that all combinations of fields used for blocking or matching be indexed. For each query block defined in the blocking query, create an index containing the fields in that block.

7.5 Creating the Database

Once you have customized the configuration files and generated the eIndex SPV Project in Enterprise Designer, you can create the master index database. Before you begin, make sure you have Oracle installed on the database server. Follow these steps to create the database.

- [Step 1: Analyze the Database Requirements](#) on page 71
- [Step 2: Create an Oracle Database and User](#) on page 72
- [Step 3: Customize the Database Scripts](#) on page 72
- [Step 4: Modify the Database](#) on page 77
- [Step 5: Specify a Starting EUID \(optional\)](#) on page 78

7.5.1 Step 1: Analyze the Database Requirements

Before you begin to create the master index database, you must perform a thorough analysis of the legacy data to be stored in the database and determine the amount of data that will be processed daily. During the analysis, be sure to define the processing codes that need to be stored in the common maintenance tables and the systems that will share data with the master index. You should also know the length and format of the local IDs assigned by each system.

An Oracle database administrator who is familiar with your data and processing requirements should perform this task.

7.5.2 Step 2: Create an Oracle Database and User

Before beginning this step, be sure that the correct version of Oracle is installed on the database server (version 9i). To install the database, you can use a standard Oracle tool, such as the Database Configuration Assistant, which will lead you through the database configuration process. During this step, you will define tablespaces, including their sizes and locations; extents; and dump file, log file, and rollback file sizes and locations. Make sure these issues have been thoroughly analyzed and designed before creating the database.

When you create the database, you should also create a user that will be used to create the database structure and to connect to the database through the EDM and through eGate. Assign this user to the “connect” and “resource” roles for the eIndex SPV tablespaces. For example:

```
create user <username> identified by <password>;
grant connect, resource to <username>;
commit;
```

where <username> is the login ID of the administrator user and <password> is the login password of the administrator user. If you prefer to assign individual permissions to the user instead of roles, the following permissions are needed.

- alter any index
- alter any procedure
- alter any table
- alter any trigger
- create any index
- create procedure
- create session
- create table
- create trigger
- create view
- delete any table
- drop any index
- drop any procedure
- drop any table
- drop any trigger
- drop any view
- insert any table
- select any table
- update any table

7.5.3 Step 3: Customize the Database Scripts

The database script that installs the database components specific to eIndex SPV is customized based on any changes you made to the object structure in the Object Definition file. You can modify any of the database script as needed.

Defining Indexes

To optimize data processing in the master index, you can define additional indexes for the database tables that store object data. Sun recommends defining indexes for each field used for searching, blocking, or matching. You can define these indexes in the **Create User Indexes** file or create a new script.

To define an index

- 1 In the Project Explorer pane, expand the **Database Script** node and then double-click the **Create User Indexes** file.

The file opens in the text editor.

- 2 Do any of the following:
 - ♦ Remove an existing index definition (not recommended).
 - ♦ Create new index definitions for the required fields.
 - ♦ Modify an existing index definition.
- 3 Save and close the **Create User Indexes** file.

Defining Systems

The **Systems** file in the eIndex SPV Project defines one default source system for eIndex SPV. You can define additional systems as needed or delete the default system.

To define a system

- 1 In the Project Explorer pane, expand the **Database Script** node and then double-click the **Systems** file.

The file opens in the text editor.

- 2 For each “insert” statement, modify the values clause according to the column descriptions in Table 2. For example:

```
INSERT into sbyn_systems (systemcode, description, status,
id_length, format, input_mask, value_mask, create_date,
create_userid)
VALUES ('ARS', 'Automated Registration System', 'A', 10, '[0-
9]{10}', 'DDD-DDD-DDDD', 'DDD^DDD^DDDD', sysdate, 'admin');
```

- 3 If needed, create additional “insert” statements for any systems that are not already defined.
- 4 Save and close the **Systems** file.

Table 2 Columns in the sbyn_systems Table

Column	Description
systemcode	The unique processing code of the system.
description	A brief description of the system, or the system name. <i>Tip: It is best to keep this short if possible; these values appear in the tree views on the Enterprise Data Manager and can cause the box containing the tree views to increase in width to accommodate all characters.</i>
status	The status of the system in the master index system. Specify “A” for active, or “D” for deactivated.

Table 2 Columns in the sbyn_systems Table

Column	Description
id_length	<p>The length of the local identifiers assigned by the system. This length does not include any additional characters added by the input mask.</p> <p>Note: <i>The default maximum length of the LID database columns is 25. If the system generates longer local IDs, be sure to increase the length of all LID columns in the database.</i></p>
format	<p>The required data pattern for the local IDs assigned by the system. For more information about possible values and using Java patterns, see “Patterns” in the class list for java.util.regex in the Javadocs provided with the J2SE Platform. Note that the pattern specified here might be further restricted by the input mask described below.</p>
input_mask	<p>A mask used by the EDM to add punctuation to the local ID. For example, you can add an input mask to display the local IDs with hyphens or constant characters. To define an input mask, enter a character type for each character in the field and place any necessary punctuation between the types. For example, to insert a hyphen after the second and fifth characters in an 8-digit ID, the input mask would be DD-DDD-DDD. The following character types can be used; any other characters are treated as constants.</p> <ul style="list-style-type: none"> ▪ D - indicates a numeric character. ▪ L - indicates an alphabetic character. ▪ A - indicates an alphanumeric character. <p>If you use an input mask, you should also define a value mask (see below) to remove the punctuation from the stored value.</p>
value_mask	<p>A mask used to strip any extra characters that were added by the input mask. This mask ensures that data is stored in the database in the correct format. To specify a value mask, type the same value entered for the input mask, but type an “x” in place of each punctuation mark. Using the 8-digit input mask described above, you would specify a value mask of DDxDDxDDD. This strips the hyphens before storing the ID.</p>
create_date	<p>The date the system information was inserted into the database. You can specify “sysdate” for this column, or use the variables defined at the beginning of the sample script.</p>
create_userid	<p>The logon ID of the user who inserted the system information into the database. You can enter the logon ID or use the variables defined at the beginning of the sample script.</p>

Defining Code Lists

The **Code List** file of the eIndex SPV Project defines default processing codes and drop-down list descriptions for the common data tables in the database. The information you define will be used to translate processing codes from incoming messages into descriptions for the EDM fields and to create drop-down lists for EDM fields.

The Code List file contains a stanza for each type of common table data element for which you use processing codes. You can create additional common table data types and additional common table data elements. This script inserts data into two tables: `sbyn_common_header`, which lists the types of common table data, and `sbyn_common_detail`, which lists each common table data element. You must define a type before you can define the elements for that type.

Note: *The codes you specify in this file can be no longer than eight characters (the codes are the second value in the value list for each common table data type and data element).*

To customize common table data

- 1 In the Project Explorer pane, expand the **Database Script** node and then double-click the **Code List** file.
- 2 The file opens in the text editor.
- 3 In the **Code List** file, scroll to the following line.

```
codes tCodeList := tCodeList(
```

The statements following this line can be customized.

- 4 Add or modify the elements of each stanza as needed. A sample stanza is shown below.

```
-- **** PHONTYPE ****
tCode('L', 'PHONTYPE', 'TELEPHONE TYPE'),
tCode('V', 'H', 'HOME'),
tCode('V', 'M', 'MOBILE'),
tCode('V', 'F', 'FAX'),
tCode('V', 'O', 'OFFICE'),
```

Note: *Do not modify the “L” or “V” in each row. These characters define whether the information is inserted into the `sbyn_common_header` or `sbyn_common_detail` table. Following the table indicator is the processing code, and the final item in each row is a description.*

- 5 In the last code module stanza, make sure each line except the last contains a comma at the end. For example:

```
-- **** ADDRTYPE ****
tCode('L', 'ADDRTYPE', 'ADDRESS TYPE'),
tCode('V', 'H', 'HOME'),
tCode('V', 'B', 'BUSINESS'),
tCode('V', 'M', 'MAILING')
```

- 6 Save and close the **Code List** file.

Defining User Code Lists

If you specified a value for the **constraint-by** and **user-code** elements of a field, you must define the user code values for those fields. Below is a sample insert statement for the `sbyn_user_code` table. eIndex SPV provides this sample in a database script for you to modify.

```
insert into sbyn_user_code (code_list, code, descr, format,
input_mask, value_mask)
values ('AUXIDDEF', 'CC', 'CREDIT CARD', '[0-9]{16}', 'DDDD-DDDD-
DDDD-DDDD', 'DDDD^DDDD^DDDD^DDDD');
```

To define a user code list

- 1 In the Project Explorer pane, expand the **Database** node and then double-click the **Create User Code Data** file.
The file opens in the text editor.
- 2 Use the above sample to define a value for the user code drop-down list and the required format for the dependent fields.
- 3 Repeat step 2 for each drop-down list value and type (for example you might have one list for credit cards and another for postal codes and their corresponding cities).
- 4 Save and close the **Create User Code Data** file.

Table 3 SBYN_USER_CODE Table Description

Column Name	Description
code_list	The code list name of the user code type (using the credit card example above, this might be similar to “CREDCARD”). This column links the values for each list.
code	The processing code of each user code element.
description	A brief description or name for the user code element. This is the value that appears in the drop-down list.
format	The required data pattern for the field that is constrained by the user code. For more information about possible values and using Java patterns, see “Patterns” in the class list for java.util.regex in the Javadocs provided with the J2SE Platform. Note that the pattern might be further restricted by the value of the input mask described below.
input-mask	A mask used by the EDM to add punctuation to the constrained field. For example, the input mask DD-DDD-DDD inserts a hyphen after the second and fifth characters in an 8-digit ID. If you use an input mask, you should also use a value mask to strip the punctuation for database storage (see below). The following character types can be used. <ul style="list-style-type: none"> ▪ D—Numeric character ▪ L—Alphabetic character ▪ A—Alphanumeric character

Table 3 SBYN_USER_CODE Table Description

Column Name	Description
value-mask	A mask used to strip any extra characters that were added by the input mask for database storage. The value mask is the same as the input mask, but with an "x" in place of each punctuation mark. Using the input mask described above, the value mask is DDxDDDxDDD . This strips the hyphens before storing the ID.

Creating a Custom Script

You can insert additional information into the database by creating a custom script under the **Database Script** node. For information about the structure of the master index database, see the *Sun SeeBeyond eIndex Single Patient View Reference Guide*.

To create a custom script

- 1 In the eIndex SPV Project, right-click the **Database Script** node.
- 2 In the Database context menu, select **New**.
- 3 Enter the name of the script, and then click **OK**.
The new script appears under the **Database Script** node.
- 4 Double-click the new script.
The text editor appears.
- 5 In the text editor, create the SQL script to insert the custom data.
- 6 Close the file and select **Yes** to save the data.

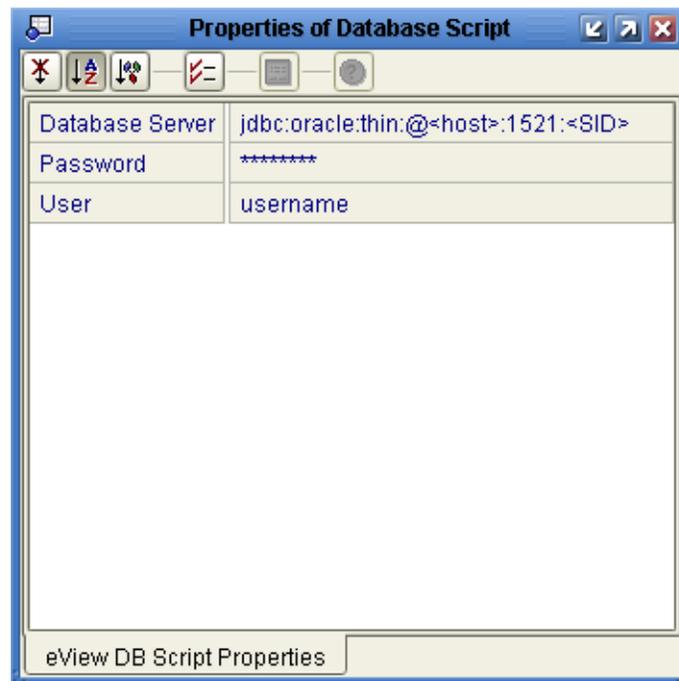
7.5.4 Step 4: Modify the Database

After you create the database instance and customize the database scripts, you can create the master index tables and insert the custom data.

To modify the database

- 1 In the eIndex SPV Project, right-click the **Database Script** node, and then select **Properties** from the **Database Script** context menu.
The **Properties of Database Script** dialog appears as shown in Figure 16.

Figure 16 Database Properties Dialog



- 2 In the **Properties of Database Script** dialog, enter the following information:
 - ♦ In the **Database Server** field, change *<host>* to the database server name and change *<SID>* to the SID name of the database you created in **“Step 2: Create an Oracle Database and User”**. You can use “localhost” as the database server name if the database resides on the same machine as the Enterprise Designer.
 - ♦ In the **Password** field, enter the password of the administrator user you created when you created the database (creating an administrator user is described under **“Step 2: Create an Oracle Database and User”** on page 72).
 - ♦ In the **User** field, enter the administrator user’s logon ID.

Important: *Make sure you enter the database logon credentials for the administrator user you created. You cannot use the logon credentials for the default system user (the database tables will be created, but the eIndex SPV application will not function correctly).*

- 3 Close the dialog by clicking the “X” icon in the upper right corner of the dialog.
- 4 Right-click **Create Person Database**, and then select **Run**. On the confirmation dialog, click **OK**.
- 5 For each additional script to run against the database, right-click the name of the script, and then select **Run**. On the confirmation dialog, click **OK**.

7.5.5 Step 5: Specify a Starting EUID (optional)

By default, the EUIDs assigned by the master index start with "0", with padded zeroes added to the left to make the EUID number the correct length (for more information, see "Threshold Configuration" in the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*). You can modify this numbering format by changing the value of the seq_name column of the sbyn_seq_table database table where the sequence name is "EUID". For example:

```
update sbyn_seq_table set seq_count=1000000001 where  
seq_name='EUID';
```

7.6 Deleting Tables and Indexes

Scripts are provided to drop the default database tables and indexes created in [Step 4: Modify the Database](#) on page 77. This is useful while testing the master index implementation.

To delete tables and indexes

- 1 To drop the indexes created by the **Create User Indexes** script:
 - A Right-click **Drop User Indexes**.
 - B Select **Run**.
 - C On the confirmation dialog, click **OK**.
- 2 To drop the database tables:
 - A Right-click **Drop Person database**.
 - B Select **Run**.
 - C On the confirmation dialog, click **OK**.
- 3 If the database is running on the Oracle 10g platform, open a SQL prompt and run the following command.

```
PURGE RECYCLEBIN;
```

Defining Connectivity Components

Once the eIndex SPV server Project is generated, you can customize the Connectivity Map that defines the application. You can also create and customize connectivity components that define how data is transformed, routed, and processed between the master index, Business Processes or Collaborations, and external systems. This chapter describes the connectivity components used in conjunction with eIndex and how to configure those components using the Enterprise Designer tools.

What's in This Chapter

- [Connectivity Overview](#) on page 80
- [Defining Connectivity Components](#) on page 82

8.1 Connectivity Overview

The Project that defines the eIndex SPV application is known as the *server* Project; the Projects that define external system or Business Process connectivity with eIndex SPV are known as *client* Projects.

Data can be processed by the master index in four ways.

- 1 Data is processed through the EDM. This process is defined by the Connectivity Map in the eIndex SPV server Project.
- 2 Data is processed from the external systems that share information with the master index via Java Collaborations. This process is defined by the Connectivity Maps in the Collaboration client Projects.
- 3 Data is processed from the external systems that share information with the master index via an eInsight Business Process. Using Business Processes, you can also develop eVision web pages to access data in the eIndex database. These processes are defined by the Connectivity Map in an eInsight client Project.
- 4 eIndex SPV can publish messages to a JMS Topic to broadcast to external systems. This topic is included in the eIndex SPV server Project and also in client Projects for the external systems receiving the broadcasts.

8.1.1 Connectivity Components

The connectivity components of an eIndex SPV server Project include the master index application files. Optional components include a JMS Topic or Oracle eWay (the Oracle

eWay is required when running on the Sun SeeBeyond Integration Server). The client Projects that connect to the master index use standard connectivity components of an eGate Project, with the addition of an eIndex SPV method OTD.

eIndex SPV Server Project Connectivity Components

The eIndex SPV server Project can include the following connectivity components.

- **Connectivity Map** - Graphically describes the relationship between the web application components and the eIndex application components.
- **Application file** - Contains the logic used by the master index to process data into and out of the eIndex database. This file is automatically created when you generate the eIndex SPV server Project.
- **Web application file** - Contains the logic used by the Enterprise Data Manager to process data and access the eIndex logic and database. This file is automatically created when you generate the eIndex SPV Project.
- **Oracle eWay** - Provides connectivity to the eIndex SPV database. This component is required if eIndex SPV is running on the Sun SeeBeyond Integration Server. It is optional if eIndex SPV is running on the Sun Java System Application Server, which allows you to define the database connection pool through the server.
- **JMS Topic** - A message destination conforming to the *publish-and-subscribe* (pub/sub) messaging paradigm. In this case, eIndex SPV publishes to the topic to broadcast messages to external systems. This component is optional.

Client Project Connectivity Components

The client Projects can include any of the following connectivity components.

- **Connectivity Map** - Graphically describes the relationship between the External Systems, Queues and Topics, Services, Web Connectors, and eIndex SPV application. The Connectivity Map also contains the configuration information for each component's connections—for example, the polling interval and transactional behavior.
- **eIndex application** - Represents the eIndex application accessed by the client Project. Each client Project in the eIndex SPV system must include the eIndex SPV application in its Connectivity Map in order for those components to exchange information with the master index.
- **Service** - Provides a framework for a process or a Collaboration that contains the information required to execute a set of business rules.
- **Collaboration** - Business rules describing the logic to be executed on the Object Type Definitions. These business rules include the data transformation and method calls to be executed by the Services and determine how data is processed into the eIndex database.
- **Object Type Definitions (OTDs)** - Meta-data containers that describe external objects including both data structure and methods. A custom method OTD is created in the eIndex SPV Project for use in the client Projects to define how data is

processed between the master index and external systems. A custom OTD is also created to publish messages from the master index to a JMS Topic.

- **External Applications** - Logical representations of external software applications (called *external systems*) that are integrated by the eGate system. External Applications allow the master index to connect with external systems via eGate and are linked to a Service by means of an eWay.
- **JMS Queues** - A message destination conforming to the *point-to-point* (p2p or PTP) messaging paradigm. This means that one sender delivers a message to exactly one receiver.
- **JMS Topics** - A message destination conforming to the *publish-and-subscribe* (pub/sub) messaging paradigm. This means that one publisher broadcasts messages to multiple subscribers, ensuring that all subscribers receive a message. Client Projects can include a JMS Topic to which the master index publishes, giving external systems access to all data updates.
- **JMS Client** - An internal link between a Service and a Message Destination (that is, a JMS Topic or Queue).
- **eWays** - An application-specific adapter linking an external application with eGate.
- **Business Processes** - A collection of actions and messages, revolving around a specific business practice, that flow in a specific pattern to produce an end result.
- **Web Connectors** - A graphical representation of a set of eVision web pages and activities.

Before creating any of the connectivity components, make sure you have read and understand the information presented in the *Sun SeeBeyond eGate Integrator User's Guide*. This guide gives more details about each component in an eGate Project.

8.2 Defining Connectivity Components

Connectivity Maps for the eIndex server and client Projects are predefined and illustrate basic connectivity between eIndex and external systems, and between eIndex and an eInsight Business Process. The maps do not include a JMS Topic to which eIndex publishes updates. To publish eIndex transactions to external systems, a JMS Topic must be added to the Connectivity Maps. The server Connectivity Map includes an Oracle eWay for database connectivity. You can remove the eWay if the master index is connecting to the database through the Sun Java System Application Server.

This section describes how to define connectivity components for eIndex SPV server and client Projects, including Collaborations, Business Processes, Services, Collaborations, Topics, eWays, and so on. Perform the following tasks to configure connectivity for the master index and connected systems.

- **Defining eIndex SPV Application Connectivity Components** on page 83
- **Defining the Collaboration Client Connectivity Components** on page 86

- [Customizing eInsight Client Connectivity Components](#) on page 93

Note: Refer to the *Sun SeeBeyond eGate Integrator User's Guide* for more details about performing any of the processes described in this chapter.

8.2.1 Defining eIndex SPV Application Connectivity Components

In the eIndex SPV server Project, the default Connectivity Map contains business logic and information about how data is processed in the master index. You can modify the Connectivity Map by removing the Oracle eWay (if you will use the Sun Java System Application Server for database connectivity) or by adding a JMS Topic to publish transactions processed by eIndex SPV.

This section describes how to add a JMS Topic to the Connectivity Map, which includes the following steps.

- [Configuring the Server Connectivity Map](#) on page 83
- [Adding a JMS Topic to the Server Connectivity Map](#) on page 85

Configuring the Server Connectivity Map

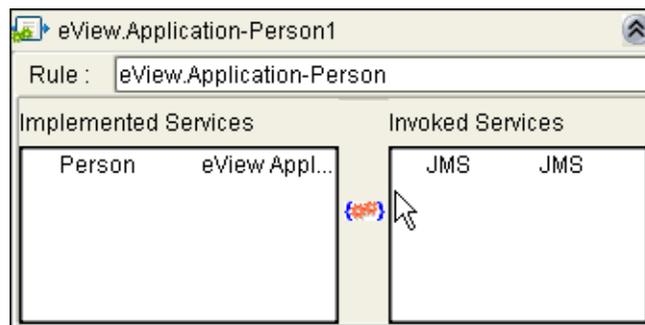
A sample server Connectivity Map is included in the eIndex SPV server Project. The map includes an Oracle eWay, which can be removed if you are running eIndex SPV on the Sun Java System Application Server and are defining the database connection pool through the server. By default the Oracle eWay is configured to connect to the database through a thin client, but you can configure it to use an OCI thick driver.

To reconnect server Connectivity Map components

If you delete the connections in the Connectivity Map or create a new Connectivity Map for the application, re-establish the connections using the Service Binding dialog.

- 1 In the eIndex server Project, check out the eIndex server Connectivity Map to configure.
- 2 Open the Connectivity Map in the Connectivity Map Editor.
- 3 Double-click **eView.Application-Person1**. The Service Binding dialog appears.

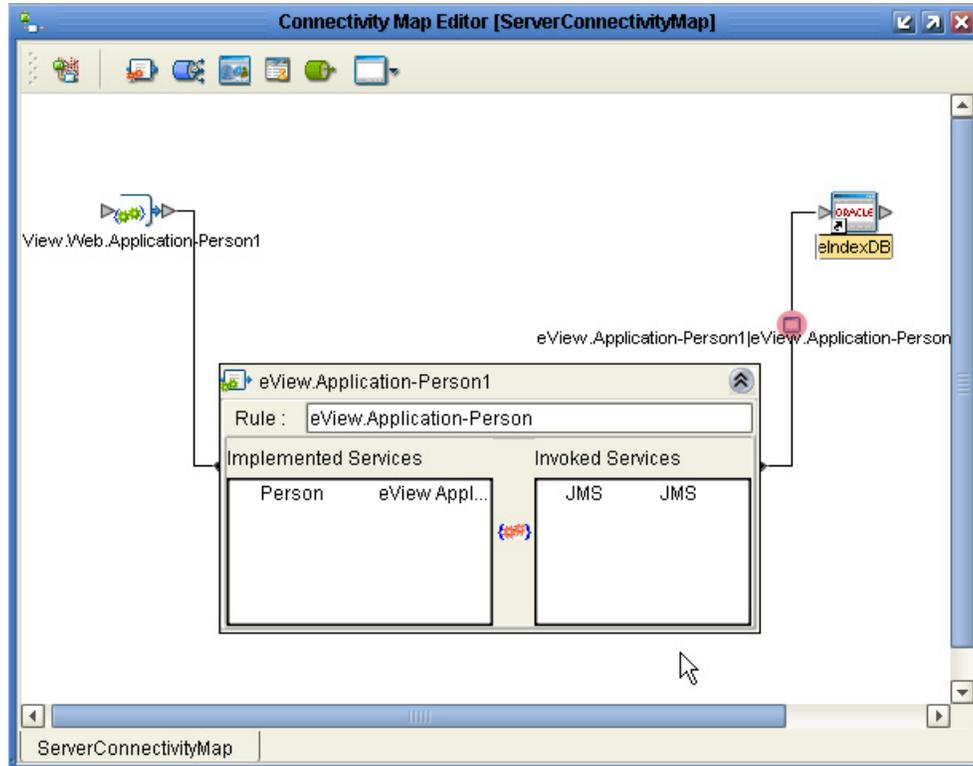
Figure 17 eIndex SPV Server Service Binding Dialog



- 4 Place the cursor over the arrow to the right of the **eView.Web.Application-Person1** icon until the cursor turns into a hand, and then drag it to **Implemented Services** in the left side of the dialog.
- 5 Place the cursor over the arrow to the right of **Invoked Services** in the right side of the dialog until the cursor turns into a hand, and then drag it to the Oracle eWay.

The Connectivity Map should look similar to Figure 18.

Figure 18 eIndex SPV Server Connectivity Map Connections



- 6 Close the Service Binding dialog.
- 7 Configure the Connectivity Map as described in the following procedure.
- 8 Save the Connectivity Map to the Repository.

To configure the server Connectivity Map

- 1 In the eIndex server Project, check out the eIndex server Connectivity Map to configure.
- 2 Open the Connectivity Map in the Connectivity Map Editor.
- 3 If eIndex SPV is connecting to the database through the Oracle eWay, configure the eWay by doing the following:
 - A Double-click the eWay icon between the eIndex SPV application and the Oracle external system.

- B On the Oracle eWay Properties window, enter new property values or accept the default values (for more information, see the *Sun SeeBeyond eWay Adapter for Oracle User's Guide*).
 - C Click **OK**.
- 4 If eIndex SPV is running on the Sun Java System Application Server **and** is connecting to the database through the server, delete the Oracle eWay by doing the following.
- A Select the Oracle external system on the Connectivity Map Editor.
 - B Press the Delete key.
 - C Click **OK** on the confirmation dialog.
 - D Delete the Oracle eWay from the Project Explorer.
- 5 Save the Connectivity Map to the Repository.

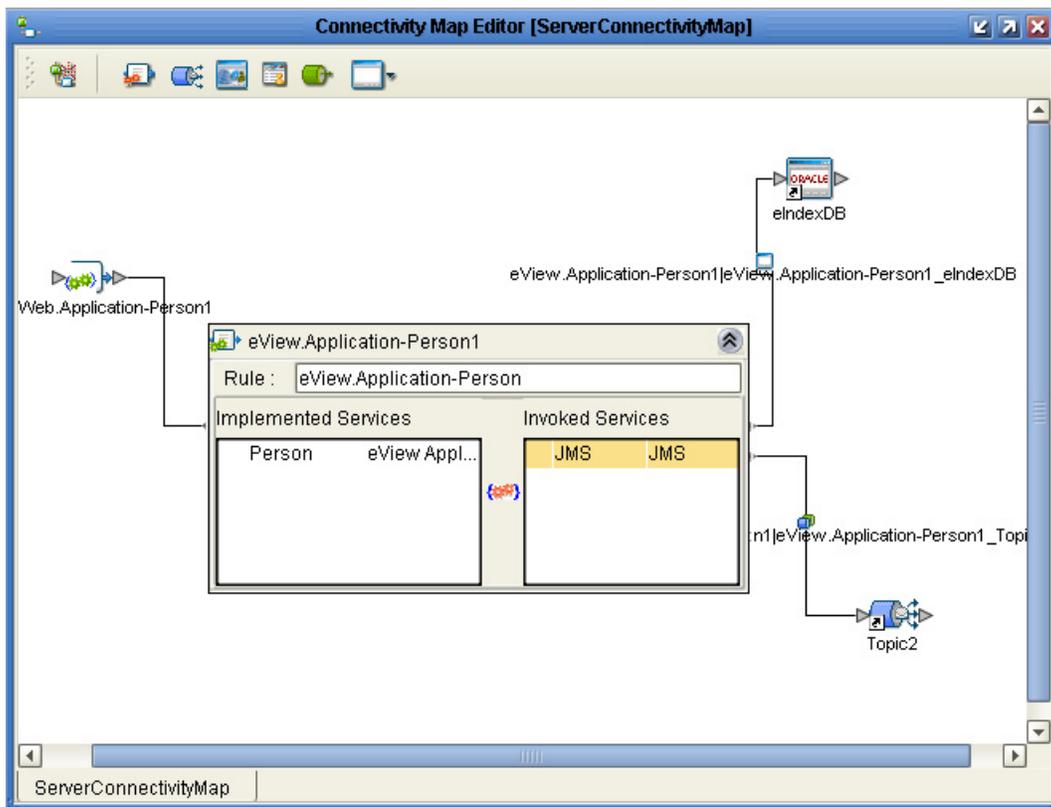
Adding a JMS Topic to the Server Connectivity Map

To use a JMS Topic to broadcast eIndex messages to external systems, you need to add the Topic to the Connectivity Map and then configure the properties. This section describes how add a JMS Topic to the eIndex SPV server Project Connectivity Map and then map the application to the new topic. You only need to perform this step if you want to publish messages processed through eIndex to external systems.

To add a JMS Topic to the eIndex Connectivity Map

- 1 In the eIndex server Project, check out the eIndex server Connectivity Map to which you want to add the new topic.
 - 2 Open the Connectivity Map in the Connectivity Map Editor.
 - 3 In the Connectivity Map Editor toolbar, select the Topic icon, and then drag it onto the canvas to the right of the **eView.Application-Person** icon.
 - 4 In the Connectivity Map Editor, double-click the **eView.Application-Person** icon. The Service Binding dialog appears (shown in [Figure 17 on page 83](#)).
 - 5 Click JMS in the Invoked Services box and drag the cursor to the Topic icon.
- The Connectivity Map should look similar to Figure 19.

Figure 19 eIndex SPV Server Connectivity Map with Topic



- 6 Close the Service Binding dialog and configure the JMS Client Connection (for more information, see the *Sun SeeBeyond eGate JMS Reference Guide*).

8.2.2 Defining the Collaboration Client Connectivity Components

In the Collaboration client Projects for external systems sharing data with the master index, the Connectivity Map contains business logic and information about how data is transferred between the master index and external systems. One default Project is provided with eIndex that includes a simple Connectivity Map illustrating an end-to-end scenario. You can create new client Projects as needed. This section describes how to modify the default map by adding a JMS Topic or by changing the Java Collaboration, which includes the following procedures.

- [Modifying the Java Collaboration](#) on page 86
- [Modifying the Collaboration Client Project Connectivity Map](#) on page 88
- [Configuring the Outbound Collaboration](#) on page 92

Modifying the Java Collaboration

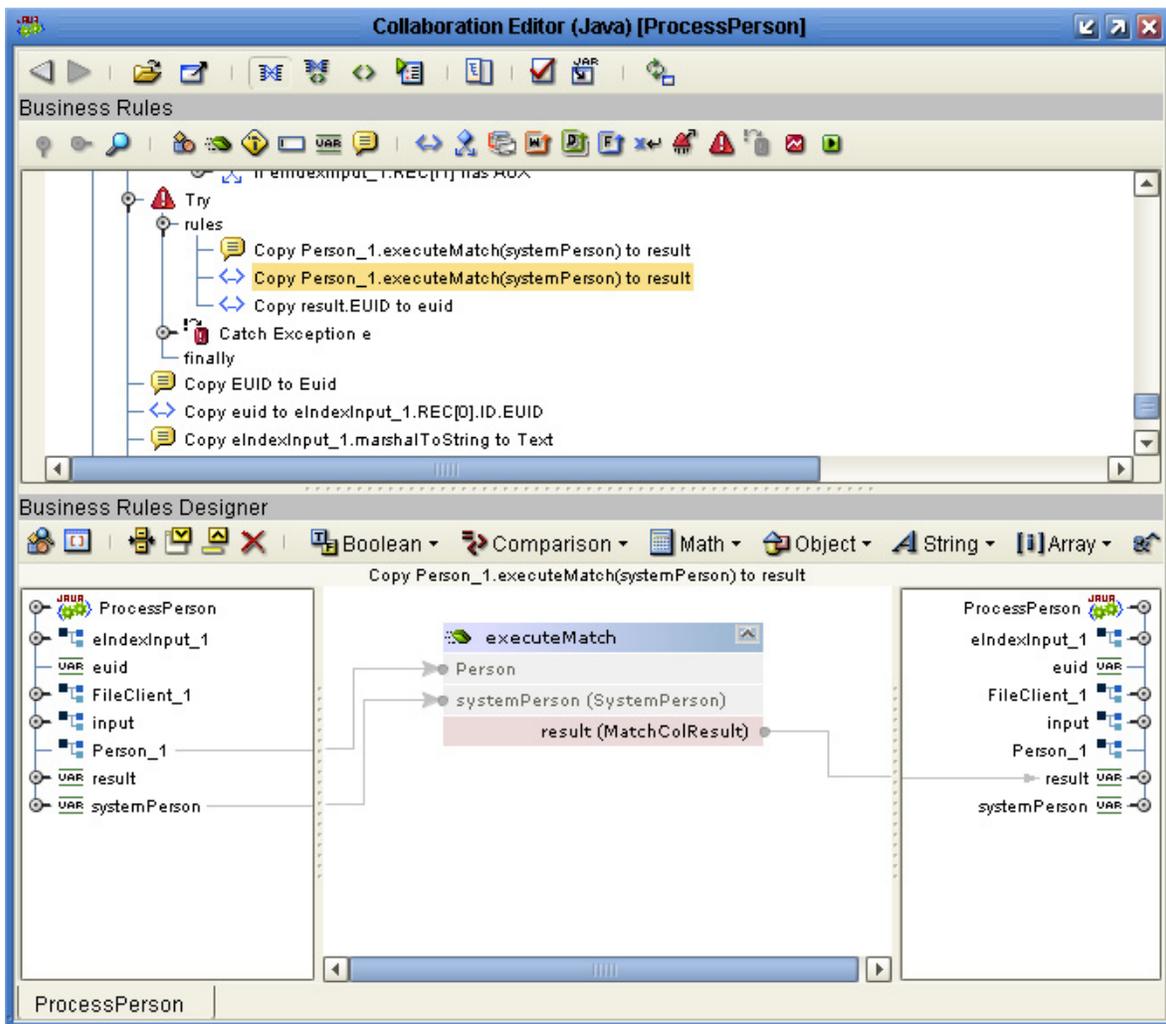
This section describes how to use the eIndex SPV method OTD in Java Collaborations for external systems. For a complete reference of the methods included in the eIndex SPV OTD, see the *Sun SeeBeyond eIndex Single Patient View Reference Guide*.

To modify the Java Collaboration

- 1 In the **eIndexClient** Project, check out the **ProcessPerson** Collaboration, and then open it in the Collaboration Editor.
- 2 To add eIndex SPV methods to the Collaboration, do the following:
 - A In the left pane of the Business Rules Designer, right-click **Person_1**. A list of available methods appears.
 - B Select the desired method from the list.
 - C Create any necessary variables for the method, and then map the input, output, and variables to the method.

Figure 20 illustrates a sample of the **executeMatch** method in the Collaboration Editor.

Figure 20 eIndex SPV Methods in Collaboration Editor



- 3 Delete or modify existing information as needed. When you are done defining the processing rules, save the Collaboration.

Modifying the Collaboration Client Project Connectivity Map

Connectivity between the eIndex SPV application and external systems is defined in the sample **eIndexClient** Project. This Project provides an end-to-end scenario using inbound and outbound file eWays to transfer data. You can use this as a sample to create custom Connectivity Maps that link external systems to eIndex SPV.

This section describes how to configure the File External Applications in the sample Connectivity Map and to incorporate a JMS Topic. You only need to incorporate the topic if you added a JMS Topic to the server Project and if you want to publish eIndex SPV messages to external systems.

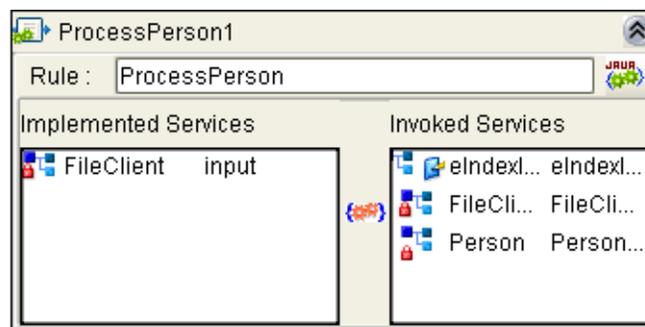
Note: The eIndex SPV application icon in the default Connectivity Map comes from the External Applications menu on the Connectivity Map Editor toolbar.

To reconnect Collaboration client connectivity components

If you delete the existing connections in the Collaboration client Connectivity Map or create a new client Connectivity Map, use the Service Binding dialog to re-establish the connections.

- 1 In the **eIndexClient** Project, check out the Collaboration client Connectivity Map to configure.
- 2 Open the client Connectivity Map in the Connectivity Map Editor.
- 3 Double-click the Service (**ProcessPerson1** in the default map). The Service Binding dialog appears.

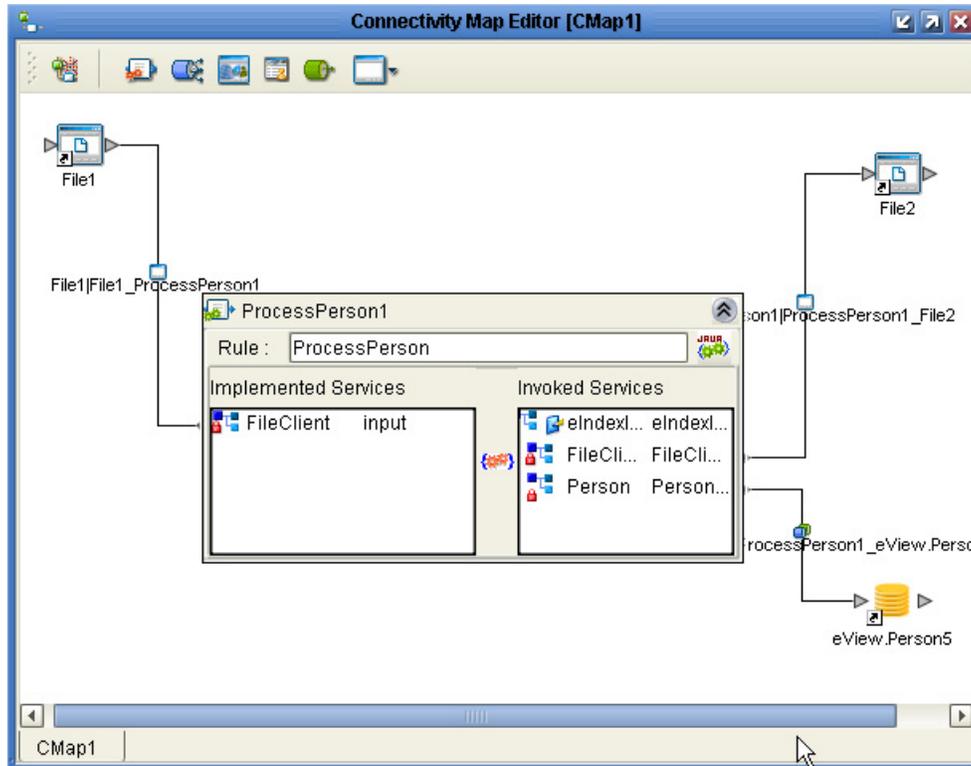
Figure 21 Collaboration Client Service Binding Dialog



- 4 Place the cursor over the arrow to the right of the input eWay icon until the cursor turns into a hand, and then drag it to the input in the Implemented Services box (**FileClient** in the default map).
- 5 Select the client in the Invoked Services box (**FileClient** in the default map), and then drag the cursor to the outbound eWay icon.
- 6 Select the eIndex SPV application in the Invoked Services box (**Person** in the default map), and then drag the cursor to the eIndex application icon.

The Connectivity Map should look similar to Figure 22.

Figure 22 Collaboration Client Connectivity Map Connections



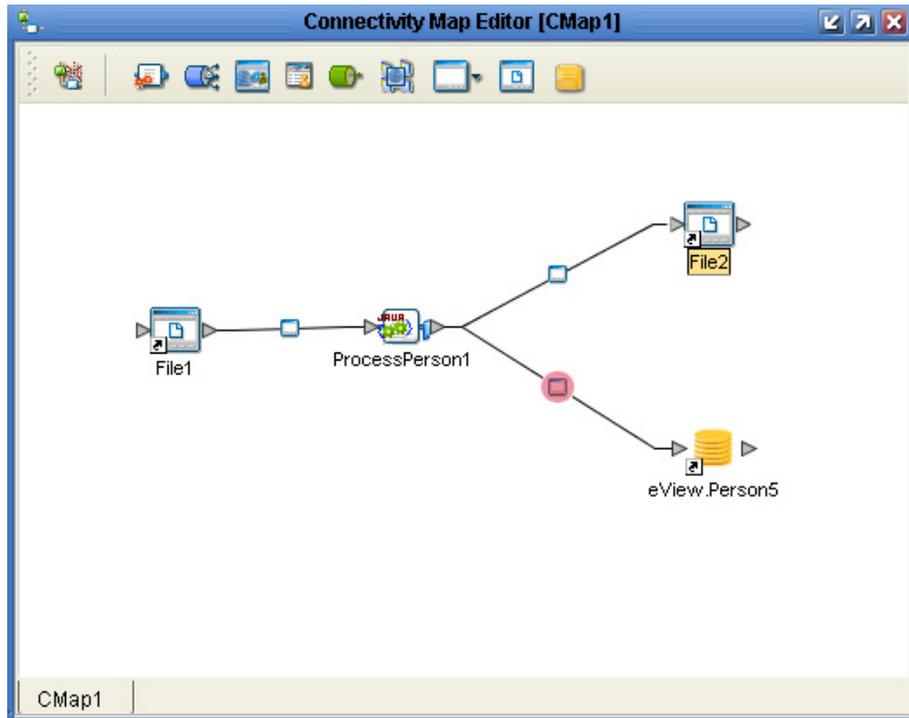
- 7 Close the Service Binding dialog.
- 8 Configure the Connectivity Map as described in the following procedure.
- 9 Save the Connectivity Map to the Repository.

To configure the Connectivity Map in the eIndexClient sample

- 1 In the **eIndexClient** Project, check out the Connectivity Map and then open it in the Connectivity Map Editor.

The map appears, as shown in Figure 23.

Figure 23 eIndexClient Connectivity Map



- 2 In the Connectivity Map, double-click the **eWay** icon to the right of **File1**.
The Properties window appears.
- 3 Configure the parameter settings for the eWay, and then click **OK** to close the Properties window.
- 4 Repeat steps 2 and 3 for the **File2** eWay.
- 5 In the Connectivity Map, double-click the icon between the **ProcessPerson1** Service and the eIndex SPV application to remove the red warning circle. No configuration is required.
- 6 Save and close the Connectivity Map.

To add the JMS Topic to the Connectivity Map

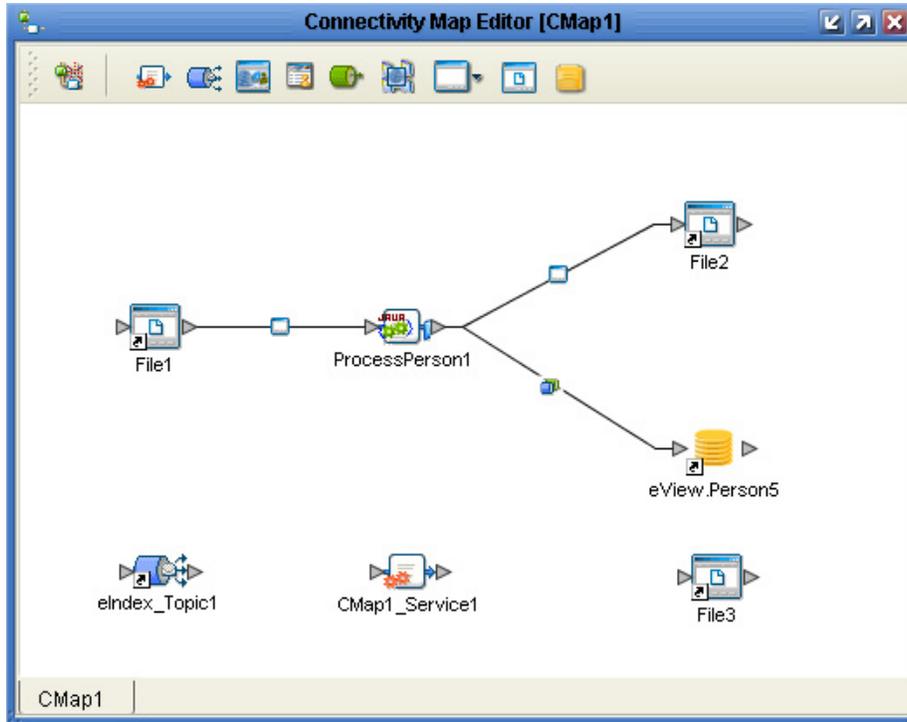
Before performing this step, make sure the server Connectivity Map contains a JMS Topic, as described in [“Adding a JMS Topic to the Server Connectivity Map”](#) on [page 85](#).

- 1 In the **eIndexClient** Project, check out the Connectivity Map and then open it in the Connectivity Map Editor.
The predefined map appears.
- 2 Drag the JMS Topic from the server Project to the Connectivity Map Editor, below the existing connectivity components (this is the topic you created in [“Adding a JMS Topic to the Server Connectivity Map”](#)).

- 3 Drag a Service from the Connectivity Map Editor toolbar to the right of the JMS Topic on the canvas.
- 4 Drag an External Application of the appropriate type from the Connectivity Map Editor toolbar to the right of the new Service on the canvas (for testing purposes, you can use a File External Application).

The Connectivity Map should now look similar to Figure 24.

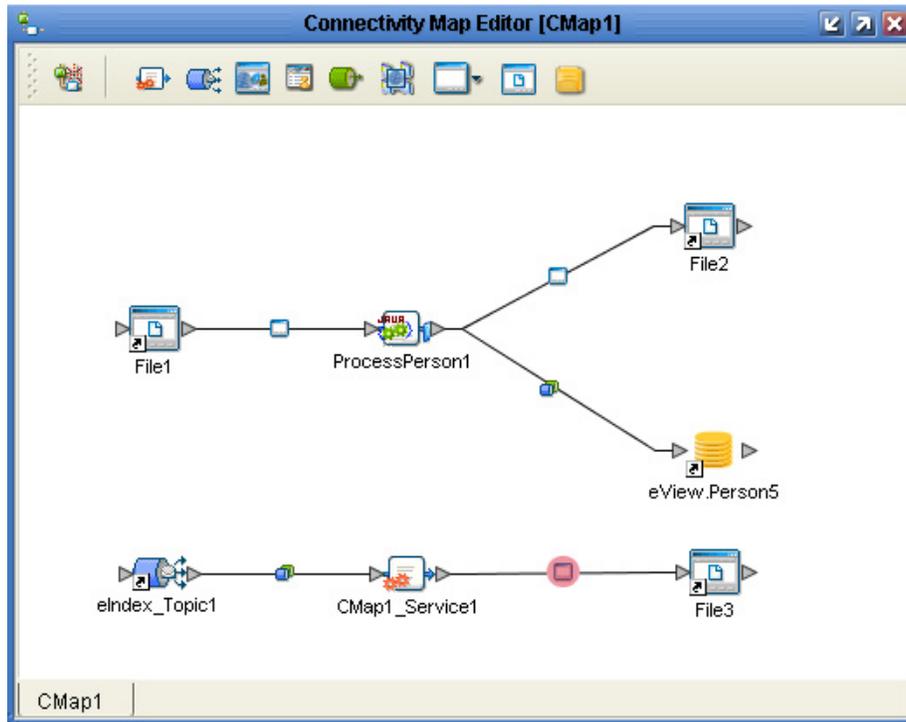
Figure 24 eIndexClient Connectivity Map with JMS Topic



- 5 In the Connectivity Map Editor, place the cursor over the arrow to the right of the **Topic** icon until the cursor turns into a hand, and then drag it into the Service to connect the two objects.
- 6 Place the cursor over the arrow to the right of the **Service** icon until the cursor turns into a hand, and then drag it into the **External Application** icon to connect the two objects.

The Connectivity Map should now look similar to Figure 25.

Figure 25 eIndexClient Connectivity Map with Connections



- 7 Double-click the JMS Client Connection icon to configure the connection (for more information, see the *Sun SeeBeyond eGate JMS Reference Guide*).
- 8 Double-click the External Application eWay to configure the location and parameter settings for the external application.
- 9 Save the Connectivity Map, and continue to “**Configuring the Outbound Collaboration**”.

Configuring the Outbound Collaboration

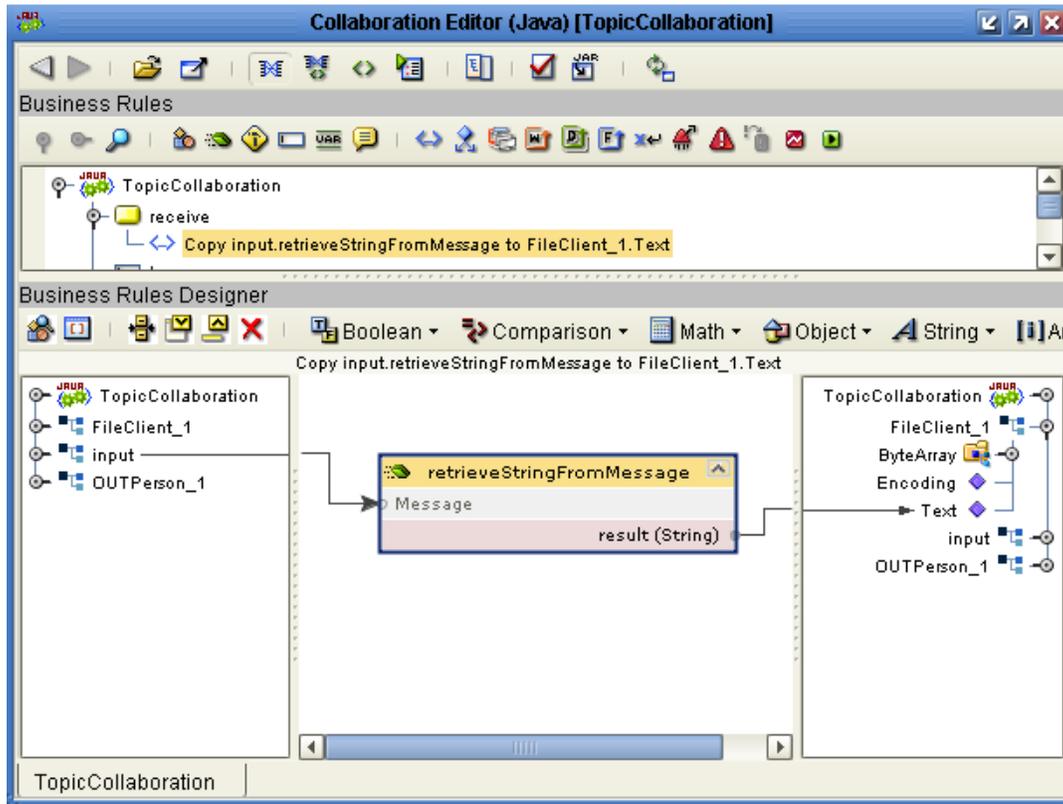
Once you create the components of a Connectivity Map for outbound message processing, you must configure the Java Collaboration that processes messages from the eIndex JMS Topic. Before you begin, make sure you have completed all of the steps in “**Modifying the Collaboration Client Project Connectivity Map**” on page 88.

To configure the outbound Collaboration

- 1 In the Project Explorer, right-click the **eIndexClient** Project.
- 2 In the **Project** context menu, select **New**, and then select **Collaboration Definition (Java)**.
- 3 Enter information into the Collaboration Definition Wizard, with the following guidelines:
 - ♦ For the **Web Service Type**, select the existing JMS receive type (navigate to **Sun SeeBeyond\eGate\JMS** and select **receive**).

- ♦ Select the appropriate outbound OTD for the external system (for testing with a File External Application, select the **FileClient** OTD).
- 4 Configure the Collaboration to map data from the JMS Topic to the external system (a sample is shown in Figure 26).

Figure 26 Outbound Java Collaboration



- 5 Save the Collaboration to the Repository.
- 6 Open the **eIndexClient** Connectivity Map, and drag the newly created Collaboration onto the Service connected to the JMS Topic.
- 7 Save the Connectivity Map to the Repository.

8.2.3 Customizing eInsight Client Connectivity Components

If eInsight was installed when you installed eIndex, a sample Project was created that implements eIndex SPV methods in an eInsight Business Process. In this Project (the eInsight client Project), the Connectivity Map contains business logic and information about how data is transferred between the master index and external systems through an eInsight Business Process. The Business Process defines how data is transformed before being sent to the master index. This section describes how to use eIndex SPV methods in the sample Business Process and to customize the Connectivity Map. Perform the following tasks to customize eInsight client connectivity components.

- **Modifying the Business Process** on page 94

- **Modifying the eInsight Client Connectivity Map** on page 95

Note: Refer to the *Sun SeeBeyond eInsight Business Process Manager User's Guide* for more details about performing any of the processes described in this section.

Modifying the Business Process

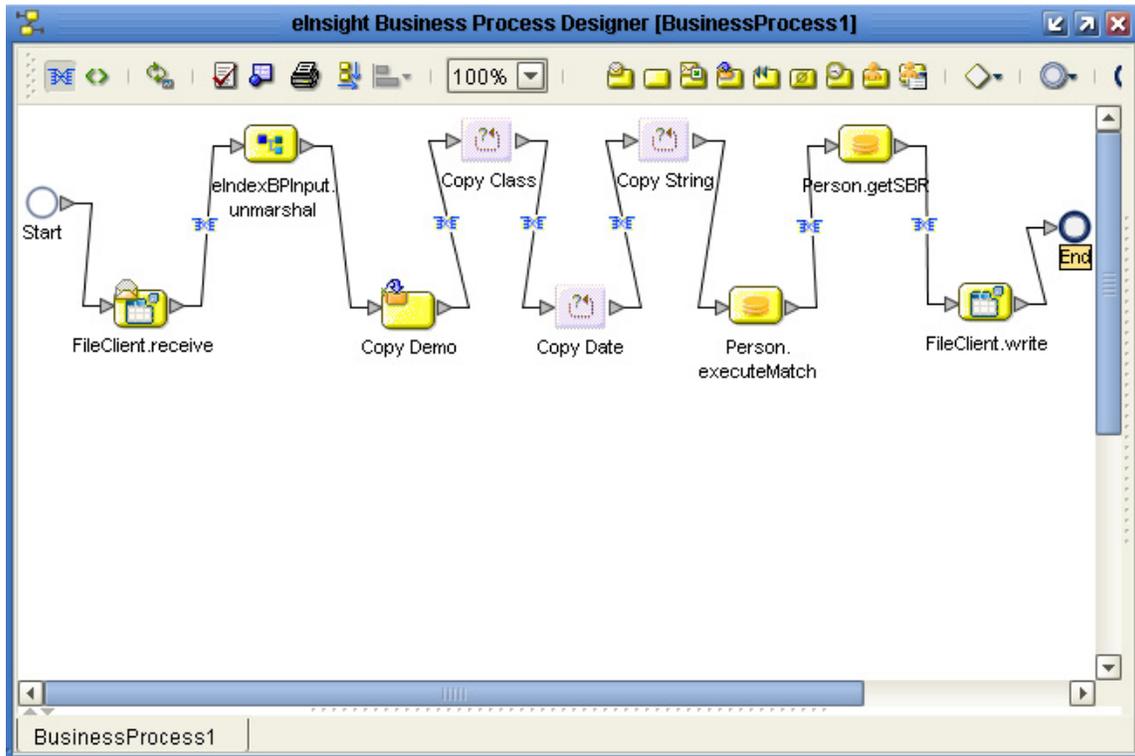
The sample eInsight client Project (**eIndexBPClient**) defines processing in the same default manner as the predefined Java Collaboration. You can process data through only the Collaboration, only the eInsight Business Process, or both. If you use both methods and you modify the processing logic for the Collaboration, be sure to modify the Business Process logic accordingly.

Before using the eInsight client Project, you must customize the Business Process. This section provides instructions for customizing host information and processing logic in the Business Process. For more information about the available eIndex methods, see the *Sun SeeBeyond eIndex Single Patient View Reference Guide*.

To modify the Business Process

- 1 In the **eIndexBPClient** Project, check out the **BusinessProcess1** Business Process, right-click **BusinessProcess1**, and then select **Properties** from the context menu.
- 2 On the **General** tab of the Business Process Properties dialog, modify the **Target Namespace** field as follows:
 - A Replace the computer name and port number with the server name and port number of your repository.
 - B Replace the repository name with the name of your repository.
- 3 On the Business Process Properties dialog, click **OK**.
- 4 To add eIndex SPV methods to the Business Process, do the following:
 - A In the **eIndexBPClient** Project, check out the **BusinessProcess1** Business Process, and then open it in the eInsight Business Process Designer.
 - B In the Project Explorer, expand the **eIndex** Project, and then expand the **Person** node. The Business Process method list appears.
 - C Select the desired method from the list and drag the method to the Business Process Designer between the appropriate Business Process components.
 - D Delete the existing link between the components on either side of the new method, and then link the new method to the activities on either side of it.
 - E For each link you create, right-click the link and select **Add Business Rule**. Configure the business rule to map data from the input to output activity. (For more information, see the *Sun SeeBeyond eInsight Business Process Manager User's Guide*.)
- 5 Delete or modify existing information in the Business Process as needed. Figure 20 illustrates a sample of adding the **getSBR** method in the Business Process Designer.

Figure 27 eIndex SPV Methods in a Business Process



- 6 When you are done defining the processing rules, save and check in the Business Process.

Modifying the eInsight Client Connectivity Map

Connectivity between the eIndex SPV application and eInsight is defined in the Connectivity Map of the sample **eIndexBPClient** Project. This Project provides an end-to-end scenario using inbound and outbound file eWays to transfer data. You can use this as a sample to create custom Connectivity Maps that link external systems to eIndex SPV through a Business Process.

This section describes how to configure the File External Applications in the sample Connectivity Map. You can also incorporate a JMS Topic into the Connectivity Map in the **eIndexBPClient** Project (for information and instructions, see [“To add the JMS Topic to the Connectivity Map” on page 90](#)). You only need to incorporate the topic if you added a JMS Topic to the server Project and you want to publish master index updates to external systems.

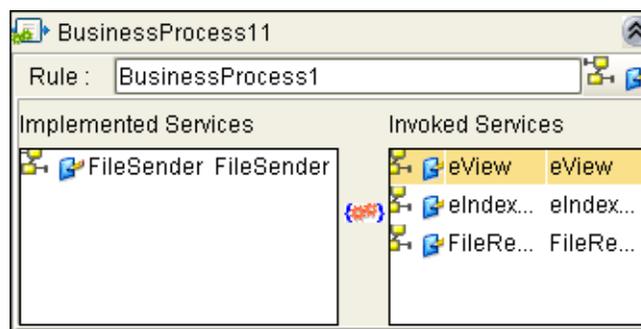
Note: The eIndex SPV application icon in this Connectivity Map comes from the External Applications menu on the Connectivity Map Editor toolbar.

To reconnect eInsight client connectivity components

If you delete the existing connections in the eInsight client Connectivity Map or create a new client Connectivity Map, use the Service Binding dialog to re-establish the connections.

- 1 In the **eIndexBPClient** Project, check out the eInsight client Connectivity Map to configure.
- 2 Open the client Connectivity Map in the Connectivity Map Editor.
- 3 Double-click the Service (**BusinessProcess1** in the default map). The Service Binding dialog appears.

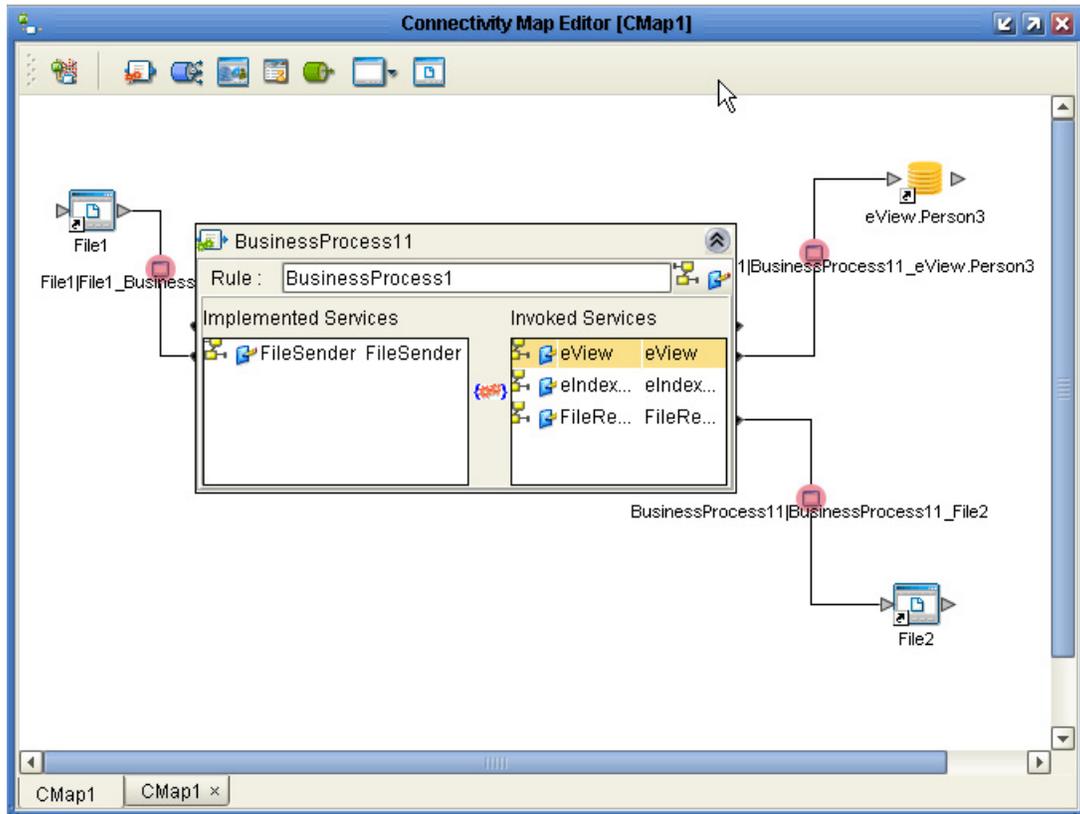
Figure 28 eInsight Client Service Binding Dialog



- 4 Place the cursor over the arrow to the right of the input eWay icon until the cursor turns into a hand, and then drag it to the appropriate item in the Implemented Services box (**FileSender** in the default map).
- 5 Select the output service in the Invoked Services box (**FileReceiver** in the default map), and then drag the cursor to the outbound eWay icon.
- 6 Select **eView** in the Invoked Services box, and then drag the cursor to the eIndex application icon.

The Connectivity Map should look similar to Figure 29.

Figure 29 eInsight Client Connectivity Map Connections



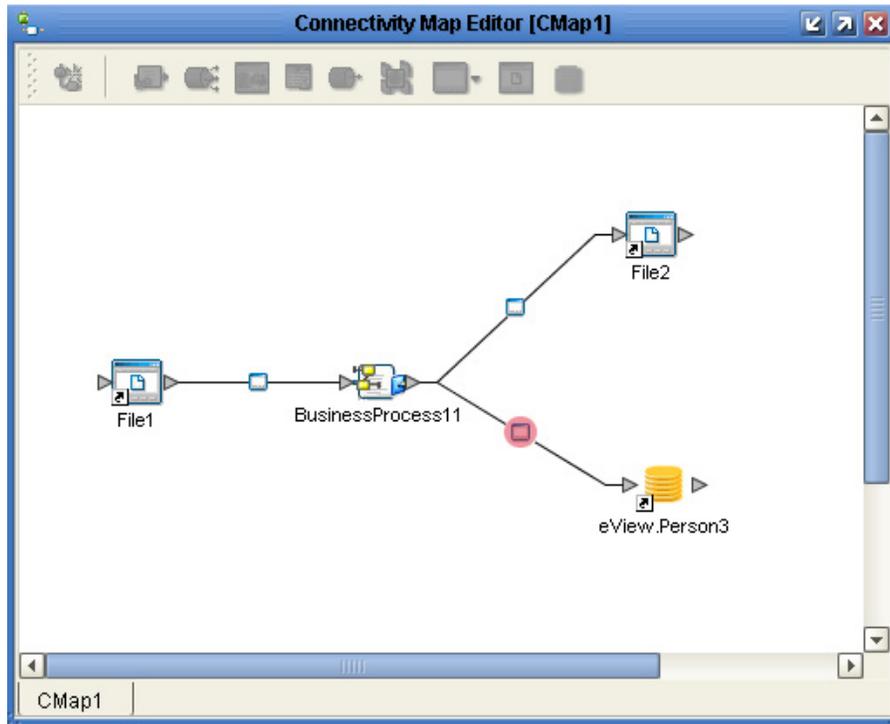
- 7 Close the Service Binding dialog.
- 8 Configure the Connectivity Map as described in the following procedure.
- 9 Save the Connectivity Map to the Repository.

To configure the eIndexBPClient sample Project

- 1 In the **eIndexBPClient** Project, check out the Connectivity Map and then open it in the Connectivity Map Editor.

The map appears, as shown in Figure 30.

Figure 30 eIndexClient Connectivity Map



- 2 In the Connectivity Map, double-click the **eWay** icon to the right the inbound eWay icon.
The Properties window appears.
- 3 Configure the parameter settings for the eWay.
- 4 Repeat steps 2 and 3 for outbound file eWay.
- 5 Double-click the icon between the BusinessProcess11 Service and the eIndex SPV application to remove the red warning circle. No configuration is required.
- 6 Save and close the Connectivity Map.

Deploying the Project

Each Project in the master index system must include a Deployment Profile that correlates the processing components to the physical components. This includes the primary eIndex SPV Project and any client Projects that connect the master index to an external system.

What's in This Chapter

- [Deployment Overview](#) on page 99
- [Creating a Domain](#) on page 99
- [Defining Deployment Profiles](#) on page 102
- [Defining Security](#) on page 114

9.1 Deployment Overview

The Deployment Profile binds the eIndex SPV Project attributes to the Environment that defines where each component runs. For example, the Deployment Profile for the server Project defines which application or integration server runs the master index. The Deployment Profiles for the client Projects that use eIndex SPV Components define which message servers host which topics, which external systems are connected to the master index via which eWays, and so on. Once you create the Deployment Profile and build the Project, you can deploy the Project to the application or integration server. When running the Sun SeeBeyond Integration Server (IS), you must create the Logical Host instance (domain) before you can deploy the Project. For applications running on the Sun Java System Application Server, you must create a domain through the Sun Java System Application Server Admin Console (see your Sun documentation for more information).

9.2 Creating a Domain

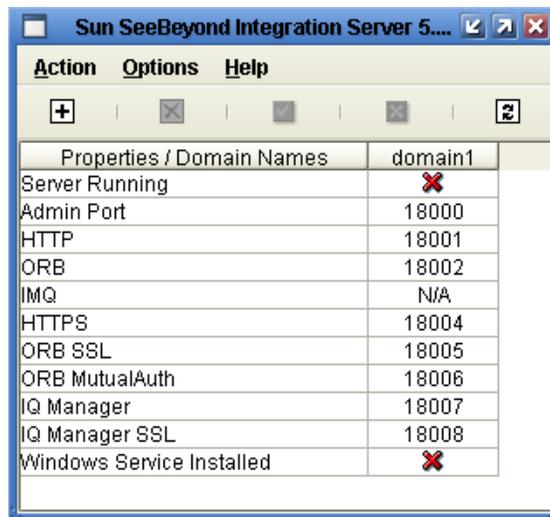
If eIndex SPV is running on the IS, make sure you have created and started an instance of the Logical Host containing the IS before defining Deployment Profiles for your Projects. This section describes how to create a domain using the Domain Manager. You can also create the instance using a command-line tool. The command-line method is described in the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Important: The instructions below only pertain to applications running on the Sun SeeBeyond Integration Server. For applications running on the Sun Java System Application Server, refer to your Sun documentation for information about creating and starting a domain. Before starting the Logical Host, make sure that the eIndex SPV databases is running.

To create a domain using the Domain Manager (IS implementations only)

- 1 In the <jis51>\logicalhost directory, run the **domainmgr.bat** script.
- 2 If there are currently no domains, a dialog box indicates that you can create a domain now. If you click **Yes**, the **Create Domain** dialog box appears. Go to step 4. The Domain Manager appears.

Figure 31 Domain Manager



- 3 On the Domain Manager toolbar, click the **Create a New Domain** tool. The **Create Domain** dialog box appears (see Figure 32).

Figure 32 Create Domain Dialog Box

4 You can change the default values of any of the fields listed in Table 4.

Note: To let the Domain Manager choose the port numbers for you, click **AutoPick Port**.

Table 4 Fields in Create Domain Dialog Box

Field	Description
Domain Name	A unique name for the domain.
Admin User Name	A name for the user who will administer the domain.
Admin User Password	A password for the administrator. The value that you enter is hidden with asterisks. The default value is STC .
Re-Type Admin User Password	Retype the password.
Admin Port	The port number used by the domain's administrative server.
HTTP	The port number used by the domain's HTTP listener.
IMQ	This port number is not currently used.
HTTPS	The port number used by the domain's HTTP listener for SSL requests.

Table 4 Fields in Create Domain Dialog Box

Field	Description
IQ Manager	The port number used by the domain's SeeBeyond JMS IQ Manager.
IQ Manager SSL	The port number used by the domain's SeeBeyond JMS IQ Manager for SSL requests.
ORB	The port number used by the domain's IIOP listener.
ORB SSL	The port number used by the domain's IIOP listener for SSL requests.
ORB MutualAuth	The port number used by the domain's IIOP listener for mutual authentication requests, in which the client and server authenticate each other.

- 5 If you want to install the SeeBeyond Integration Server as a Windows service, select the **Install Runtime as Windows Service** check box. The service name will be **IS 5.1 <domain_name>**.
- 6 Click **Create**.
- 7 When the **Message** dialog box indicates that the domain has been successfully created, click **OK**.
- 8 (Make sure the eIndex SPV database is running before performing this step.) On the Domain Manager window, select the new domain and then click **Start an Existing Domain**.
- 9 When the **Message** dialog box indicates that the domain has been successfully started, click **OK**.

9.3 Defining Deployment Profiles

A Deployment Profile must be defined for the eIndex SPV application Project and for any client Projects that connect to the master index. You can use the same Environment components for each Deployment Profile.

Once Project components are mapped in the Deployment Profile, you can build and deploy the Project. You must deploy the Project before you can run any of the components. You can deploy a Project from either Enterprise Designer or Enterprise Manager. These instructions describe deploying from Enterprise Manager. For more information and instructions on deploying a Project using Enterprise Designer, see the *Sun SeeBeyond eGate Integrator User's Guide*. For information about deploying a Project using Enterprise Manager, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Note: When deploying from Enterprise Manager or Sun Java System Application Server Admin Console, you must specify the .ear file to deploy. This file is located at **<edesigner>/builds/<project><deployment_profile>/<integration_server>** (where <edesigner> is the Enterprise Designer home

directory, <project> is the name of the eIndex SPV Project, <deployment_profile> is the name of the Project's Deployment Profile, and <integration_server> is the name of the server on which the application is deployed). The file is named <project><deployment_profile>.ear; for example, eIndexPersonIndexDeploy.ear).

9.3.1 Deploying the eIndex SPV Server Project

Creating a Deployment Profile for the eIndex SPV server Project (which defines the eIndex SPV application) consists of the following steps.

- [Creating the Deployment Profile](#) on page 103
- [Mapping Project Components to Environment Components](#) on page 105
- [Building and Deploying the Server Project](#) on page 107

Important: *Make sure you have created and started a domain before attempting to deploy the Project.*

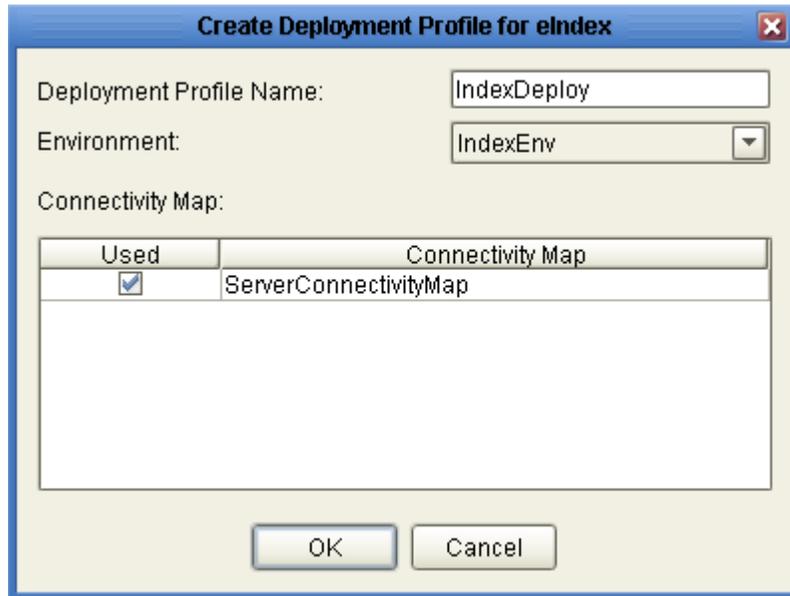
Creating the Deployment Profile

The first step to deploying a Project is to create and name the Deployment Profile. Perform the following steps to create a new Deployment Profile for the eIndex SPV server Project.

To create an eIndex SPV application Deployment Profile

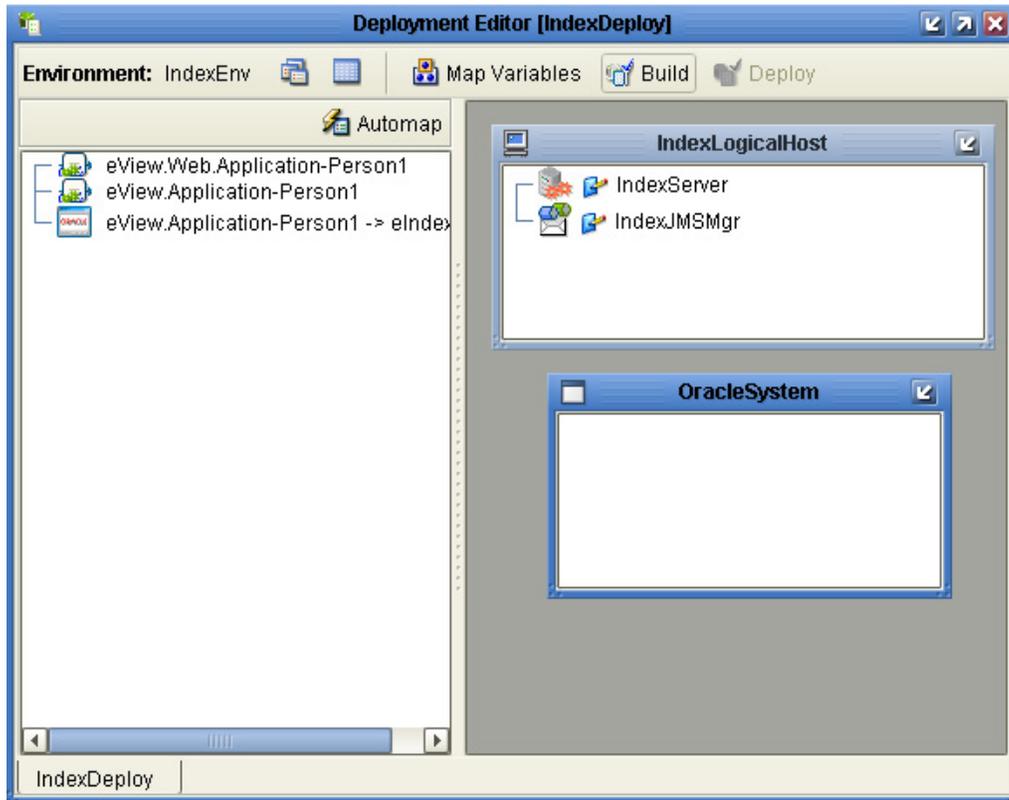
- 1 In Enterprise Explorer, click the **Project Explorer** tab.
- 2 Select the eIndex SPV server Project folder.
- 3 Right-click the mouse to launch the **Project** context menu.
- 4 Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 33.

Figure 33 Create Deployment Profile Dialog Box



- 5 In the **Deployment Profile Name** field, type a name for the Deployment Profile.
- 6 In the **Environment** drop-down list, select the Environment you created for the eIndex SPV Project.
- 7 In the Connectivity Map table, deselect the Used check box for any Connectivity Maps you do not want to include in the deployment.
- 8 Click **OK** to add a **Deployment Profile** icon to the eIndex SPV Project and display the Deployment Editor window, shown in Figure 34.

Figure 34 Deployment Editor - Server Project



9 Continue to “Mapping Project Components to Environment Components”.

Mapping Project Components to Environment Components

Once you create a Deployment Profile, you can map the Project components to the deployment Environment. When you map the Project components to the Environment containers, you are specifying the Logical Host to handle the transactions.

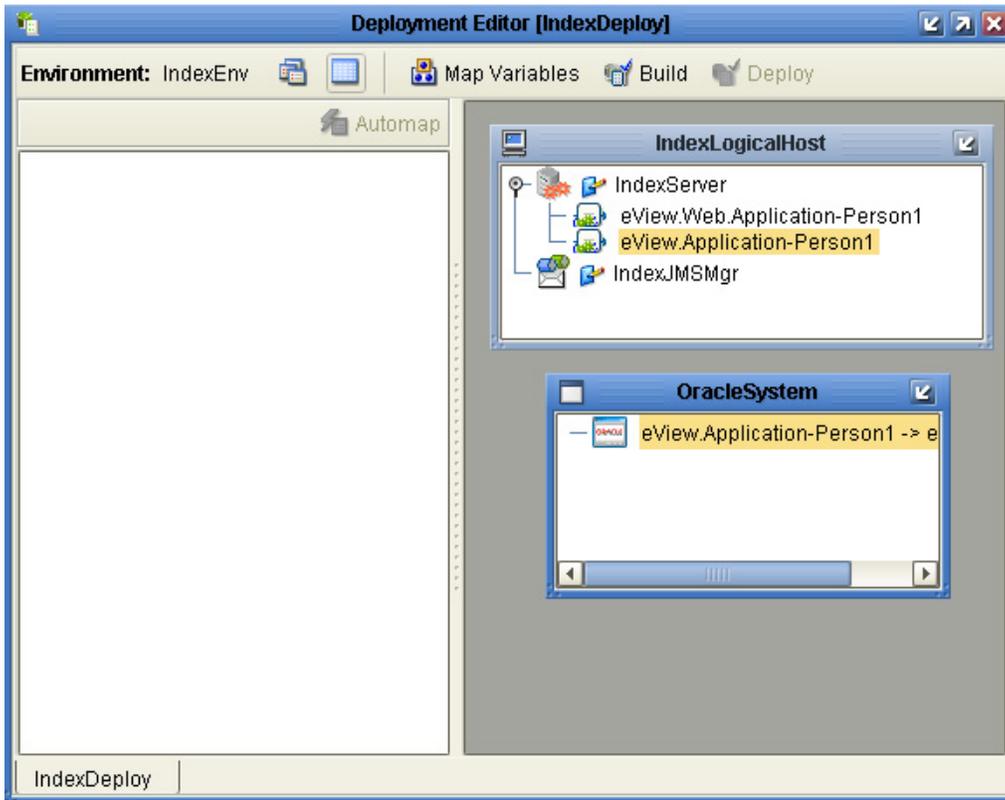
You can map Project components manually, or you can use the Automap feature to have the components automatically mapped for you. Use the Automap feature when a one-to-one correspondence exists between the available Project components and the containers in the Environment to which you want to deploy them.

To map eIndex SPV Project components manually

- 1 With the eIndex SPV Project Deployment Profile open in the Deployment Editor, drag the **eView.Web.Application-Person** and **eView.Application-Person** icons onto the integration or application server in the Deployment Editor (the server appears in the Logical Host).
- 2 If you defined a JMS Topic to publish the master index messages, drag the **JMS Topic** icon onto the JMS IQ Manager in the Logical Host.
- 3 If the server Project contains an Oracle eWay, drag the **Oracle External Application** icon onto the Oracle External System in the right pane.

Figure 35 illustrates the updated Deployment Profile.

Figure 35 Mapped eIndex SPV Components in the Deployment Editor

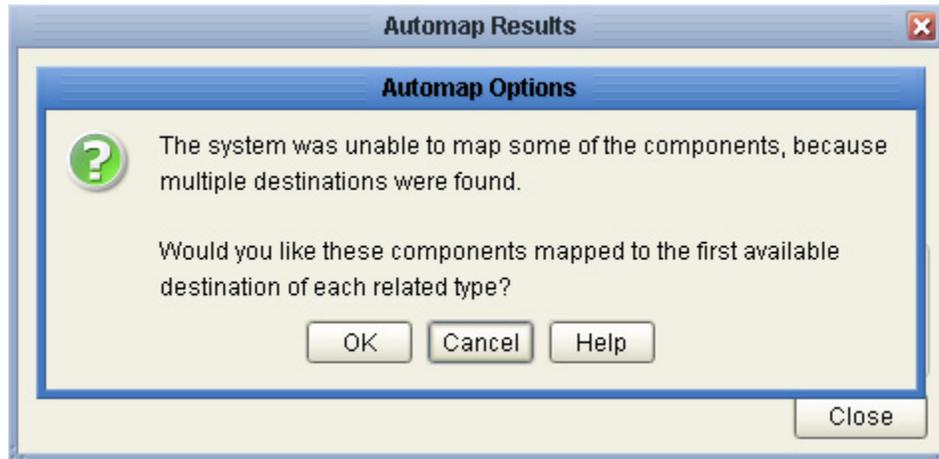


4 Continue to “**Building and Deploying the Server Project**”.

To automatically map eIndex SPV Project Components

- 1 With the eIndex SPV Project Deployment Profile open in the Deployment Editor, click **Automap**.
The Automap Results dialog appears.
- 2 Review the results of the mapping on the Automap Results dialog. If the appropriate container for the component cannot be found, the component remains unmapped.
- 3 If more than one container was found for a component, click **Automap Option** to review mapping options and do one of the following:
 - ◆ To accept the Automapping option, click **OK**.
 - ◆ To map the components manually, click **Cancel**.

Figure 36 Automap Options Dialog



- 4 To map any components that could not be automatically mapped, follow the instructions under [To map eIndex SPV Project components manually](#) on page 105.
- 5 Continue to [“Building and Deploying the Server Project”](#).

Building and Deploying the Server Project

Once all Project components are mapped to the Environment, build the Project to create and compile all the files needed to run the Project. Building the Project places the application file at `<edesigner>\builds\<project_name><deploy_profile_name>\<logicalhost_name>\<server_name>`. Deploying the Project extracts the files to the domain.

To build and deploy the server Project

- 1 Click **Build** to generate the Project files.
- 2 After the build process finishes successfully, click **Deploy** to deploy the Project to a server.

Note: *For this to be successful, the domain must be started.*

- 3 Define EDM users, as described under [“Defining Security”](#) on page 114.

9.3.2 Deploying the Collaboration Client Project

Creating a Deployment Profile for the Projects that define external system connections to the eIndex SPV application via Collaborations consists of the following steps.

- [Creating a Collaboration Client Deployment Profile](#) on page 108
- [Mapping Collaboration Client Project Components](#) on page 109
- [Building and Deploying the Collaboration Client Project](#) on page 110

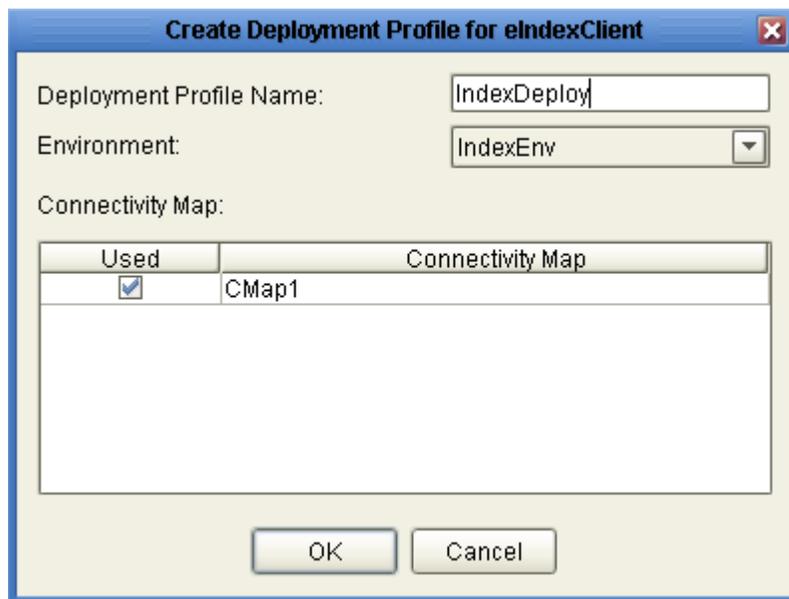
Creating a Collaboration Client Deployment Profile

Before beginning this procedure, you must have created, built, and deployed the eIndex SPV application Deployment Profile, as described in [“Creating the Deployment Profile” on page 103](#). Doing this creates an eIndex SPV application component in the Environment, which is required for the client deployment.

To create a Collaboration client Deployment Profile

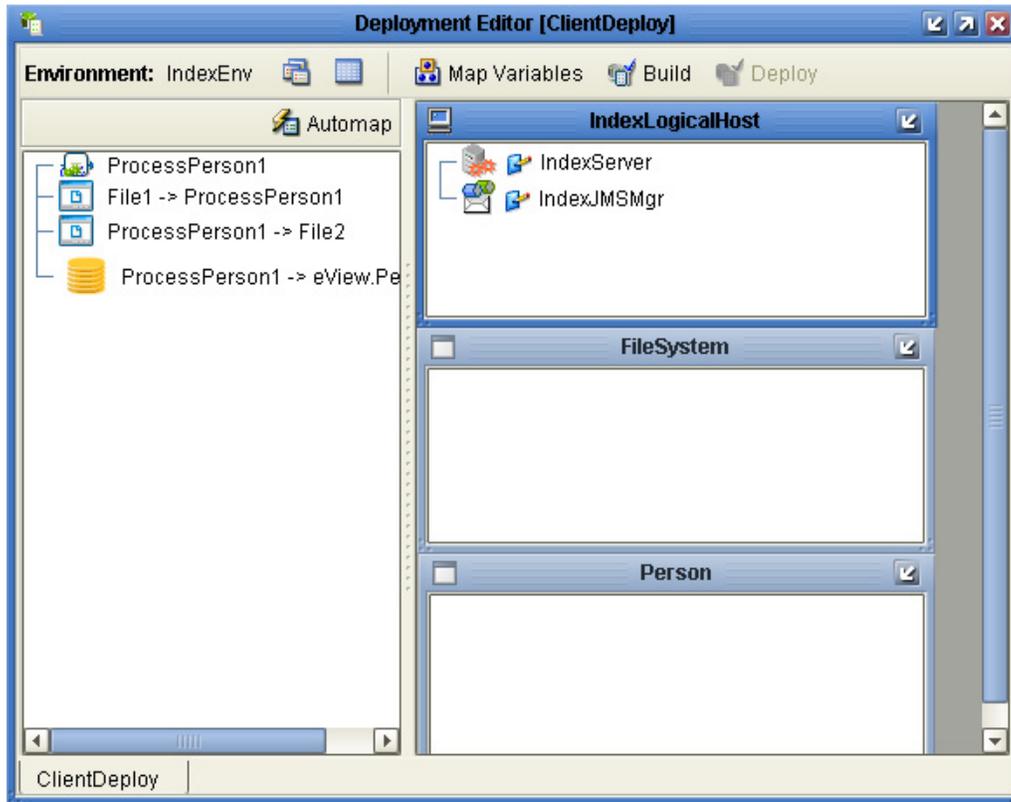
- 1 In Enterprise Explorer, click the **Project Explorer** tab.
- 2 Select the Collaboration client Project folder.
- 3 Right-click the mouse to launch the **Project** context menu.
- 4 Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 37.

Figure 37 Create Deployment Profile Dialog Box



- 5 In the **Deployment Profile Name** field, type a name for the Deployment Profile.
- 6 In the **Environment** drop-down list, select the Environment you created for the eIndex SPV Project.
- 7 Click **OK** to add a **Deployment Profile** icon to the eIndex SPV Project and display the Deployment Editor window, shown in Figure 38.

Figure 38 Deployment Editor - Client Project



Note: Your Deployment Editor might differ from the above illustration depending on whether you implemented a JMS Topic for processing outbound messages.

8 Continue to “Mapping Collaboration Client Project Components”.

Mapping Collaboration Client Project Components

Once you create a Deployment Profile, you must configure the Project components in the deployment Environment. When you map the Project components to the Deployment Profile, you are specifying the Logical Host to handle the transactions.

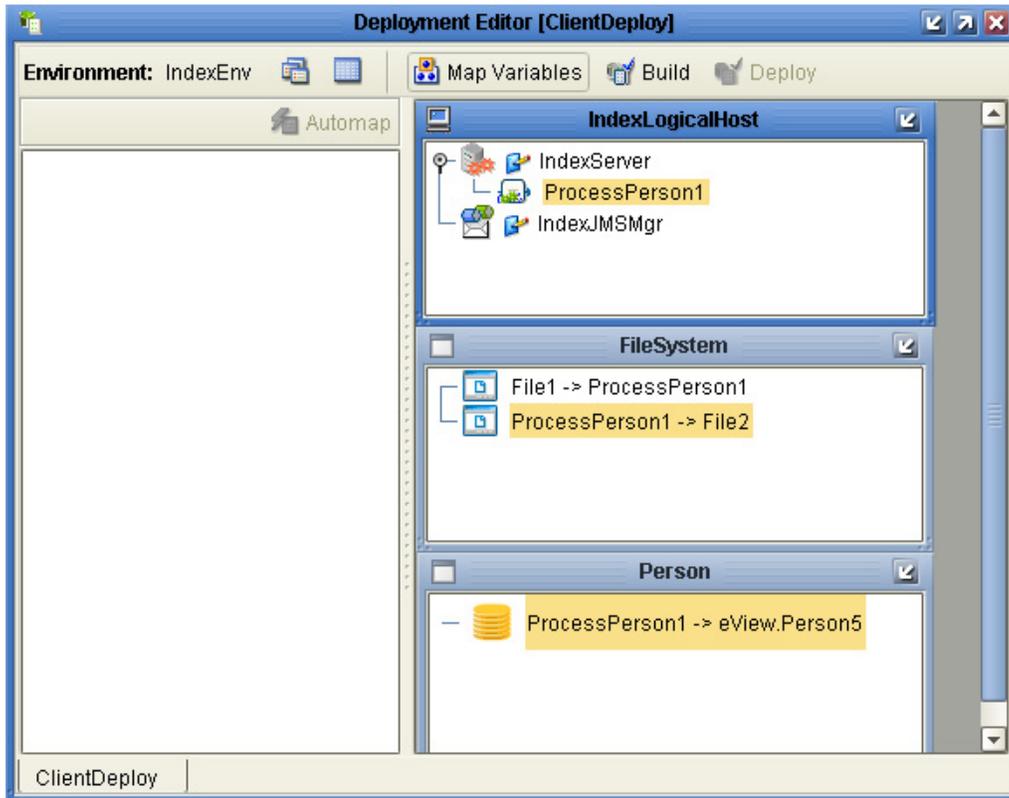
Note: Project components can also be automatically mapped. For more information, see [Mapping Project Components to Environment Components](#) on page 105 and [To automatically map eIndex SPV Project Components](#) on page 106.

To map Collaboration client Project components

- 1 With the Collaboration client Project Deployment Profile open in the Deployment Editor, drag the Service icon(s) onto the integration or application server in the Deployment Editor (the server appears in the Logical Host).
- 2 Drag the eIndex SPV application icon onto the eIndex SPV application deployment component (this is named after the eIndex SPV application).

- 3 Drag the external system eWays to the appropriate external systems in the Environment.
- 4 If you implemented a JMS Topic, drag the topic to the JMS IQ Manager in the Logical Host.

Figure 39 Mapped Client Components in the Deployment Editor



- 5 Continue to “Building and Deploying the Collaboration Client Project”.

Building and Deploying the Collaboration Client Project

Once all Project components are mapped to the Environment, you can build the Project to create and compile all the files needed to run the Project. Deploying the Project places the files on the server.

Note: For this to be successful, the domain must be started.

To build and deploy the Collaboration client Project

- 1 Click **Build** to generate the Project files.
- 2 Click **Deploy** to deploy the Project to a server.

9.3.3 Deploying the eInsight Client Project

Creating a Deployment Profile for the Projects that incorporate eIndex SPV methods into an eInsight Business Process consists of the following steps

- [Creating an eInsight Client Deployment Profile](#) on page 111
- [Mapping eInsight Client Project Components](#) on page 112
- [Building and Deploying the eInsight Client Project](#) on page 113

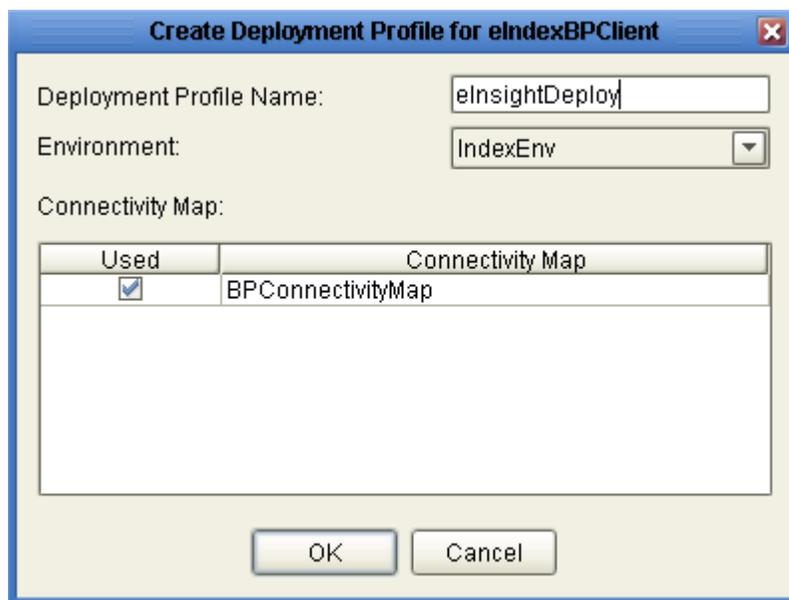
Creating an eInsight Client Deployment Profile

Before beginning this procedure, you must have created, built, and deployed the eIndex SPV application deployment profile, as described in [“Creating the Deployment Profile” on page 103](#). Doing this creates the eIndex SPV application component in the Environment, which is required for the client deployment.

To create an eInsight client Project Deployment Profile

- 1 In Enterprise Explorer, click the **Project Explorer** tab.
- 2 Select the eInsight client Project folder.
- 3 Right-click the mouse to launch the **Project** context menu.
- 4 Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 40.

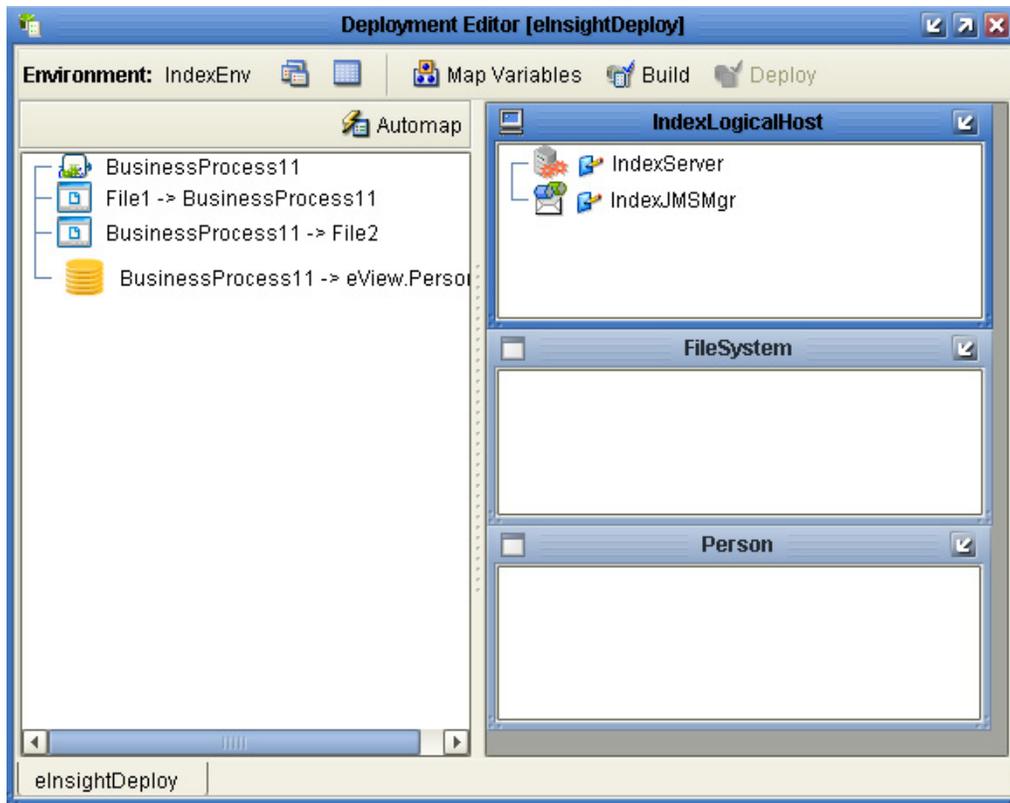
Figure 40 Create Deployment Profile Dialog Box



- 5 In the **Deployment Profile Name** field, type a name for the Deployment Profile.
- 6 In the **Environment** drop-down list, select the Environment you created for the eIndex SPV Project.

- 7 Click **OK** to add a **Deployment Profile** icon to the eIndex SPV Project and display the Deployment Editor window, shown in Figure 41.

Figure 41 Deployment Editor Window



- 8 Continue to “Mapping eInsight Client Project Components”.

Mapping eInsight Client Project Components

Once you create a Deployment Profile, you can map the Project components to the deployment Environment. When you map the Project components to the Deployment Profile, you are specifying the Logical Host to handle the transactions.

Note: Project components can also be automatically mapped. For more information, see [Mapping Project Components to Environment Components](#) on page 105 and [To automatically map eIndex SPV Project Components](#) on page 106.

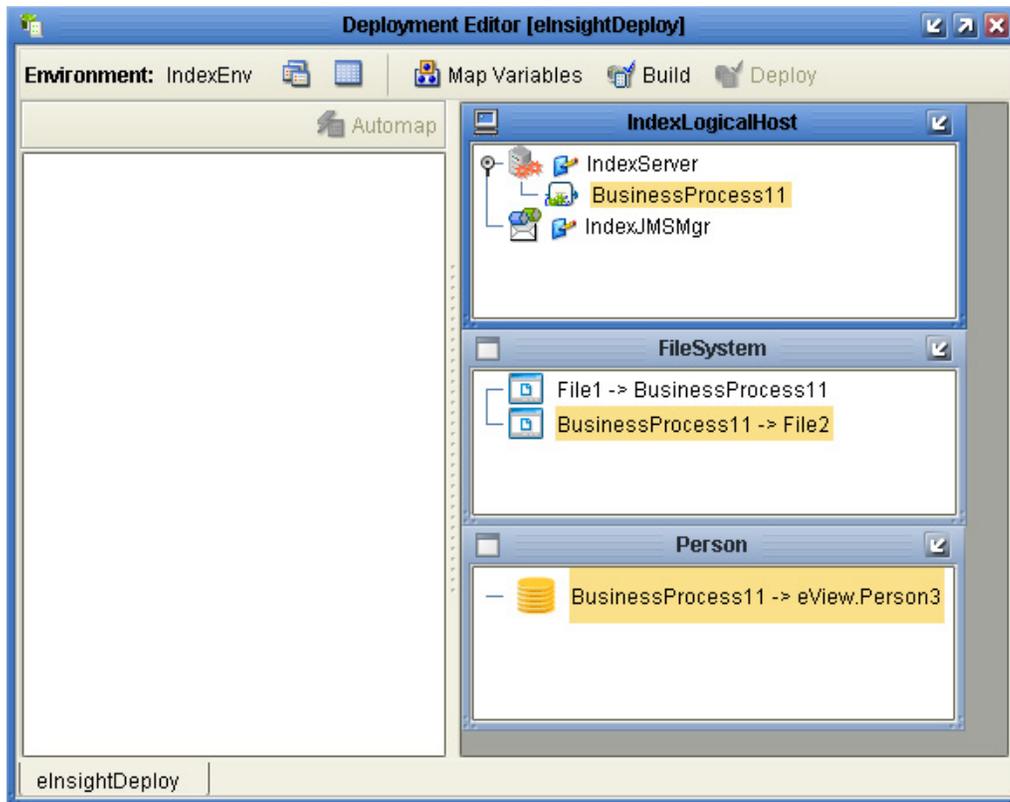
To map eInsight client Project components

- 1 With the eInsight client Project Deployment Profile open in the Deployment Editor, drag the Service/Business Process icon onto the integration or application server in the Deployment Editor (the server appears in the Logical Host).
- 2 Drag the eIndex SPV application icon onto the eIndex SPV application deployment component (this is named after the eIndex SPV application).

- 3 Drag the external system eWays to the appropriate external systems in the Environment.
- 4 If you implemented a JMS Topic, drag the topic to the JMS IQ Manager in the Logical Host (not shown).

Figure 42 illustrates the updated Deployment Profile.

Figure 42 Mapped Client Components in the Deployment Editor



- 5 Continue to “Building and Deploying the eInsight Client Project”.

Building and Deploying the eInsight Client Project

Once all Project components are mapped to the Environment, you can build the Project to create and compile all the files needed to run the Project. Deploying the Project places the files on the server.

Note: For this to be successful, the domain must be started.

To build and deploy the eInsight client Project

- 1 Click **Build** to generate the Project files.
- 2 Click **Deploy** to deploy the Project to a server.

9.4 Defining Security

eIndex SPV supports security at the user and function level, and also supports Secure Sockets Layer (SSL) authentication. For information about configuring the server to enable SSL, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

A secure user name and password must be defined for each eIndex SPV application to connect to the database and to log on to the EDM. For each user account you define, you must also specify one or more groups in order for that user to be able to perform any functions in the EDM.

Make sure the domain for the eIndex SPV application is running before performing this procedure.

Important: *In order for security roles to function correctly for the EDM, authorization security must be enabled in the Enterprise Data Manager configuration file. To enable security, set the **enable-security** element to "true". By default, this element is set to "true". For more information, see chapter 9 of the *Sun SeeBeyond eIndex Single Patient View Configuration Guide*.*

9.4.1 Defining Security (for the Sun SeeBeyond Integration Server)

Security for eIndex SPV running on the Sun SeeBeyond Integration Server (IS) is configured in the Logical Host user management category using the Enterprise Manager (see the *Sun SeeBeyond eGate Integration System Administration Guide* for more information about user management categories and setting up security). You must be connected to the server in the Enterprise Manager to perform this task.

To connect to the server

- 1 Log in to the Enterprise Manager.
- 2 In the frame on the right side of the page, click **J2EE**.

The Application Server Deployer page appears with the Manage Servers tab displayed.

- 3 In the **Add Application Server** section, fill in the fields listed in Table 5.

Table 5 Manager Server Fields

In this field ...	Type or select ...
Server Type	The type of server you are adding.
Host Name	The name of the computer on which the server resides. You can enter "localhost" if the IS resides on the same computer.
HTTP Administration Port	The HTTP port for connecting to the server. By default, the port number is 18000 .
Username	The user name to log on to the integration server.

Table 5 Manager Server Fields

In this field ...	Type or select ...
Password	The password associated with the given user name.

- 4 Click **Connect to Server**.
- 5 Once the connection is made, click **Save current user preferences**.

To create an eIndex SPV user account in the Enterprise Manager

- 1 Log on to Enterprise Manager.
- 2 On the Enterprise Manager, expand the J2EE list until you see the server to which you want to add a user account.
- 3 Right-click the server, and then click **Manage Integration Server Users**.
The **Users List** window appears.
- 4 In the **Users List** window, click **Add New User**.
The **Add/Edit User** window appears (see Figure 43).

Figure 43 Add/Edit User Window

[Users List](#) > Add/Edit User

Specify the details for this User.

User Name:*

Password:*

Confirm Password:*

Group List:

Separate multiple groups with commas.

- 5 In the **User Name** field, enter a name for the user.
- 6 In the **Password** field, enter a password for the user.
- 7 In the **Confirm Password** field, enter the password again.
- 8 In the **Group List** field, enter one or more eIndex SPV user groups, separating multiple groups with a comma. Table 6 lists and describes each eIndex SPV user group.

9 After you have added all required user groups, click **Submit**.

Table 6 User Groups and Descriptions

User Group	Description
AL.View	Gives access permission to search for and view audit log entries, and to generate and print the search results report.
Duplicate.All	Gives access permission to all potential duplicate functions.
Duplicate.AutoResolve	Gives access permission to permanently resolve potential duplicate records.
Duplicate.Print	Reserved for future functionality.
Duplicate.Resolve	Gives access permission to resolve potential duplicate records.
Duplicate.SearchAndView	Gives access permission to search for and view potential duplicate records, and to view and print the potential duplicate search results report.
Duplicate.Unresolve	Gives access permission to unresolve potential duplicate records that were previously resolved.
EO.All	Gives access permission to all enterprise object functions described below.
EO.Activate	Gives access permission to activate enterprise records.
EO.Create	Gives access permission to create new enterprise records.
EO.Compare	Gives access permission to compare enterprise records.
EO.Deactivate	Gives access permission to deactivate enterprise records.
EO.Edit	Gives access permission to modify the SBR in enterprise records.
EO.Merge	Gives access permission to merge enterprise records.
EO.OverwriteSBR	Gives access permission to modify the SBR and to lock SBR fields for overwrite.
EO.PrintComparison	Reserved for future functionality.
EO.PrintSBR	Reserved for future functionality.
EO.SearchAndViewSBR	Gives access permission to search for and view single best records, and to generate and print the search results report.
EO.Unmerge	Gives access permission to unmerge enterprise records.
EO.ViewMergeTree	Gives access permission to view a merge history of an enterprise object.

Table 6 User Groups and Descriptions

User Group	Description
eView.Admin	Gives access permission to all functions of the Enterprise Data Manager.
eView.Reports	Gives access permission to generate and view reports on the EDM. (Note that this is not required to print search results reports, which is granted by the individual search access permissions.)
eView.User	Gives access to the EDM. This group must be assigned to each user except those assigned the eView.Admin group.
eView.VIP	Gives permission to view fields masked by the VIP feature.
History.All	Gives access permission to all history functions described below.
History.Print	Reserved for future functionality.
History.SearchAndView	Gives access permission to search for and view the transaction history of enterprise records and to generate and print the search results report.
SO.All	Gives access permission to all system record functions described below.
SO.Add	Gives access permission to add system records.
SO.Edit	Gives access permission to modify system records.
SO.Merge	Gives access permission to merge system records.
SO.Print	Reserved for future functionality.
SO.Remove	Gives access permission to delete system records.
SO.Unmerge	Gives access permission to unmerge system records.
SO.View	Gives access permission to view system records.

9.4.2 Defining Security (for the Sun Java System Application Server)

Security for eIndex SPV running on the Sun Java System Application Server is configured on the Admin Console.

To create an eIndex SPV user account in the Admin Console

- 1 Log on to the Sun Java System Application Server Admin Console.
- 2 In the left portion of the page, expand **Configurations** and then expand the server on which eIndex SPV is running.
- 3 Under the server, expand **Security**, expand **Realms**, and then select **file**.
- 4 On the Edit Realm page, select **Manager Users**.
- 5 On the File Users page, select **New**.
- 6 In the **User ID** field, enter a name for the user.

- 7 In the **Password** field, enter a password for the user.
- 8 In the **Confirm Password** field, enter the password again.
- 9 In the **Group List** field, enter one or more eIndex SPV user groups, separating multiple groups with a comma. Table 6 lists and describes each eIndex SPV user group.
- 10 After you have added all required user groups, click **OK**.

Defining the Environment

The eIndex SPV Environment defines the configuration of the physical environment of the master index, including the Logical Host, integration or application server, JMS IQ Manager, constants, and external systems. This chapter describes building a generic environment for an eIndex SPV application. For more information about Environments and Environment components, see the *Sun SeeBeyond eGate Integrator User's Guide*. Additional information about Logical Hosts and domains is included in the *Sun SeeBeyond eGate Integrator System Administration Guide*.

What's in This Chapter

- [Environment Components](#) on page 119
- [Building an Environment](#) on page 120
- [Configuring a Connection Pool Through the Server](#) on page 126

10.1 Environment Components

All Projects accessing the eIndex SPV system must be configured to use the same Environment, including client Projects defining Collaborations and Business Processes that use eIndex SPV methods. The Environment requirements are different for the eIndex SPV Project and client Projects. When you deploy an eIndex SPV Project, the master index application defined by that Project becomes available to the client Projects in the Environment.

An Environment that supports the eIndex SPV server Project can include the following components.

- **Logical Hosts** - Each Environment contains one or more Logical Hosts, each of which can contain multiple instances of the Logical Host, known as *domains*. Logical Hosts are instances of the eGate runtime environment installed on a host hardware platform.
- **Integration or Application Servers** - The Logical Host contains one or more integration or application servers, which are the engines that run eGate Services and eWays. They provides services for security, transactions, and business rules execution. eIndex SPV can use the Sun SeeBeyond Integration Server or the Sun Java System Application Server.

- **JMS IQ Managers** - The Logical Host contains one or more JMS IQ Managers, which manage JMS topics (publish-and-subscribe messaging) and queues (point-to-point messaging).
- **External Systems** - An external system is a representation of a real, physical system that exists within the specific Environment, with configuration properties for locating and accessing that system. This component is required for client Projects connecting external systems with the eIndex SPV application.
- **eVision External Systems** - An eVision external system is a representation of an eVision web application. This component is required for client Projects integrating eIndex SPV with eVision Studio.
- **Oracle External Systems** - An Oracle external system is a representation of an Oracle eWay. This component is required to define the database connection pool for Projects connecting to the database using the Oracle eWay.
- **Environmental Constants** - Name/value pairs that are visible across the Environment. You can define constants for a specific Environment.

10.2 Building an Environment

Each Environment represents a unit of software that implements one or more eIndex SPV applications. You must define and configure an Environment for the master index before you can deploy the application. The tasks you can perform to build an Environment for the eIndex SPV application are described on the following pages. For client Projects that reference the eIndex SPV server Project, additional components might be required. For eVision or eInsight Environments, refer to the appropriate user's guide.

- [Creating the Environment](#) on page 120
- [Adding a Logical Host](#) on page 121
- [Adding Servers](#) on page 122
- [Adding an External System](#) on page 124
- [Adding an Oracle External System](#) on page 125

10.2.1 Creating the Environment

Once you create the Environment, you can add the necessary components. Perform the following steps to create an Environment for an eIndex SPV Project.

To create an Environment

- 1 In the Enterprise Explorer, click the **Environment Explorer** tab.
- 2 Select the **Repository** icon.
- 3 Right-click to display the **Repository** context menu.

- 4 Select **New Environment** to add an **Environment** icon to the **Environment Explorer** tab.
- 5 Click twice on the new Environment, enter a unique name for the Environment, and then press **Enter** (see Figure 44).

Figure 44 New Environment



10.2.2 Adding a Logical Host

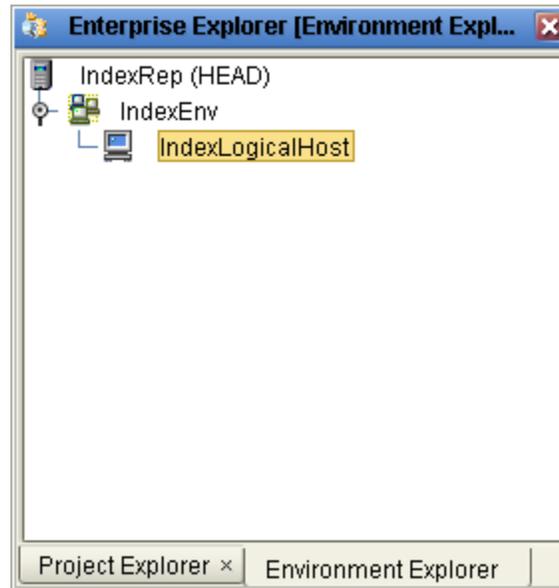
The Logical Host contains the servers that run the eIndex SPV application and messaging services. A Logical Host must be added to the Environment for all eIndex SPV implementations.

Note: A Logical Host must be added even if eIndex SPV runs on the Sun Java System Application Server, which does not run on a Sun SeeBeyond Logical Host domain.

To add a Logical Host

- 1 Select the **Environment** icon for the new Environment you created.
- 2 Right-click to display the **Environment** context menu.
- 3 Point to **New**, and then select **Logical Host**. A **Logical Host** icon appears on the **Environment Explorer** tab.
- 4 Click twice on the new Logical Host, enter a unique name for the Logical Host, and then press **Enter** (see Figure 45).

Figure 45 eIndex SPV Logical Host



10.2.3 Adding Servers

Each Environment must include at least one integration or application server and, if the eIndex SPV server Project uses a Queue, at least one message server. You can run the eIndex SPV application on either the Sun SeeBeyond Integration Server or the Sun Java System Application Server. For more information about servers, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Adding a Sun SeeBeyond Integration Server and JMS IQ Manager

Perform the following steps to add a Sun SeeBeyond Integration Server (IS) and a JMS IQ Manager to the Environment.

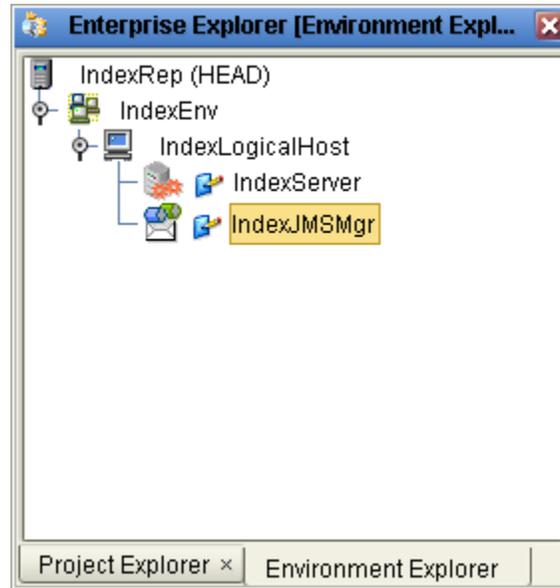
To add an Integration Server and JMS IQ Manager

- 1 In Enterprise Explorer, click the **Environment Explorer** tab.
- 2 Select the Logical Host icon of the eIndex SPV Logical Host.
- 3 To add an IS, do the following:
 - A Right-click the Logical Host icon to display the **Logical Host** context menu.
 - B Point to **New**, and then select **Sun SeeBeyond Integration Server**.
 - C Modify the name of the server in the Environment Explorer.
 - D Right-click the IS, and then select **Properties**.
 - E Enter the server's URL, administrator login ID, and password.
- 4 To add a JMS IQ Manager, do the following:
 - A Right-click the Logical Host icon to redisplay the **Logical Host** context menu.

- B Point to **New**, and then select **Sun SeeBeyond JMS IQ Manager**.
- C Modify the name of the IQ manager in the Environment Explorer.
- D Right-click the message server, and then select **Properties**.
- E Enter the server's URL, administrator login ID, and password.

Figure 46 illustrates an Environment with a Logical Host, Sun SeeBeyond Integration Server, and Sun SeeBeyond JMS IQ Manager.

Figure 46 Integration and JMS Servers



Adding Sun Java System Servers

Perform the following steps to add a Sun Java System Application Server and Sun Java System JMS Server to the Environment. You must have Java System Application Server installed on your system in order to use this feature.

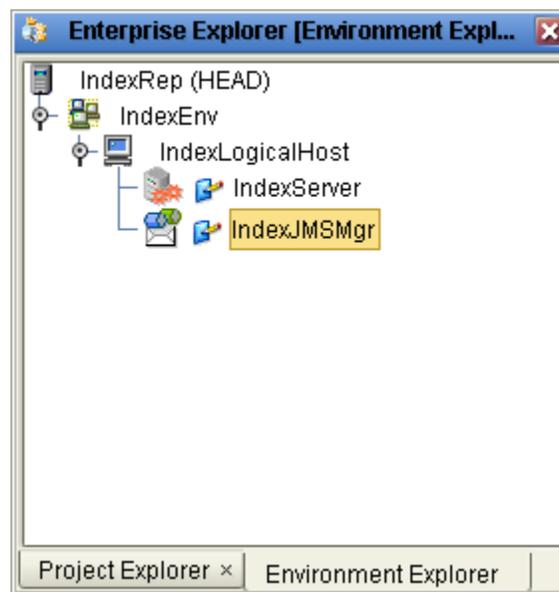
To add Sun Java System servers

- 1 In Enterprise Explorer, click the **Environment Explorer** tab.
- 2 Select the Logical Host icon of the eIndex SPV Logical Host.
- 3 To add a Sun Java System Application Server, do the following:
 - A Right-click the Logical Host icon to display the **Logical Host** context menu.
 - B Point to **New**, and then select **Sun Java System Application Server**.
 - C Modify the name of the server in the Environment Explorer.
 - D Right-click the application server, and then select **Properties**.
 - E Enter the server's URL, administrator login ID, and password.

- 4 To add a Sun Java System JMS Server, do the following:
 - A Right-click the Logical Host icon to redisplay the **Logical Host** context menu.
 - B Point to **New**, and then select **Sun Java System JMS Server**.
 - C Modify the name of the IQ manager in the Environment Explorer.
 - D Right-click the message server, and then select **Properties**.
 - E Enter the server's URL, administrator login ID, and password.

Figure 47 illustrates an Environment with a Logical Host, Sun Java System Application Server, and Sun Java System JMS Server.

Figure 47 Application and JMS Servers



10.2.4 Adding an External System

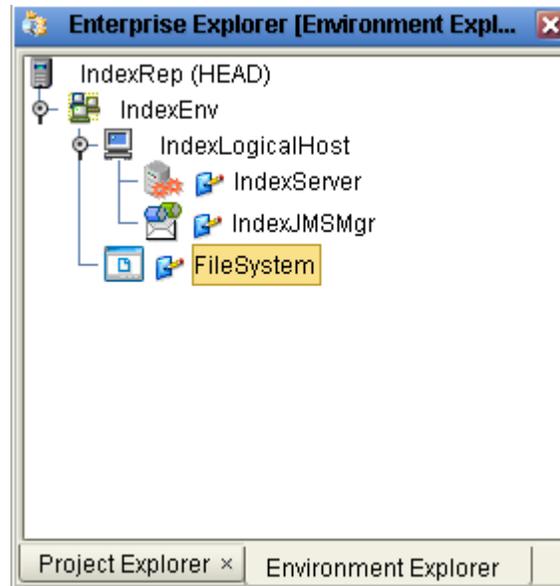
External systems are required for Projects that connect an external system to the eIndex SPV master index. The Connectivity Maps in these Projects include External Applications that will be mapped to the external system. Perform the following steps to add an external system to the eIndex SPV Environment.

To add an external system

- 1 In Enterprise Explorer, click the **Environment Explorer** tab.
- 2 Select the eIndex SPV Environment icon.
- 3 Right-click the eIndex SPV Environment icon to display the **Environment** context menu.
- 4 Point to **New**, and then select **<type> External System**, where **<type>** is the type of eWay connecting the external system to eGate (such as Oracle, TCP/IP, and so on).

- 5 In the **External System Name** field, enter the name of the new external system.
- 6 To configure the external system, right-click the new external system in the Environment, and then select Properties from the context menu.
- 7 Configure the properties as described in the appropriate user's guide.
- 8 Repeat these steps for each external system defined in the Projects that will be using this Environment.

Figure 48 File External System



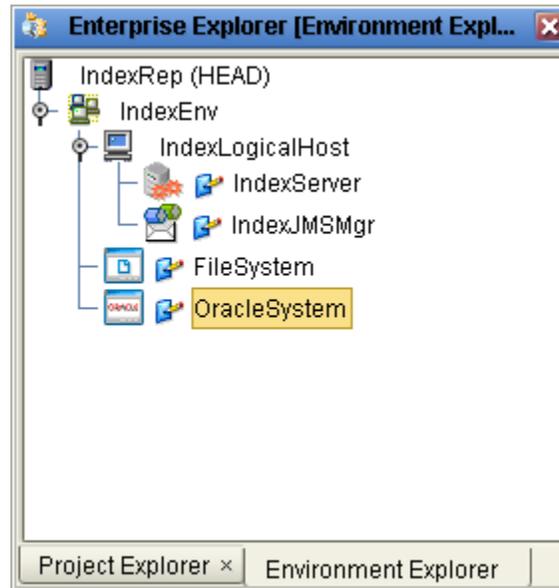
10.2.5 Adding an Oracle External System

An Oracle external system is required for eIndex SPV server Projects that connect to the database using an Oracle eWay. It is required for Projects running on the Sun SeeBeyond Integration Server, but is optional for Projects running on the Sun Java System Integration Server. Perform the following steps to add an Oracle external system to the eIndex SPV Environment. If you are not using an Oracle eWay, be sure to follow the instructions under [Configuring a Connection Pool Through the Server](#) on page 126 to create a database connection pool.

To add an Oracle external system

- 1 In Enterprise Explorer, click the **Environment Explorer** tab and then select the eIndex SPV Environment icon.
- 2 Right-click the eIndex SPV Environment icon to display the context menu.
- 3 Point to **New**, and then select **Oracle External System**.
- 4 In the **External System Name** field, enter the name of the new Oracle external system.
- 5 Click **OK**.

Figure 49 Oracle External System



- 6 To configure the connection pool, right-click the Oracle External System icon, and then select **Properties**.

The **Properties** window appears.

- 7 Expand **Outbound Oracle eWay**, and then select **JDBC Connector Settings**.
- 8 Define each property in the right portion of the window with information specific to the eIndex SPV database you created. All fields on the Properties windows are described in the *Sun SeeBeyond eWay Adapter for Oracle User's Guide*. Be sure to only define the properties for the driver type you are using.

Note: Creating the database is described in [Chapter 7](#) of this guide. Use the information for that database for the properties on this window.

- 9 When you finish defining the properties, click **OK** to close the **Properties** dialog.

10.3 Configuring a Connection Pool Through the Server

If you are using an Oracle eWay and have added and configured the Oracle external system in the Environment, you have already configured your database connection pool for the eIndex SPV database and do not need to perform this step. If you are running the application on the Sun Java System Application Server and are **not** using the Oracle eWay to connect to the database, you must configure the JDBC connection pool and resources using the Sun Java System Application Server Admin Console.

For more information about the procedures in this section, see the online help provided with the Sun Java System Application Server Admin Console.

Important: Before creating the connection pool, install the Oracle driver on the application server or copy the **ojdbc14.jar** file from your Oracle client installation (<Oracle_client>\jdbc\lib) to <application_server_home>\lib).

Step 1: Create a JDBC Connection Pool

The JDBC connection pool provides connections for the master index database. Before proceeding, make sure you have the relevant information about the master index database (such as the database name, URL, and administrator login credentials).

To create a JDBC connection pool

- 1 In the left portion of the Sun Java System Application Server Admin Console, expand **Resources**, expand **JDBC**, and then select **Connection Pools**.
- 2 On the Create Connection Pool page, click **New**.
- 3 Enter values in the fields described in Table 7, and then click **Next**.

Table 7 Connection Pool General Settings

Field	Description
Name	A name for the connection pool.
Resource Type	The Java class for the connection pool.
Database Vendor	The database platform for the eIndex SPV database. Select Oracle .

- 4 Enter a value in the **DataSource Classname** field, and then click **Next**.

Note: You can accept the default value for this field if it is provided.

- 5 On the Create Connection Pool page, do the following:
 - ♦ In the **General Settings** section, verify that the values are correct.
 - ♦ In the **Properties** section at the bottom of the page, enter information for the eIndex SPV database.
 - ♦ Keep the default values in the remaining sections (you can edit these later if needed).
- 6 Click **Finish**.

Step 2: Create a JDBC Resource

A JDBC resource (also known as a data source) gives the master index applications the ability to connect to the database.

To create a JDBC resource:

- 1 In the left portion of the Sun Java System Application Server Admin Console, expand **Resources**, expand **JDBC**, and then select **JDBC Resources**.
- 2 On the Create JDBC Resource page, click **New**.

- 3 Enter the field values described in Table 8.

Table 8 JDBC Resources Fields

Field	Description
JNDI Name	A unique name for the JDBC resource. This name must begin with "jdbc/", which should be followed by "<app_name>DataSource" (where <app_name> is the name of the eIndex SPV application). For example: jdbc/PersonDataSource.
Pool Name	The name of the JDBC connection pool associated with the JDBC resource.
Description	(Optional) A brief description of the JDBC resource.

- 4 In the **Available** box in the **Targets** section, select the server on which the resource is available, and then click **Add**.
- 5 Click **OK**.

Maintenance Tasks

Once you move the eIndex SPV system into production, you should perform certain maintenance tasks regularly to keep the system running smoothly. Primary tasks include archiving Repository components, monitoring and troubleshooting runtime components, and backing up the eIndex SPV database. You might also need to make changes to the eIndex SPV server Project or to the Java Collaborations or Business Processes that reference the eIndex SPV server Project. This chapter describes these maintenance tasks.

What's in This Chapter

- [Archiving Repository Information](#) on page 129
- [Maintaining the Database](#) on page 130
- [Monitoring Day to Day Activity](#) on page 131
- [Implementing Changes to the eIndex SPV Project](#) on page 137

11.1 Archiving Repository Information

Back up the Repository on a regular basis. The frequency of these backups depends on the internal policies and procedures of your organization. You can back up the entire Repository using a command line script. When you perform a full Repository backup, the Repository is locked and cannot be modified while the backup is in progress.

You can also back up just the eIndex SPV server and client Projects using the export function of Enterprise Designer. Exporting Projects allows you to maintain a snapshot of each Project before and after changes were made to Project components. Use this method to backup any customizations you make to a Project, including custom plugins, changes to configuration files, custom database scripts, and so on.

For information and instructions for exporting Projects, see the *Sun SeeBeyond eGate Integrator User's Guide*. For information and instructions for backing up and restoring the Repository, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

11.2 Maintaining the Database

The database requires periodic maintenance tasks, such as backing up information or archiving certain tables. Perform backups regularly, and use the standards and policies of your organization to determine the best methods for backing up data.

11.2.1 Backing up the Database

The eIndex SPV database must be backed up on a regular basis. Typically, the database should be backed up once a month or once a quarter, depending on the size of the database and the volume of data being processed. The frequency of your database backups depends on your organization's internal policies and practices. Use your normal procedures for backing up a high availability database (this procedure should be determined by a database administrator).

Online Backups

The best practice for backing up the eIndex SPV database is an online backup during which the database is not shut down. (Note that this does require an offline backup as a starting point to which any online changes can be applied in the event the database must be restored). An online backup will always take a consistent snapshot, though it might not backup all transactions in progress.

Each transaction in eIndex SPV is saved under one commit command, so the state of the database is always consistent when a backup is performed. The history tables always match the transactions in the current tables and no partial transactions are committed. Even if a transaction is underway at the time of the backup, the database is consistent.

For the most reliable backups, Oracle recommends that you run your Oracle database in ARCHIVELOG mode. ARCHIVELOG mode ensures that your database is protected from both instance and media failure and, because all changes made to the database are saved in a redo log, all database updates are available for recovery rather than just the most recent changes. Online backups are available in this mode.

Offline Backups

If needed, you can perform offline backups of the eIndex SPV database. In this case, you must queue any incoming messages using the JMS IQ Manager and undeploy the eIndex SPV application before beginning the backup. Once the backup is complete, restart the database, redeploy and enable the eIndex SPV application, and then process the messages queued by the JMS IQ Manager.

11.2.2 Database Restoration

In the unlikely event that you need to restore the eIndex SPV database to a previously archived version, you must undeploy the eIndex SPV application prior to performing the restoration to ensure that the application retrieves the correct sequence numbers from the database once it is restored. Any new transactions that occurred after the

archived version was created will be lost, but they can be resent through eGate if the JMS IQ Manager is configured to journal all messages.

11.2.3 Archiving

In addition to regular database backups, some of the eIndex SPV database tables can grow very large. For performance reasons, you might want to archive the information in the sbyn_assumedmatch and the sbyn_audit tables.

11.3 Monitoring Day to Day Activity

The following tools are available for finding and correcting errors in runtime components. These should be monitored on a daily basis to ensure your system is running smoothly and error-free.

- [Enterprise Manager Monitor](#) on page 131
- [Log Files](#) on page 132
- [Alerts](#) on page 133

11.3.1 Enterprise Manager Monitor

The Enterprise Manager Monitor (Monitor) allows you to quickly identify problems with components or systems in the Repository framework and, in some cases, to correct the problem.

About the Enterprise Manager Monitor

The Monitor alerts you to the status of components (for example, whether they are running) and allows you to send commands to the components such as start or shut down. From the Monitor, you can double-click on an eIndex SPV application or related Project components to go directly to the problem. The Monitor allows you to filter the list of displayed instances to quickly identify exceptions and to navigate to specific versions of a Service to monitor the progress of each instance. eIndex SPV components cannot be stopped or restarted from the Monitor, but eWays and other associated components can.

The Monitor provides visual cues to let you know when a component needs attention. For example, the Connectivity Map in the Project Explorer displays a flashing red square when a Service becomes inactive. If you configure the Alert Agent or SNMP Agent, you can avoid having to run the Monitor continuously. The agent will notify you when the specified problem occurs.

For more information on using Enterprise Manager Monitor, see the following documents:

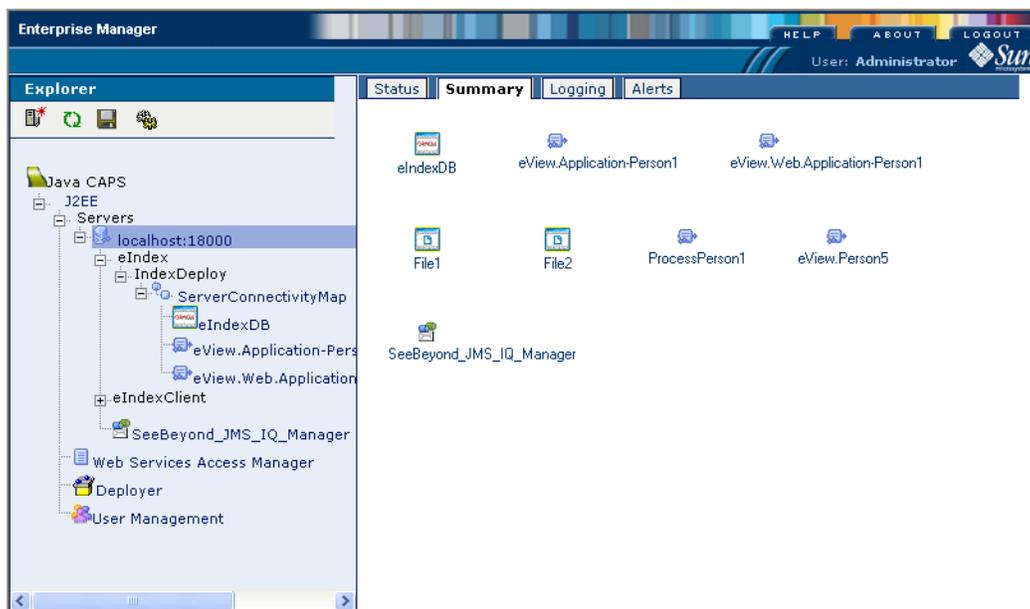
- *Sun SeeBeyond eGate Integrator System Administration Guide*
- *Sun SeeBeyond eGate Integrator JMS Reference Guide*

- *Sun SeeBeyond eInsight Business Process Manager User's Guide*

Enabling Monitoring for eIndex SPV

You can use the Enterprise Manager to monitor most components of an eIndex SPV system, including Collaborations, Business Processes, and eWays. In order to view the special tools for each component type, you must have the Enterprise Manager Plug-in files uploaded and installed for those types. For more information about installing the eView Monitor client, see **“Installing the eIndex SPV Monitor Client” on page 40**. For information on installing the plug-in for eWays, eInsight BPM, and other components, see the appropriate user's guide. Figure 50 shows the **Summary** tab of the Monitor for an eIndex SPV server Project and Collaboration Project. You can click on any of the displayed components for more information about their status, alerts, log entries, and so on.

Figure 50 Monitoring eIndex SPV Project Components

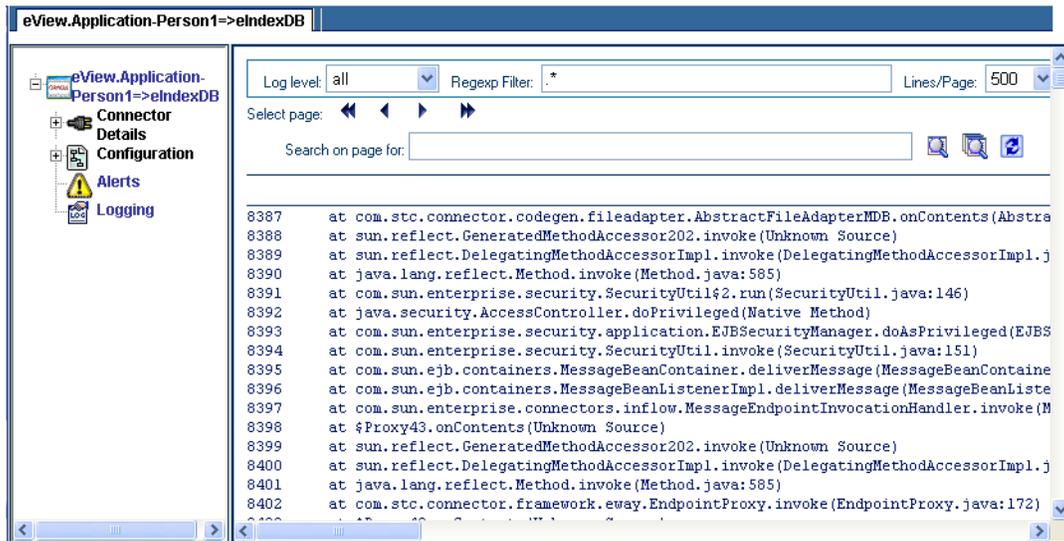


11.3.2 Log Files

When a component or system is not working, errors are written to a log file to help you diagnose the problem. You can specify the level at which events are recorded in the log files. On a daily basis, review the runtime log files for eIndex SPV Project components and examine any messages with a severity level of FATAL, ERROR, or WARN. Periodically, you might want to archive the log file.

Figure 51 shows a log file for the Oracle eWay in the eIndex SPV server Project. For more information on log files, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Figure 51 eIndex SPV Database Logging



11.3.3 Alerts

Alerts are triggered when certain conditions occur in Project components. The condition might be some type of problem that must be corrected, such as when the connection to the database is lost, or it could be simply a warning, such as when a transaction error occurs on the EDM. From the Monitor, you can view component alerts, modify the status of an alert, or delete alerts.

Figure 52 shows an alert for a Collaboration in an eIndex SPV client Project. A list of eIndex SPV alerts appears in Table 9. For more information on alerts, see the *Sun SeeBeyond eGate Integrator System Administration Guide*.

Figure 52 Alert for eIndex SPV Collaboration

Component: e51x Servers localhost:18000 eIndexClient Coll											
Summary:		Fatal: 0		Critical: 0		Major: 0		Minor: 0		Warning: 0	Info: 1
Date	ID	Environment	Logical Host	Server	Project	Deployment	Component	Severity	Type		
Wed Mar 01 13:43:57 PST 2006	10	IndexEnv	IndexLH	IndexServer	eIndexClient	CollabDeploy	ProcessPerson1	INFO	COLLABORATION	U	

The following table list and describes the alerts generated by eIndex SPV. Most of these alerts are the result of calling specific eIndex SPV methods in Collaborations or Business Processes. Variables in the error message appear in italics. The alert code for all eIndex SPV alerts is “EVIEW-00001”.

Table 9 eIndex SPV Alert Messages

Error Message	Cause	Severity Level
com.stc.eindex.ejb.update.UpdateException: OPSEException: <i>message_text</i>	Generally an Oracle database error occurred and an Oracle error message is generated; for example, SQL Statement: insert into <table> (<i>columns</i>) values (<i>values</i>) ORA-00001: unique constraint (EVIEW.PK_SBYNSYSTEMOBJECT) violated	Warning
com.stc.eindex.master.ConnectionInvalidException: Failed to get connection.	The connection to the database is down. Make sure the database is running and that the logon information is correct in the data source configuration.	Warning
com.stc.eindex.master.ProcessingException: Inactive system object not found.	A call to activateSystemObject did not find an active system record matching the system, local ID, and status.	Warning
com.stc.eindex.master.ProcessingException: activateEnterpriseObject(): Invalid EUID: <i>eid</i>	A null or invalid EUID was specified in a call to activateEnterpriseObject.	Warning
com.stc.eindex.master.ProcessingException: activateEnterpriseObject(): EUID: <i>eid</i> does not have inactive status.	A call to activateEnterpriseObject is attempting to activate an EUID that is already active.	Warning
java.lang.NullPointerException: null	This indicates a severe alert. Contact Sun SeeBeyond support for assistance.	Warning
com.stc.eindex.master.ProcessingException: addSystemObject(): system key (<i>system_code, local_ID</i>) already mapped to EUID: <i>eid</i>	A call to addSystemObject is trying to add a system record that already exists in the master index database.	Warning
com.stc.eindex.master.ProcessingException: addSystemObject(): EUID: <i>eid</i> does not exist	A call to addSystemObject is trying to add a system record to an EUID that does not exist.	Warning
com.stc.eindex.master.ProcessingException: createEnterpriseObject(): system key (<i>system_code, local_ID</i>) already mapped to EUID: <i>eid</i>	A call to createEnterpriseObject is trying to add an enterprise record with a system record that already exists in the master index database.	Warning
com.stc.eindex.master.ProcessingException: deactivateSystemObject(): system key (<i>system_code, local_ID</i>) is not active or does not exist	A call to deactivateSystemObject is trying to deactivate a record that is already inactive, or the master index could not find a system object matching the given system and local ID.	Warning
com.stc.eindex.master.ProcessingException: Invalid EUID.	A null or invalid EUID was specified in a call to deactivateEnterpriseObject.	Warning

Table 9 eIndex SPV Alert Messages

Error Message	Cause	Severity Level
com.stc.eindex.master.ProcessingException: deactivateEnterpriseObject(): EUID <i>eid</i> does not have active status. Status is: inactive	A call to deactivateEnterpriseObject is trying to deactivate a record that is already inactive.	Warning
com.stc.eindex.master.ProcessingException: undoAssumedMatch(): Record has been modified by another user. EUID has already been merged: <i>eid</i>	A call to undoAssumedMatch cannot be completed because the record to be unmatched was already merged with another record.	Warning
com.stc.eindex.assumedmatch.AssumedMatchException: com.stc.eindex.page.PageException: com.stc.eindex.ops.exception.OPSExeption: OPSExeption: Child node is null	This is a serious error that generally indicates that data in the sbyn_transaction log is corrupt. Contact Sun SeeBeyond support for assistance.	Warning
com.stc.eindex.master.ProcessingException: undoAssumedMatch(): Record has been modified by another user. Assumed match has already been undone: <i>assumed_match_id</i>	A call to undoAssumedMatch cannot be completed because the assumed match was already reversed.	Warning
com.stc.eindex.master.ProcessingException: mergeEnterpriseObject(): Record has been modified by another user. Destination EUID not found: <i>eid</i>	The destination EUID specified in a call to mergeEnterpriseObject does not exist, no destination EUID is specified, or the EUIDs specified are already merged.	Warning
com.stc.eindex.master.ProcessingException: mergeEnterpriseObject(): Record has been modified by another user. Source EUID not found: <i>eid</i>	The source EUID specified in a call to mergeEnterpriseObject does not exist, no source EUID is specified, or the source EUID specified was already merged into another EUID record.	Warning
com.stc.eindex.master.ProcessingException: EUID must be a selected field.	The EUID is not specified as part of the search options (class SearchOptions). By default, the EUID is specified.	Warning
com.stc.eindex.master.ProcessingException: At least one SystemObject must be populated.	A search was attempted using a system object with no field values, which means there was no criteria on which to search.	Warning
com.stc.eindex.master.ProcessingException: transferSystemObject(): transfer must be between two different EUIDs. Both EUIDs are: <i>eid</i>	The EUID specified in a call to transferSystemObject is the same EUID to which the system object already belongs.	Warning
com.stc.eindex.master.ProcessingException: unmergeEnterpriseObject(): Record has been modified by another user. EUID: <i>eid</i>	A call to unmergeEnterpriseObject failed because the record to be unmerged was modified by another user before the unmerge transaction was finalized.	Warning

Table 9 eIndex SPV Alert Messages

Error Message	Cause	Severity Level
com.stc.eindex.master.ProcessingException: unmergeEnterpriseObject(): Record has been modified by another user. EUID has already been unmerged: <i>eid</i>	A call to unmergeEnterpriseObject failed because the records are already unmerged.	Warning
com.stc.eindex.master.ProcessingException: unmergeSystemObject(): Source system record not found: (<i>system_code</i> , <i>local_ID</i>)	The source system record specified in a call to unmergeSystemObject is invalid or its status could not be found.	Warning
com.stc.eindex.master.ProcessingException: unmergeSystemObject(): Record has been modified by another user. Source system record has already been deactivated: (<i>system_code</i> , <i>local_ID</i>)	The source system record specified in a call to unmergeSystemObject has a status of "inactive" and cannot be unmerged.	Warning
com.stc.eindex.master.ProcessingException: unmergeSystemObject(): Record has been modified by another user. Source system record is not in merged status: (<i>system_code</i> , <i>local_ID</i>)	The system records specified in a call to unmergeSystemObject have already been unmerged.	Warning
com.stc.eindex.master.ProcessingException: unmergeSystemObject(): Record has been modified by another user. Source system status unrecognized: (<i>system_code</i> , <i>local_ID</i>)	The status of the source system specified in a call to unmergeSystemObject is invalid.	Warning
com.stc.eindex.master.ProcessingException: unmergeSystemObject(): Destination system record not found: (<i>system_code</i> , <i>local_ID</i>)	The destination system record specified in a call to unmergeSystemObject is invalid.	Warning
com.stc.eindex.master.ProcessingException: unmergeSystemObject(): no transactions found for LID merge.	The system records specified in a call to unmergeSystemObject were not previously merged.	Warning
com.stc.eindex.master.ProcessingException: updateSystemObject(): SO (<i>system_code</i> - <i>local_ID</i>) is not Active.	The system record specified by a call to updateSystemObject is not active and cannot be updated.	Warning
com.stc.eindex.master.ProcessingException: mergeSystemObject(): Record has been modified by another user. Source system record not found: (<i>system_code</i> , <i>local_ID</i>)	The status of the source system record specified in a call to mergeSystemObject could not be found or the source system specified is invalid.	Warning
com.stc.eindex.master.ProcessingException: mergeSystemObject(): Record has been modified by another user. Source system record has already been deactivated: (<i>system_code</i> , <i>local_ID</i>)	The source system specified in a call to mergeSystemObject has a status of "inactive" and cannot be merged.	Warning

Table 9 eIndex SPV Alert Messages

Error Message	Cause	Severity Level
com.stc.eindex.master.ProcessingException: mergeSystemObject(): Record has been modified by another user. Source system record has already been merged: (system_code, local_ID)	The source system specified in a call to mergeSystemObject has a status of "merged" (that is, it has already been merged into another record) and it cannot be merged.	Warning
com.stc.eindex.master.ProcessingException: mergeSystemObject(): Record has been modified by another user. Destination system record not found: (system_code, local_ID)	The status of the destination system record specified in a call to mergeSystemObject could not be found or the system specified is invalid.	Warning
com.stc.eindex.master.ProcessingException: mergeSystemObject(): Record has been modified by another user. Destination system record has already been deactivated: (system_code, local_ID)	The destination system specified in a call to mergeSystemObject has a status of "inactive" and cannot be merged.	Warning
com.stc.eindex.master.ProcessingException: mergeSystemObject(): Record has been modified by another user. Destination system record has already been merged: (system_code, local_ID)	The destination system specified in a call to mergeSystemObject has a status of "merged" (that is, it has already been merged into another record) and it cannot be merged.	Warning
com.stc.eindex.master.ProcessingException: mergeSystemObject(): system object keys are equal (system_code, local_ID)	The system objects specified in a call to mergeSystemObject are the same system object.	Warning

11.4 Implementing Changes to the eIndex SPV Project

After eIndex SPV has been in production, you might need to make changes to your project. For example, if you add a new external system, you need to add that system to the eIndex SPV database and you might need to modify the object structure and OTDs as well as update the application files. Changes occur as the needs of your end users evolve and as additional external systems are added. Do not make changes to the system hastily. Handle changes using the same change management process that was originally used to deploy your project. Applying this same process of planning, configuration, testing, migration, monitoring, and re-evaluation will help ensure successful updates.

11.4.1 Modifying Configuration Files

Over time, you might need to make changes to your configuration files, such as adding fields or objects to the object structure, changing the appearance of the EDM, or fine-tuning the matching process. Whenever you make a change to an eIndex SPV

configuration file, you must disable the eIndex SPV server Project, regenerate the application, and then redeploy the Project. In addition, if any Java Collaborations in client Projects reference the eIndex SPV application, you must re-import the regenerated .jar files from the eIndex SPV server Project into the Java Collaboration.

This section provides tips for updating components of the configuration files. In order for any of these changes to take affect, you must regenerate the application and rebuild and redeploy the Project.

Updating the Object Structure

If you make any changes to the object structure, keep the following in mind.

- If you want the new fields or objects to appear on the EDM, make sure to add them to the first section of the Enterprise Data Manager file and to any of the EDM page definitions later in the file (this includes search pages).
- If the new fields require normalization, parsing, or phonetic encoding, define the new structures in the Match Field file.
- If a new field will be used for matching, add it to the blocking query used for match processing as well as to the match string in the Match Field file.
- If the new fields or objects will be included in incoming messages, add them to the inbound OTD structure (the outbound OTD will be updated when you regenerate the application).

Updating Normalization and Standardization Structures

If you define normalization, standardization, or phonetic encoding for fields that are not currently defined in the Match Field file, or if you change existing standardization structures, make sure to do the following.

- Use the appropriate standardization type, domain selector, and field IDs.
- Add the new fields that will store the standardized versions of the original field value to the appropriate objects in the Object Definition file.
- Add new columns to the database to store the standardized field values.

Updating the Match String

If you make changes to the match string, update the database indexes and the blocking query in the Candidate Select file accordingly. For example, if you remove a field from the match string, you might also want to remove that field from the blocking query and database indexes. If you add a field to the match string, add the field to the blocking query and to the appropriate database index to maintain performance.

11.4.2 Modifying Standard Project Components

Whenever changes are made to components of an eIndex SPV Project outside of the eIndex SPV application, such as modifying the Connectivity Map, eWay properties, Collaborations, or OTDs, the eIndex SPV application does not need to be regenerated;

however the Project containing the changed components must be redeployed in order for the changes to take effect.

11.4.3 Modifying the Database

There might be times when you need to modify the eIndex SPV database. For example, you might need to add or modify a stored procedure or index, or you might need to add new external systems. You must modify the database if you add fields or objects to the eIndex SPV object structure; the database should be updated to reflect the new structure. If you make changes to the database, rebuild and redeploy the eIndex SPV server Project to ensure the changes are picked up by the application.

11.4.4 Modifying Security

You can define new users for the database at any time using standard SQL statements to create the type of user you want to define. You can also add new users for the Enterprise Data Manager through the Integration Server User Management function of the Enterprise Manager. Neither of these procedures require any stoppage of the database or of the eIndex SPV application and no redeployment is required.

11.4.5 Modifying the Local ID Format

If you need to modify the local ID format for an external system, regenerate the application after you make the changes and then redeploy the Project. Any Collaborations that reference the eIndex SPV application must also be recompiled and those Projects redeployed. If you extend the length of a local ID past 20 characters, make sure to increase the length of any database columns containing local IDs. Local ID columns are found in the following tables: sbyn_<parent_object>, sbyn_assumedmatch, sbyn_enterprise, sbyn_systemobject, and sbyn_transaction.

Field Notations

The configuration files use specific notations to define a specific field in an enterprise or system object. There are three different type of notations used to specify a specific field or group of fields: ePath, qualified field name, and simple field name. This appendix describes each type of notation.

What's in This Appendix

- [ePath](#) on page 140
- [Qualified Field Names](#) on page 142
- [Simple Field Names](#) on page 143

A.1 ePath

In Best Record file, an *element path*, called “ePath”, is used to specify the location of a field or list of fields. ePaths are also used in the **StandardizationConfig** element of the Match Field file. An ePath is a sequence of nested nodes in an enterprise record where the most nested element is a data field or a list of data fields. ePaths allow you to retrieve and transform values that are located in the object tree.

ePath strings can be of four basic types:

- **ObjectField** - represents a field defined in the eIndex object structure.
- **ObjectNode** - represents a parent or child object defined in the eIndex object structure.
- **ObjectField List** - a list of references to certain ObjectFields in the eIndex object structure.
- **ObjectNode List** - a list of references to certain ObjectNodes in the eIndex object structure.

A context node is specified when evaluating each ePath expression. The context is considered as the root node of the structure for evaluation.

Syntax

The syntax of an ePath consists of three components: nodes, qualifiers, and fields, as shown below.

```
node{.node{'['qualifier']'}+}.field
```

- **Node** - specifies the node type and optionally includes qualifiers to restrict the number of nodes. A node without any qualifier defaults to only the first node of the specified type. Use “node.*” to address a node rather than a field.
- **Qualifier** - restricts the number of nodes addressed at each level. The following qualifiers are allowed:
 - ♦ * (asterisk) - denotes all nodes of the specified type.
 - ♦ **int** - accesses the node by index.
 - ♦ **@keystring= valuestring** - accesses the node using a key-value pair. Only one instance of the node is addressed using keys. If a composite key is defined, then multiple key-value pairs can be separated by a comma in the ePath (for example, [**@key1=value1,@key2=value2**]). The following ePath uses the keystring qualifier and returns the alias where the unique key field type is “Main”. It returns only one alias in a given record.


```
Person.Alias[@type=Main]
```
 - ♦ **filter=value** - considers only nodes whose field matches the specified value. A subset of nodes is addressed using filters. Multiple filter-value pairs can be separated by a comma (for example, [**filter1=value1, filter2=value2**]). The following ePath uses the filter qualifier and returns all aliases where the last name is “Jones”.


```
Person.Alias[lastname=Jones]
```
- **Field** - designates the field to return, and is in the form of a string.

Example

The following sample illustrates an object structure containing a system object from Site A with a local ID of 111. The object contains a first name, last name, and three addresses. Following the sample, there are several ePath examples that refer to various elements of this object structure along with a description of the data in the sample object structure referred by each ePath.

```
Enterprise
  SystemObject - A 111
    Person
      FirstName
      LastName
      -Address
        AddressType = Home
        Street = 800 Royal Oaks Dr.
        City = Monrovia
        State = CA
        PostalCode = 91016
      -Address
        AddressType = Office
        Street = 181 E. Huntington Dr.
        City = Monrovia
        State = CA
        PostalCode = 91016
      -Address
        AddressType = Billing
        Street = 100 Grand Avenue
        City = El Segundo
        State = CA
```

PostalCode = 90245

- **Person.Address.City**
Equivalent to **Person.Address[0].City**.
- **Person.FirstName** (uses Person as the context)
Equivalent to **Enterprise.SystemObject[@SystemCode=A, @Lid=111].Person.FirstName** with Enterprise as the context.
- **Person.Address[@AddressType=Home].City**
Returns a single ObjectField reference to "Monrovia" (the City field of the home address).
- **Person.Address[City=Monrovia,State=CA].Street**
Returns a list of ObjectField references: "800 Royal Oaks Dr.", "181 E. Huntington Dr." (the street fields for both addresses where the city is Monrovia and the state is CA). Note that a reference to the **Billing** address is not returned.
- **Person.Address[*].Street**
Returns a list of ObjectField references: "800 Royal Oaks Dr.", "181 E. Huntington Dr.", "100 Marine Parkway". Note that all references to **Street** are returned.
- **Person.Address[2].***
Addresses the second address object as an ObjectNode instead of an ObjectField.

A.2 Qualified Field Names

The Candidate Select file and the **MatchingConfig** element of the Match Field file use qualified field names to specify the location of a field. This method defines a specific field and is not used to define a list of fields. A qualified field name is a sequence of nested nodes in an enterprise record where the most nested element is a data field. There are two types of qualified field names.

- **Fully qualified field names** - Allows you to define fields within the context of the enterprise object; that is, the field name uses "Enterprise" as the root. These are used in the **MatchingConfig** element of the Match Field file and to specify the fields in a query block in the Candidate Select file.
- **Qualified field names** - Allows you to define fields within the context of the Person object; that is, the field name uses the Person object as the root. These are used in the Candidate Select file to specify the source fields for the blocking query criteria.

Syntax

The syntax of a fully qualified field name is:

```
Enterprise.SystemSBR.<parent_object>.<child_object>.<field_name>
```

where *<parent_object>* refers to the name of the parent object in the index, *<child_object>* refers to the name of the child object that contains the field, and *<field_name>* is the full name of the field. If the parent object contains the field being defined, the child object is not required in the path.

The syntax of a qualified field name is:

```
<parent_object>.<child_object>.<field_name>
```

Example

The following sample illustrates an object structure that could be defined in the Object Definition file. The object contains a Person parent object, and an Address and Phone child object.

```
Person
  FirstName
  LastName
  DateOfBirth
  Gender
  -Address
    AddressType
    StreetAddress
    Street
    City
    State
    PostalCode
  -Phone
    PhoneType
    PhoneNumber
```

The following *fully* qualified field names are valid for the sample structure above.

- **Enterprise.SystemSBR.Person.FirstName**
- **Enterprise.SystemSBR.Person.Address.StreetAddress**
- **Enterprise SystemSBR.Person.Phone.PhoneNumber**

The qualified field names that correspond with the fully qualified names listed above are:

- **Person.FirstName**
- **Person.Address.StreetAddress**
- **Person.Phone.PhoneNumber**

A.3 Simple Field Names

The Enterprise Data Manager file uses simple field names to specify the location of a field that appears on the EDM. These are used in the GUI configuration section of the file. Simple field names define a specific field and are not used to define a list of fields. They include only the field name and the name of the object that contains the field. Simple field names allow you to define fields within the context of an object.

Syntax

The syntax of a simple field name is:

```
<object>.<field_name>
```

where *<object>* refers to the name of the object that contains the field being defined and *<field_name>* is the full name of the field.

Example

The following sample illustrates an object structure that could be defined in the Object Definition file. The object contains a Person parent object, and an Address and Phone child object.

```
Person
  FirstName
  LastName
  DateOfBirth
  Gender
  -Address
  AddressType
    StreetAddress
    Street
    City
    State
    PostalCode
  -Phone
    PhoneType
    PhoneNumber
```

The following simple field names are valid for the sample structure above.

- **Person.FirstName**
- **Address.StreetAddress**
- **Phone.PhoneNumber**

Initial Load Tips

Once you configure the eIndex SPV application, you need to load existing legacy data from the external systems that will share data with eIndex SPV into the master index database. Due to the large number of records being loaded into the database, this process can be time-consuming and resource-intensive. This appendix provides tips to help make the initial data load process more efficient.

What's in This Chapter

- [Legacy Data](#) on page 145
- [Database Indexes](#) on page 145
- [Threshold Parameters](#) on page 146

1.1 Legacy Data

Prior to loading any legacy data into the eIndex SPV database, perform a thorough analysis and validation of the data you are loading, and then try to correct any errors you find. Things you should check for include the following.

- Null values in fields that do not allow null values
- Proper formatting, such as left or right justification, leading zeros, allowed characters, and so on
- Default values in fields that are used for blocking or matching
- Values that should not exist in specific fields
- Invalid day or month is a date field
- Missing system code in a local ID/system object
- Missing first or last name in an alias field (for person indexes)

1.2 Database Indexes

The database indexes required for the initial load process differ somewhat from the optimal indexes for running eIndex SPV in production. At a minimum, you should create an index for each block defined in the blocking query used for matching. It will

also help to remove the indexes from the `sbyn_transaction` and `sbyn_potentialduplicate` tables. After the initial load is complete, you can restore any deleted indexes. The code for restoring the indexes can be found in the database creation script in the eIndex SPV server Project.

1.3 Threshold Parameters

You can modify the parameters of the Threshold file to help the initial load process run more quickly. If you are relatively certain of the quality of the data you are loading, you can raise the match and duplicate thresholds so fewer assumed matches are generated and fewer potential duplicate records are found. In addition, you can run in optimistic mode rather than pessimistic mode to avoid unnecessarily re-evaluating potential duplicate records. Set the **SameSystemMatch** parameter to “false” to avoid assumed matches for records that were generated from the same system.

1.4 Reports

After this initial load runs, run each production report to help analyze the results. These reports can help you handle potential duplicates, identify data issues, and determine how to set your match and duplicate thresholds. When you run the reports against initial load data, set the **max-result-size** element for each report very high in the report configuration file. This ensures that you will capture all potential duplicate issues. Once the data has been loaded and analyzed, you can set the **max-result-size** element lower for performance.

Glossary

alphanumeric search

A type of search that looks for records that precisely match the specified criteria. This type of search does not allow for misspellings or data entry errors, but does allow the use of wildcard characters.

assumed match

When the matching weight between two records is at or above a weight you specify and the records are from two different systems, (depending on the configuration of matching parameters) the objects are considered an assumed match and are automatically combined.

Blocking Query

Also known as a blocker query, this is the query used during matching to search the database for possible matches to a new or updated record. This query makes multiple passes against the database using different combinations of criteria, which are defined in the Candidate Select file. This query can also be used for searches performed from the Enterprise Data Manager.

Candidate Select file

The eIndex SPV configuration file that defines the queries you can perform from the Enterprise Data Manager (EDM) and the queries that are performed for matching.

candidate selection

The process of performing the blocking query for match processing. See *Blocking Query*.

candidate selection pool

The group of possible matching records that are returned by the blocking query. These records are weighed against the new or updated record to determine the probability of a match.

checksum

A value added to the end of an EUID for validation purposes. The checksum for each EUID is derived from a specific mathematical formula.

code list

A list of values in the `sbyn_common_detail` database table that is used to populate values in the drop-down lists of the EDM.

code list type

A category of code list values, such as states or country codes. These are defined in the `sbyn_common_header` database table.

duplicate threshold

The matching probability weight at or above which two records are considered to potentially represent the same person. See also *matching threshold*.

EDM

See *Enterprise Data Manager*.

Enterprise Data Manager

Also known as the EDM, this is the web-based interface that allows monitoring and manual control of the master index database. The configuration of the EDM is stored in the Enterprise Data Manager file in the eIndex Project.

enterprise object

A complete object representing a specific entity, including the SBR and all associated system objects.

ePath

A definition of the location of a field in an eIndex SPV object. Also known as the *element path*.

EUID

The enterprise-wide unique identification number assigned to each member profile in the master index. This number is used to cross-reference member profiles and to uniquely identify each member throughout your organization.

eIndex SPV Manager Service

An eIndex SPV component that provides an interface to all eIndex SPV components and includes the primary functions of eIndex. This component is configured by the Threshold file.

field IDs

An identifier for each field that is defined in the standardization engine and referenced from the Match Field file.

Field Validator

An eIndex SPV component that specifies the Java classes containing field validation logic for incoming data. This component is configured by the Field Validation file.

Field Validation file

The eIndex SPV configuration file that specifies any default or custom Java classes that perform field validations when data is processed.

LID

See *local ID*.

local ID

A unique identification code assigned to a member in a specific local system. A member profile may have several local IDs in different systems. The combination of a local ID and system constitutes a unique identifier for a system record. The name of the

local ID field is configurable on the EDM, and might have been modified for your implementation.

master person index

A database application that stores and cross-references information about the members in a business organization, regardless of the computer system from which the information originates.

Match Field File

An eIndex SPV configuration file that defines normalization, parsing, phonetic encoding, and the match string for an instance of eIndex SPV. The information in this file is dependent on the type of data being standardized and matched.

match pass

During matching several queries are performed in turn against the database to retrieve a set of possible matches to an incoming record. Each query execution is called a match pass.

match string

The data string that is sent to the match engine for probabilistic weighting. This string is defined by the match system object defined in the Match Field file and must match the string defined in the match engine configuration files.

match type

An indicator specified in the **MatchingConfig** section of the Match Field file that tells the match engine which rules in the match configuration file to use for determine matching weights between records.

matching probability weight

An indicator of how closely two records match one another. The weight is generated using matching algorithm logic, and is used to determine whether two records represent the same member. See also *duplicate threshold* and *matching threshold*.

Matching Service

An eIndex SPV component that defines the matching process. This component is configured by the Match Field file.

matching threshold

The lowest matching probability weight at which two records can be considered a match of one another. See also *duplicate threshold* and *matching probability weight*.

matching weight or match weight

See *matching probability weight*.

member

Any person who participates within your business enterprise. A member could be a customer, employee, patient, and so on.

member profile

A set of information that describes characteristics of one member. A profile includes demographic and identification information about a member and contains a single best record and one or more system records.

merge

To join two member profiles or system records that represent the same person into one member profile.

merged profile

See *non-surviving profile*.

non-surviving profile

A member profile that is no longer active because it has been merged into another member profile. Also called a *merged profile*.

normalization

A component of the standardization process by which the value of a field is converted to a standard version, such as changing a nickname to a common name.

object

A component of a member profile, such as a person object, which contains all of the demographic data about a person, or an address object, which contains information about a specific address type for a person.

parsing

A component of the standardization process by which a freeform text field is separated into its individual components, such as separating a street address field into house number, street name, and street type fields.

phonetic encoding

A standardization process by which the value of a field is converted to its phonetic version.

phonetic search

A search that returns phonetic variations of the entered search criteria, allowing room for misspellings and typographic errors.

potential duplicates

Two different enterprise objects that have a high probability of representing the same entity. The probability is determined using matching algorithm logic.

probabilistic weighting

A process during which two records are compared for similarities and differences, and a matching probability weight is assigned based on the fields in the match string. The higher the weight, the higher the likelihood that two records match.

probability weight

See *matching probability weight*.

Query Builder

An eIndex SPV component that defines how queries are processed. The user-configured logic for this component is contained in the Candidate Select file.

SBR

See *single best record*.

single best record

Also known as the SBR, this is the best representation of a member's information. The SBR is populated with information from all source systems based on the survivor strategies defined for each field and child object. It is a part of a member's enterprise object and is recalculated each time a system record is updated.

standardization

The process of parsing, normalizing, or phonetically encoding data in an incoming or updated record. Also see *normalization*, *parsing*, and *phonetic encoding*.

survivor calculator

The logic that determines which field values or child objects from the available source systems are used to populate the SBR. This logic is a combination of Java classes and user-configured logic contained in the Best Record file.

survivorship

Refers to the logic that determines which field values are used to populate the SBR. The survivor calculator defines survivorship.

system

A computer application within your company where information is entered about the members in eIndex and that shares this information with eIndex (such as a registration system). Also known as a source system, local system, or external system.

system object

A record received from a local system. The fields contained in system objects are used in combination to populate the SBR. The system objects for one person are part of that person's enterprise object.

tab

A heading on an application window that, when clicked, displays a different type of information. For example, click the Create System Record tab to display the Create System Record page.

Threshold file

An eIndex SPV configuration file that specifies duplicate and match thresholds, EUID generator parameters, and which blocking query defined in the Candidate Select file to use for matching.

transaction history

A stored history of an enterprise object. This history displays changes made to the object's information as well as merges, unmerges, and so on.

Update Manager

The component of the master index that contains the Java classes and logic that determines how records are updated and how the SBR is populated. The user-configured logic for this component is contained in the Best Record file.

Index

A

analysis 29
 application file
 in an eIndex Project 81
 application server 23
 Application Servers 119
 audit log 54
 Automap 105, 106

B

backup
 repository 129
 basic searches 53
 Best Record file 20, 21, 29, 51, 52, 54, 55
 block picker, customizing 61
 blocking queries 51, 53, 55
 blocking query 138
 Business Process methods 64
 Business Processes 18, 93
 in a client Project 82
 including eIndex methods 94–95

C

Candidate Select file 20, 30, 51, 53, 138
 change management 137
 child objects 29, 53
 client Projects 22
 Connectivity Map 81
 Environments 119
 Code List script 21, 67
 modifying 75
 Collaboration
 processing from JMS Topic 92–93
 Collaborations 30
 in a client Project 81
 common table data 70
 defining 75
 components
 eIndex Project 19
 eIndex Repository 18
 Environment 23
 runtime 25–28

configuration
 Candidate Select file 138
 Object Definition 138
 connectivity components 22, 80–82
 in a client Project 81
 in an eIndex Project 81
 Connectivity Map 31
 adding JMS Topic to client Project 88–92
 adding to an eIndex Project 83–86
 adding to an eInsight Project 93
 External System Projects 86
 in a client Project 81
 in an eIndex Project 81
 in an eInsight Project 95
 linking components 85, 92
 Constants, environment 120
 Create database script 21, 67
 running 78
 Create User Code Data 21
 Create User Code Data script 67
 Create User Indexes 21
 Create User Indexes script 67
 cross-reference 23
 custom database scripts 77
 Custom Plug-ins 21, 64
 about 56–62
 creating 62

D

data analysis 69
 data structure 12, 17, 50, 53
 database
 creating 30
 designing 69
 factors 70
 hardware requirements 68
 indexes 71, 72
 installation 34
 operating systems 68
 optimization 69
 platforms 68
 requirements 67
 structure 69
 database connection pool 22, 34, 81, 82, 83, 120, 125, 126
 Database Script node 67
 database scripts
 Code List 21, 67
 Create database 21, 67
 Create User Code Data 21, 67
 Create User Indexes 21, 67
 custom 77
 Drop database 21, 67

- Drop User Indexes 21, 67
 - modifying 72–77
 - running 77–78
 - Systems 21, 67
- database tables
 - dropping 79
- Deployment Profile 23, 102
 - about 99
 - activating 102
 - for eInsight Projects 111–113
 - for External System Projects 107–110
 - for the server Project 103–107
 - mapping eInsight components 112
 - mapping External System components 109
 - mapping the server Project 105
- domainmgr.bat script 100
- Drop database script 21, 67
 - running 79
- Drop User Indexes 21
- Drop User Indexes script 67
- drop-down lists 70
- duplicate threshold 53

E

- editors
 - Java source 19
 - text 19
 - XML 19
- eGate Integrator 18
- eIndex
 - installing in Enterprise Designer 43
 - runtime components 25–28
- eIndex application
 - in a client Project 81
- eIndex Manager Service 20, 27, 53
- eIndex methods
 - in Business Processes 94–95
 - in Collaborations 87
- eIndex Projects
 - components 19
 - Connectivity Map 81
 - Environments 119
- eIndex Repository
 - components 18
- eInsight 64
 - Business Processes 94–95
 - Connectivity Map 81, 93
 - integration 18
 - Java methods for 22
- eInsight Project
 - Connectivity Map 95
 - JMS Topic 95
- element path
 - See ePath
- enterprise create policy 57
- Enterprise Data Manager
 - configuration 30
- Enterprise Data Manager file 20, 28, 30, 51, 54
- Enterprise Designer 18
 - Projects 19
- Enterprise Manager Monitor 131
- enterprise merge policy 57
- enterprise record 28
- enterprise unmerge policy 57
- enterprise update policy 57
- EnterpriseCreatePolicy element 57
- EnterpriseMergePolicy element 57
- EnterpriseUnmergePolicy element 57
- EnterpriseUpdatePolicy element 57
- Environment
 - adding a logical host 121
 - creating 120
- Environment components 23
- Environments
 - adding an external system 124
 - adding an Oracle external system 125
 - components 119–120
 - Constants 120
- ePath
 - about 140
- EUID 25, 53
 - configuration 52
 - generator 52
 - modifying the first 79
- eView.Application 105
- eView.Web.Application 105
- eVision External Systems 120
- eVision Studio
 - Java methods for 22
- eWays
 - in a client Project 82
- ExecuteMatchLogics 59
- External Applications
 - in a client Project 82
- External Systems 23
 - in eIndex Environments 120
 - method OTD for 22
- external systems
 - JMS Topic 88–92

F

- field locations
 - defining 140
- field names
 - syntax 140, 142, 143
- Field Validation file 21, 52, 54

field validations, defining 58
fields
 parsing 51
 phonetic 51
 standardized 51
fully qualified field names 142

G

Generate Project Files
 results 64
generating application files 64

I

identification 23
indexes 71, 72
installation
 overview 33
Integration Server
 adding 122
 configuring 122
Integration Servers 119

J

JAR files 64
Java API 13
Java methods, dynamic 22
Java source editor 19
java.util.regex 74
JMS Client
 in a client Project 82
JMS IQ Manager
 configuring 123
JMS IQ Managers 23, 120
JMS Queues
 in a client Project 82
JMS Topic
 in an eIndex Topic 81
 in an eInsight Project 95
JMS Topics
 in a client Project 82

L

local identifiers
 format 74
 length 74
Logical Host 23
Logical Hosts 119

M

master index application
 generating 64
match engine 20, 51, 53, 55
Match Engine node 22
match engine, customizing 61
Match Field file 20, 51, 53, 55
match threshold 53
matching 53
matching algorithm 13
matching configuration 51
Matching Service 20, 27, 53
method OTD 22, 51, 64, 86
monitoring
 setting up 132

O

Object Definition 138
Object Definition file 20, 50, 53, 55
Object Persistence Service 28
object structure 13, 22, 51, 53
Object Type Definition 22, 30
 in a client Project 81
Oracle eWay 81
Oracle External Systems 120
Outbound OTD 64

P

parent objects 29, 53
pass controller, customizing 61
performance optimization, database 69
phonetic encoders, customizing 62
process 29
processing codes 70, 73
Project components
 Custom Plug-ins 21
 Deployment Profile 23
 for connectivity 22
 Match Engine node 22
 outbound OTD 22
 Standardization Engine node 22
Projects
 client 22
Properties of Database Script 77–78

Q

qualified field names
 fully qualified 142
 qualified 142
queries

- basic 53
- blocking 53
- Query Builder 20, 27, 53
- query builder, customizing 60
- query definitions 51
- Query Manager 27

R

- Readme.txt file
 - up-to-date OS requirements
 - Windows Server 2003, Windows 2000/XP 35
- records, objects in 29
- relationships 50
- repository backup 129
- requirements 35
 - database 67
- runtime environment
 - features 24–25
 - functions 23
 - overview 23

S

- SBR
 - see single best record
- sbyn_common_detail 75
- sbyn_common_header 75
- sbyn_seq_table 79
- screenshots 15
- search definitions 51
- searches
 - basic 53
 - blocking 53
- Security
 - defining 114–117
 - file 21
- security permissions 116
- SeeBeyond Match Engine 13
 - configuration files 22
- Service
 - in a client Project 81
- Services 23
- simple field names 143
- single best record 13, 27, 29, 54
- standardization 53
- standardization configuration 51
- standardization engine 20, 51
- Standardization Engine node 22
- standardization engine, customizing 62
- Sun Java System Application Server 123
 - configuring 123
- Sun Java System JMS Server
 - adding 123

- configuring 124
- Sun SeeBeyond Integration Server 33
- Sun SeeBeyond Match Engine
 - configuration files 55
- survivor calculator 13, 20, 27, 51, 54, 55
- survivor calculator, customizing 60
- survivor strategy 27
- SurvivorHelperConfig 51, 54
- system codes 70
- system merge policy 57
- system records 28
- system unmerge policy 58
- SystemMergePolicy element 58
- systems
 - status 73
- Systems script 21, 67
- systems table description 73–74
- SystemUnmergePolicy element 58

T

- text editor 19, 55
- Threshold file 20, 51
- transaction history 24

U

- undo assumed match policy 58
- UndoAssumeMatchPolicy element 58
- Update Manager 20, 27
- update policies 21, 52
 - implementing 56–58
- user accounts 114–117
- user code data 70
- user groups 116

W

- Web application file 81
- Web Connectors 23
 - in a client Project 82
- WeightedCalculator 51, 54

X

- XML editor 19