SeeBeyond ICAN Suite

eIndex Global Identifier User's Guide

Release 5.0

Version 20031014191034.

Contents

List of Figures

Chapter 1

Introduction

This guide provides comprehensive information on installing and working with the SeeBeyond® eIndex Global Identifier (eIndex). As a component of SeeBeyond's Integrated Composite Application Network (ICAN) Suite, eIndex helps you integrate information from disparate systems throughout your organization. This guide explains how to install and customize the components of eindex, including eGate project files, the eIndex database, the runtime environment, and the Enterprise Data Manager (EDM). It also includes information about the architecture and components of eIndex. This guide is intended to be used with the *eIndex Global Identifier Configuration Guide* and *Implementing the SeeBeyond Match Engine with eIndex* or *Implementing Ascential INTEGRITY with eIndex*.

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.

1.1 Document Purpose and Scope

This guide provides step-by-step instructions for installing eIndex 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 components. These topics are covered in the appropriate user guide (for more information, see **"Supporting Documents" on page 14**).

1.1.1. Intended Audience

Any user who installs any component of eIndex, or customizes any of the components, should read this guide. A thorough knowledge of eIndex 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 eIndex 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.

1.1.2. Using this Guide

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

1.1.3. Document Organization

This guide is divided into ten chapters and one appendix that cover the topics shown below.

- **Chapter 1 "Introduction"** gives a general preview of this document—its purpose, scope, and organization—and provides sources of additional information.
- **Chapter 2 "eIndex Global Identifier Overview"** provides information about the architecture of eIndex and the runtime environment, and describes how the runtime environment is created.
- **Chapter 3 "Installation"** gives instructions for installing the eIndex files and setting up the environment for eIndex and the runtime environment.
- **Chapter 4 "Configuring eIndex"** gives a summary of the configuration files in the eIndex Project, and provides information about the XML Editor.
- **Chapter 5 "Generating the Project"** explains how to generate the eIndex application to update the application files with your customizations.
- **Chapter 6 "Creating Custom Plug-ins"** describes how to implement custom processing code in eIndex.
- **Chapter 7 "Creating the Database"** describes how to design, install, and configure the Oracle database for eIndex.
- **Chapter 8 "Defining Connectivity Components"** describes how to work with the connectivity components of the eIndex Project to share and process data through the eIndex system.
- **Chapter 9 "Defining the Environment"** describes how to set up the physical environment of eIndex.
- **Chapter 10 "Deploying the Project"** explains how to create the Deployment Profile for eIndex and how to activate the profile.
- **Appendix A "Field Notations"** describes the different notations used in the configuration files to define different fields in the messages being processed and stored.

1.2 Writing Conventions

Before you start using this guide, it is important to understand the special notation and mouse conventions observed throughout this document.

1.2.1. Special Notation Conventions

The following special notation conventions are used in this document.

Table 1 Special Notation Conventions

Text	Convention	Example
Titles of publications	Title caps in *italic* font	*eIndex Global Identifier Configuration Guide*
Button, Icon, Command, Function, and Menu Names	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**.
Parameter, Variable, and Method Names	**Bold** text	▪ Use the **executeMatch()** method. ▪ Enter the **field-type** value.
Command Line Code and Code Samples	`Courier` font (variables are shown in ***`bold italic`***)	▪ `bootstrap -p password` ▪ `<tag>Person</tag>`
Hypertext Links	Blue text	For more information, see **"Writing Conventions" on page 12**.
File Names and Paths	**Bold** text	To install eIndex, upload the **eIndex.sar** file.
Notes	***Bold Italic*** text	***Note:*** *If a toolbar button is dimmed, you cannot use it with the selected component.*

Additional Conventions

Windows Systems—The eIndex system is fully compliant with Windows NT, Windows 2000, and Windows XP platforms. When this document refers to Windows, such statements apply to all three Windows platforms.

UNIX Systems—This guide uses the backslash (\ ) as the separator within path names. If you are working on a UNIX system, please make the appropriate substitutions.

1.2.2. Mouse Conventions

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

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

- **Point** means to position the mouse pointer until the tip of the pointer rests on whatever you want to point to on the screen.
- **Click** means to press and then immediately release the left mouse button without moving the mouse.
- **Double-click** means to click the left mouse button twice in rapid succession.
- **Right-click** means to click the right mouse button once.

- **Drag** means to point and then hold down the mouse button as you move the mouse. **Drop** means to let go of the mouse button to place the dragged information where you want it to be moved.
- **Move** means to point to an object on the screen and then drag the mouse to move the object to a screen location of your choice.
- **Highlight** means to select an area of text by dragging the mouse over the desired portion of text that appears on a window.
- **Select** means to point to a list of information on an eIndex window, and then click once to choose the data you want. The information becomes highlighted when selected.
- **Expand** means to double-click a row of information on an expandable list to display more details. The details appear on another row, below the row you double-click.
- **Collapse** means to double-click a row of information on an expandable list to hide the details that appear on the following row.

1.3 Supporting Documents

SeeBeyond has developed a suite of user's guides and related publications that are distributed in an electronic library. The following documents may provide information useful in creating your customized index. In addition, complete documentation of the eIndex Java API is provided in Javadoc format.

- *eIndex Enterprise Data Manager User's Guide*
- *eIndex Global Identifier Configuration Guide*
- *eIndex Global Identifier Reference Guide*
- *Implementing the SeeBeyond Match Engine with eIndex*
- *Implementing Ascential INTEGRITY with eIndex*
- *eGate Integrator User's Guide*
- *SeeBeyond ICAN Suite Deployment Guide*
- *eVision Studio User's Guide*
- *eInsight Business Process Manager User's Guide*

1.4 Online Documents

The documentation for the SeeBeyond ICAN Suite is distributed as a collection of online documents. These documents are viewable with the Acrobat Reader application from Adobe Systems. Acrobat Reader can be downloaded from:

http://www.adobe.com

1.5 SeeBeyond Web Site

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

http://www.SeeBeyond.com

Chapter 2

eIndex Global Identifier Overview

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

2.1 Learning about eIndex

2.1.1. Overview

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

In eIndex, 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.

2.1.2. eIndex Features and Functions

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

- **Rapid Development**—eIndex 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 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 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 implement a custom survivor strategy.
- **Flexible Architecture**—eIndex provides a flexible platform that allows you customize the object structure, allowing you to design an application that specifically meets your data processing needs.
- **Configurable Matching Algorithm**—eIndex provides support for both the SeeBeyond Match Engine and the Ascential™ INTEGRITY™ matching algorithm. In addition, you can plug in a custom matching algorithm.
- **Configurable Server Support**—The runtime eIndex application supports the SeeBeyond Integration Server (SIS).
- **Custom Java API**—eIndex generates a Java API that is customized to the object structure you define. You can call the methods in this API in the Collaborations that define the transformation rules for data inbound to eIndex.

2.1.3. eIndex and the SeeBeyond ICAN Suite

eIndex is tightly integrated within the ICAN suite, and can leverage the features of other ICAN suite components.

eGate Integrator

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

eInsight

eInsight Business Process Manager (eInsight) is the component within the SeeBeyond ICAN Suite that facilitates the automation of the flow of business activities. eInsight 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 Project in Business Processes for eInsight, enabling access to the eIndex database through eInsight.

eVision

eVision Studio (eVision) is a graphical design studio for the WYSIWYG creation of specialized Web applications, specifically developed for integration with the eGate and eInsight runtime environments. Using eVision, you can create custom Web pages to access information stored in the eIndex database.

2.2 eIndex Repository Components

eIndex has two types of components: Repository and runtime. The Repository components of eIndex are designed to work within the eGate Enterprise Designer to create and configure eIndex, and to define connectivity between external systems and eIndex. This section describes the Repository components; the runtime components are described in **"Runtime Environment Components" on page 25**.

The primary Repository components of eIndex are:

- Editors
- Project Components
- Environment Components

2.2.1. Editors

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

- **XML Editor**—allows you to review and customize the XML configuration files. This editor provides verification services for XML syntax (schema validation is provided through eIndex). The XML editor is automatically launched when you open an eIndex configuration file.
- **Text Editor**—allows you to review and customize the database scripts. 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 database script.
- **Java Source Editor**—allows you to create and customize custom plug-in classes for eIndex. This editor is a simple text editor, similar to the Java Source Editor in the Java Collaboration Editor. The Java source editor is automatically launched when you open a custom plug-in file.

2.2.2. Project Components

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

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

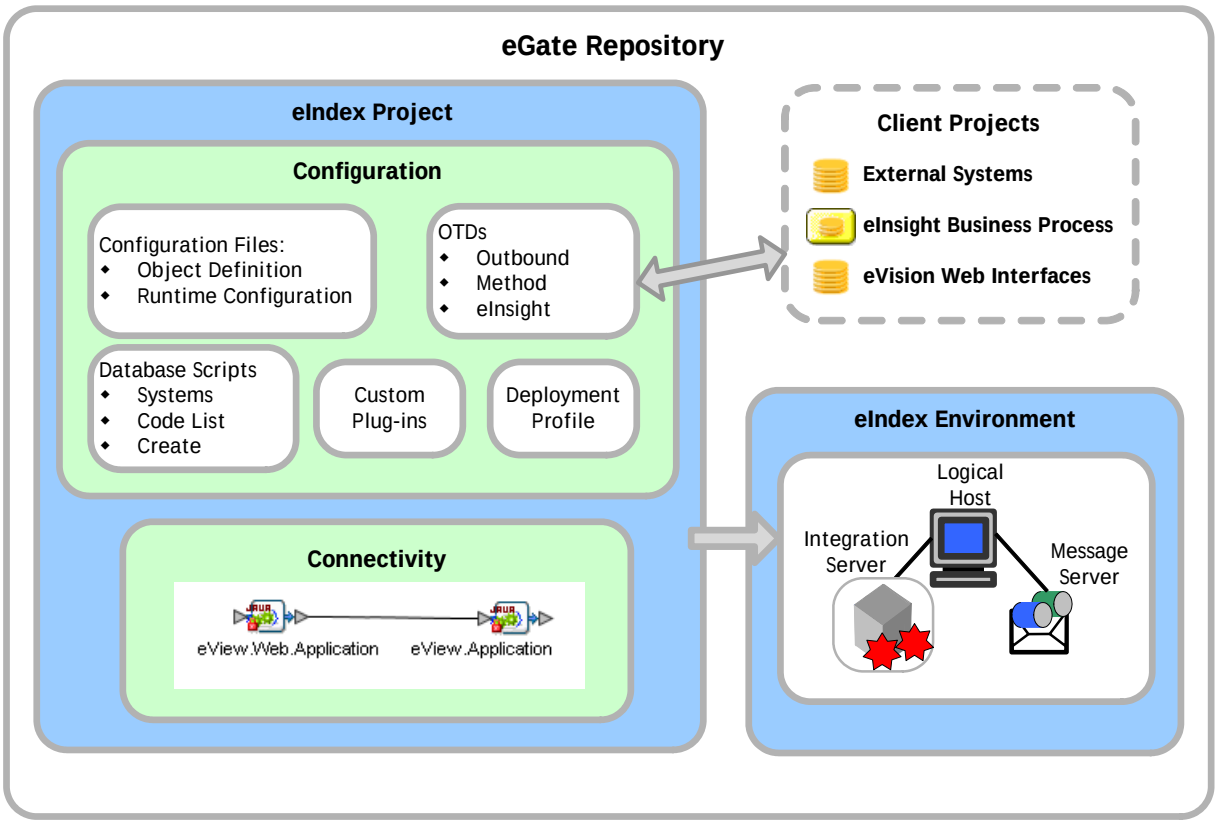
Following is a list of eIndex Project components.

- 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 19 illustrates the Project and Environment components of eIndex.

Figure 1 eIndex Project and Environment Components



Configuration Files

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

- **Object Definition**—This file defines the data structure of eIndex.
- **Enterprise Data Manager**—This file configures the search functions and appearance of the EDM, along with debug information and security information.

- **Candidate Select**—This file configures the Query Builder component of eIndex, and defines the available queries.
- **Match Field**—This file configures the Matching Service, and defines the fields to be standardized and the fields to use for matching. It also specifies the match and standardization engines to use.
- **Threshold**—This file configures the eIndex Manager Service, and defines certain system parameters, such as match thresholds, EUID attributes, and update modes. It also specifies the query from the Query Builder to use for matching queries.
- **Best Record**—This file configures the Update Manager and defines the strategies used by the survivor calculator to determine the field values for the 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 Project to allow you to create the eIndex table structure, indexes, and custom start-up data. Additional scripts are provided for testing purposes that drop the tables and indexes you created.

- **Systems**—Contains a sample SQL insert statement for adding information about an external system to the database. You can modify this script to define you 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 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 "insert" 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 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 Project and run against the database.

Custom Plug-ins

eIndex 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 eIndex by adding it to the appropriate configuration file. You can create custom update procedures and field validations. Update procedures must be referenced in the update policies of the Best Record file, and field validations must be referenced in the Field Validation file. Custom plug-ins can also be used to create custom eIndex components, such as a custom query builder or block picker.

Match Engine Configuration Files

If you are using the SeeBeyond Match Engine, several configuration files for the engine are stored in the eIndex 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 fields as necessary. For more information, refer to *Implementing the SeeBeyond Match Engine with eIndex*.

Outbound Object Type Definition (OTD)

eIndex 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 to the external systems that share data with eIndex. 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 data available to external systems, you must define a topic in the eIndex Connectivity Map to which eIndex can publish transactions.

Dynamic Java API

Due to the flexibility of the object structure, eIndex includes several dynamic Java methods for use in Collaborations and in the Web service. One set is provided in a method OTD for use in Collaborations and and one set is provided for Web services. 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 Global Identifier 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 on incoming records, and so on.

Web Services Java Methods

In addition to the method OTD, which can be used in Collaborations, eIndex includes a set of Java methods that can be incorporated into an eInsight Business Process for 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 eIndex database from eInsight Web pages.

Connectivity Components

The eIndex Project Connectivity Map only consists of two components: the web application file and the application file. However, in client Projects you can use any of the standard Project components to define connectivity and data flow for eIndex. Client Projects include those created for the external systems sharing data with the index and those created for eVision Web pages. The eIndex Connectivity Map may include one additional component, a JMS Topic, to which eIndex can publish all processed messages for broadcasting to external systems.

For the client Projects, you can use connectivity components from the eIndex server Project and create any standard eGate connectivity components, such as OTDs, Services, Collaborations, queues, topics, and eWays. Client Project components transform and route incoming data into the eIndex database according to the rules contained in the Collaborations. 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 of eIndex. It contains information about the assignment of Services and message destinations to integration servers and JMS IQ Managers within the eIndex system. Each eIndex 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 activate the deployment before you can use the eIndex runtime environment.

2.2.3. Environment Components

The eIndex Environments define configuration of the deployment environment of the runtime environment, including the Logical Host and application server. If eIndex client Projects use the same Environment, it may also include a JMS IQ Manager, constants, Web Connectors, and External Systems. Each Environment represents a unit of software that implements eIndex. You must define and configure at least one Environment for eIndex before you can deploy the application. The integration server hosting eIndex is configured within the Environment in the Enterprise Designer. Security is defined through the Environment configuration.

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

2.3 About the Runtime Environment

In today's business environment, important information about certain business objects in your organization may 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 can help you manage this data, and ensure that the data you have is the most current and accurate information available.

Regardless of how you define the data structure and configure the runtime environment for eIndex, 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.

2.3.1. Functions of the Runtime Environment

eIndex in the runtime environment 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 member by recording all changes to each member'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 member 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 member or a set of members. 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 system is its ability to match records and identify possible duplicates. Using matching algorithm logic, eIndex 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.

2.3.2. Runtime Environment Features

The components of the eIndex 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 person index suited to your specific data processing needs. Primary features of runtime environment include:

- **Centralized Information**—eIndex maintains a centralized database, enabling the integration of data records throughout the enterprise while allowing local systems to continue operating independently. eIndex stores copies of local system records and of SBRs, which represent the most accurate and complete data for each member. This database is the central location of information and identifiers, and is accessible throughout the enterprise.
- **Configurability**—Before deploying eIndex, 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, EDM appearance, and field validation rules.
- **Cross-referencing**—eIndex is a global cross-indexing application that automates record-matching across disparate source systems, simplifying the process of sharing data between systems. eIndex uses the local identifiers assigned by your existing systems as a reference for cross-indexing, allowing you to maintain your current systems and practices. eIndex maintains the most current information by providing accurate identification and cross-referencing of person records.
- **Data Cleansing**—eIndex uses configurable matching algorithm logic to uniquely identify object records, and to identify duplicate and potential duplicate records. eIndex provides the functionality to easily merge or resolve duplicates, and can be configured to automatically merge records that are found to be duplicates of one another.
- **Data Updates**—eIndex 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 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 members within an enterprise.
- **Integration**—Relying on the eGate Integrator, eIndex 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 any external system, providing seamless integration with the systems in your enterprise.
- **Matching Algorithm**—eIndex is designed to use the SeeBeyond Match Engine or Ascential's INTEGRITY matching algorithm to provide a matching probability weight between records. Both algorithms provide the flexibility to create user-defined matching thresholds, which control how potential duplicates and

automatic merges are determined. You can configure eIndex to use the match engine of your choice.

- **Shared Information**—Each time a record is updated, added, merged, or unmerged from the user interface, eIndex generates a message that can be transmitted to external systems. It also receives, processes, and broadcasts messages, containing information about the members in the database.
- **Unique Identifier**—Records from various systems are cross-referenced using an enterprise-wide unique identifier, known as an EUID, that eIndex assigns to each object record. eIndex uses the EUID to cross-reference the local IDs assigned to each object by the various computer systems throughout the enterprise.

2.4 Runtime Environment Components

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

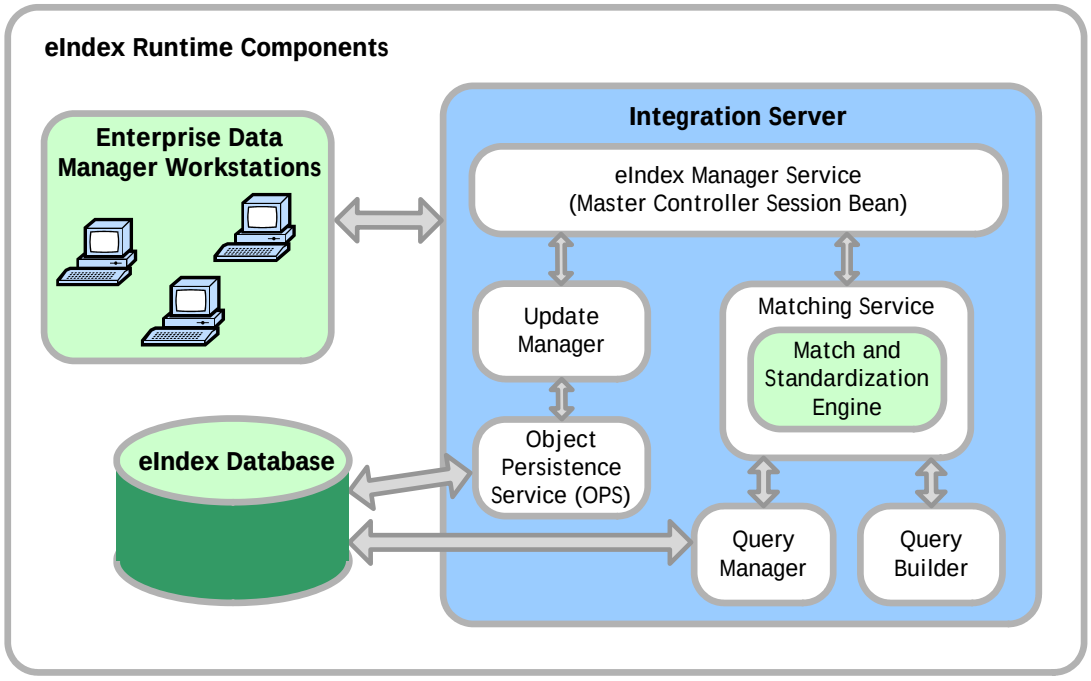
- eIndex Manager Service
- Matching Service
- Query Builder
- Query Manager
- Update Manager
- Object Persistence Service
- Database
- Enterprise Data Manager

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

The eGate Repository stores information about the configuration and structure of the runtime environment. Because eIndex is deployed within eGate, it can be implemented in a distributed environment. The eIndex system requires the SeeBeyond Integration Server to enable Web service connectivity.

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

Figure 2 eIndex Runtime Environment Architecture



2.4.1. 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, such as the configuration files for the SeeBeyond Match Engine or the rule set files for INTEGRITY. The configuration of the Matching Service is defined in the *Match Field* file.

2.4.2. 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 provides 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 Manager Service is defined in the *Threshold* file.

2.4.3. Query Builder

The Query Builder defines all queries available to. This includes the queries performed automatically by 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.

2.4.4. 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 are executed through the Query Manager.

2.4.5. Update Manager

The Update Manager controls how updates are made to a member'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 is populated 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.

2.4.6. 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.

2.4.7. Database

eIndex uses an Oracle database to store the types of information you specify for member records. The database stores local system records, the SBR for each member 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 eIndex database structure is based on the object structure defined in the Object Definition file.

2.4.8. 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 database. The configurable attributes of the EDM are defined in the Enterprise Data Manager file, which you can modify after you generate the eIndex application. The EDM provides the ability to manually search for records; update, add, deactivate, and reactivate records; merge and unmerge records; view potential duplicates; and view comparisons of member records.

2.5 Enterprise Records

An *enterprise record* includes all components of a record that represents one member. eIndex stores two different types of records in each enterprise record: *system records* and a *single best record* (SBR). A system record contains a member's information as it appears in an incoming message from a local system. An enterprise record's SBR stores data from a combination of local systems, and it represents the most reliable and current information contained in all system records for a member. An enterprise record consists of both system records and the SBR.

2.5.1. System Records

The structure of system records 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 SBR in that enterprise record. If an enterprise record only contains one system record, the SBR will be identical to that system record. However, if it contains multiple system records, the SBR may be identical to one system record but will more likely include a combination of information from all system records.

2.5.2. The Single Best Record

The SBR for a member is created from the most reliable information contained in each system record for a particular enterprise record. The information used from each local 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 a local 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.

2.5.3. Objects in an Enterprise Record

In eIndex, each system record and SBR in an enterprise record typically contain a set of objects that store different types of information about member. A record contains only one parent object and several 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 member 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 object types (*address*, *phone*, and *alias* objects respectively).

2.6 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 eIndex database and to share among the systems connected to eIndex. Once your analysis is complete, you can define the object structure and begin to customize the configuration files for your processing requirements.

2.6.1. Process Overview

The following steps outline the procedures you need to follow to build the runtime environment using the eIndex 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 (**Chapter 4**).

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. Generate the master index (**Chapter 5**).
4. Define and build custom plug-ins, and specify the plug-ins in the appropriate configuration file (**Chapter 6**). This step is optional.
5. Create the database (**Chapter 7**).
   - 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 (**Chapter 8**).
   - 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 (**Chapter 9**).
8. Create the Deployment Profile and activate the Project (**Chapter 10**).

2.6.2. From XML to the Database

The eIndex database is created using a standard Oracle database and a database script created from the Object Definition file. Running the database script against a standard Oracle 8.1.7 or 9i database creates the tables necessary for eIndex. 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.

2.6.3. From XML to the Enterprise Data Manager

The Enterprise Data Manager file 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 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.

2.6.4. From XML to the Connectivity Components

The eIndex Project contains several connectivity components, including a method OTD, eInsight methods, and an outbound OTD. All are created based on the Object Definition. The method OTD contains certain Java methods for use in Collaborations to specify how data is processed into the database. The eInsight methods are used for eInsight integration. The outbound OTD is used when publishing messages processed by eIndex for broadcasting to external systems. Generating a Project also creates application files that you can drag into the Connectivity Map.

2.6.5. From XML to the Runtime Environment

The information you specify in the eIndex configuration files is stored in the eGate Repository, and is read at runtime when the Logical Host is started. The only exception is the Object Definition, 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 Project and reactivate the Deployment profile 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 may result in loss of data integrity or unexpected weighting and matching results.

Chapter 3

Installation

eIndex Global Identifier is installed in an eGate environment, and is installed into the Enterprise Designer. This chapter provides instructions on installing eIndex once the eGate environment is in place.

3.1 Installation Overview

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

3.1.1. eIndex Installation

eIndex is installed by uploading the eIndex files to the eGate Repository using the Enterprise Manager, and then installing the eIndex Module and eIndex Help to the Enterprise Designer. eIndex must be uploaded via an active eGate Repository, and the eIndex Module and eIndex Help must be installed on a computer with an existing Enterprise Designer. You must have access to a web browser for the initial upload.

Before installing eIndex, make sure you have followed the instructions in the *eGate Integrator Installation Guide* to install your eGate environment, including the Repository, monitors, Logical Host, Enterprise Designer, and integration server. eIndex runs on the SeeBeyond Integration Server.

3.1.2. Database Installation

All of the eIndex components are stored in the eGate Repository. The only external component is the eIndex database. The database does not need to be installed in order to use the eIndex tools, but it must be installed as a part of eIndex implementation. For optimal performance, the database should be installed on its own server. The eIndex database can be installed on an Oracle 8.1.7 or 9*i* platform, and can run on any operating system platform supported by Oracle 8.1.7 or 9*i*.

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 Project. Running these scripts against the database creates the eIndex 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 installation is a multi-stage process that includes the following:

1. Uploading eIndex into the Repository.
2. Uploading INTEGRITY into the Repository (optional).
3. Installing eIndex in the Enterprise Designer.

You can also install these components:

- eIndex documentation in PDF format
- eIndex Javadocs

3.3 System Requirements

eIndex is installed within an eGate Integrator environment. For system requirement information for the eGate environment, see the *eGate Integrator Installation Guide*. This guide also contains information about resource considerations. In addition, the **Readme.txt** file (located in the Root directory of the installation CD-ROM) contains the most up-to-date operating system requirements for the supported platforms. The requirements listed below are in addition to the operating system requirements.

The following operating systems are supported by eIndex.

- Windows Server 2003, Windows XP SP1a, and Windows 2000 SP3
- HP Tru64 V5.1A with required patches
- HP-UX 11.0 and 11i with required patches and parameter changes
- IBM AIX 5.1 and 5.2 with required Maintenance level patches
- Red Hat Enterprise Linux AS 2.1
- Red Hat Linux 8 (Intel Version)
- Sun Solaris 8 and 9 with required patches

Important: *Linux is not supported for implementations using the Ascential INEGRITY matching algorithm.*

Requirements for the eIndex database are included in **Chapter 7**, **"Creating the Database"**. The client workstations accessing the eIndex EDM requires Internet Explorer 6.0 with SP1.

3.4 Requirements for the eGate Environment

You will be issued a license that allows you to install eIndex. eIndex can be installed after you have done the following:

- Installed the Repository.
- Installed Enterprise Manager, including these files:
  - A **ProductsManifest.xml**
  - B **egate.sar**.
  - C **fileeWay.sar**
- Installed the Enterprise Designer and all required modules, including:
  - A Code Generation Framework, and all modules required by the framework, such as Logical Host, Project Explorer, Environment Explorer, and so on
  - B Deployment Editor
  - C OTD, Collaboration, and Connectivity Map Editors
  - D DTD OTD Wizard

Note: *These modules are required for most Enterprise Designer implementations, and will most likely already be installed (unless you are installing Enterprise Designer for the first time).*

3.4.1. Before Installing eIndex

Before you install eIndex, you must do the following:

1. Obtain an activation key, which is required for every installation (see the *eGate Integrator Installation Guide* for more information).
2. Select the Window(s) computers that will host eIndex. This must be installed on a computer running the Enterprise Designer, which only runs on Windows systems.
3. Determine the add-on applications, if any, you will require.
4. Make sure you have the appropriate administrator permissions to install eIndex in the Enterprise Manager and to update the Enterprise Designer through the Update Center.
5. Make sure the File eWay is installed. The eIndex installation relies on the File eWay in order to create the start-up Project, and will fail if the File eWay is not installed first.
6. Before you begin the installation, exit all Windows applications.

3.5 Installing eIndex

To install eIndex, you must complete the following tasks:

- **Uploading eIndex to the Repository** on page 34
- **Installing eIndex in the Enterprise Designer** on page 40

3.5.1. Uploading eIndex to the Repository

The first step in installing eIndex is uploading the files into the eGate Repository. These files are in the form of a SeeBeyond archive file that contains all the actual components of the eIndex package and licensing information. Make sure you have installed all the necessary eGate components before beginning.

These steps are performed using the Enterprise Manager, which serves as an update center, management center, and a dash board to gain access to available applications. Additionally, system administrators use Enterprise Manager to upload components to the Repository server. Perform these steps to upload all eIndex files.

- **Uploading eIndex** on page 34
- **Uploading the eIndex Documentation** on page 37
- **Uploading the INTEGRITY Add-on** on page 38

Important: *Make sure eGate Integrator and Enterprise Designer are installed before performing these steps. This procedure automatically installs the eIndex server and client Projects in Enterprise Designer.*

Uploading eIndex

The first step is to upload the eIndex product files.

To upload eIndex

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

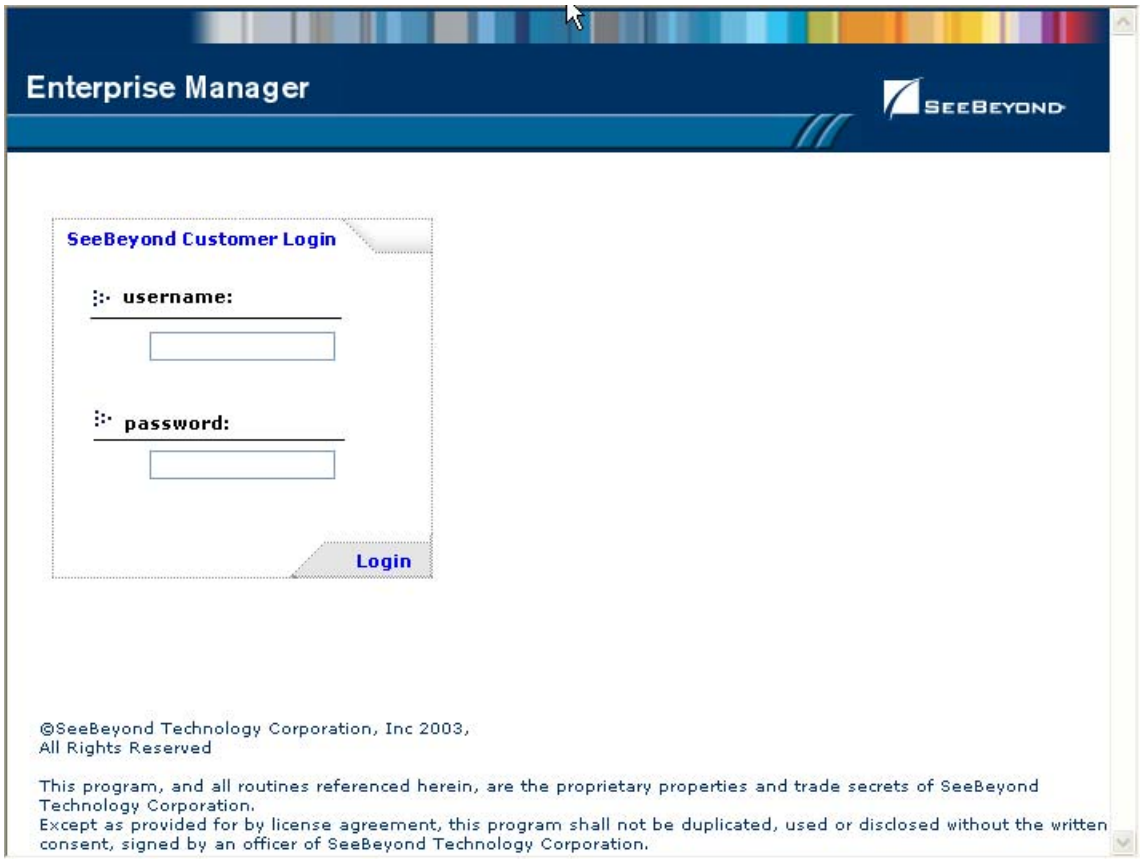
   <***hostname***> is the TCP/IP host name of the server where you installed the Repository—not the name of the Repository itself.

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

   When ready, press **Enter**.

   The **SeeBeyond Customer Login** window of Enterprise Manager appears (see Figure 3).

Figure 3 Enterprise Manager Logon Window



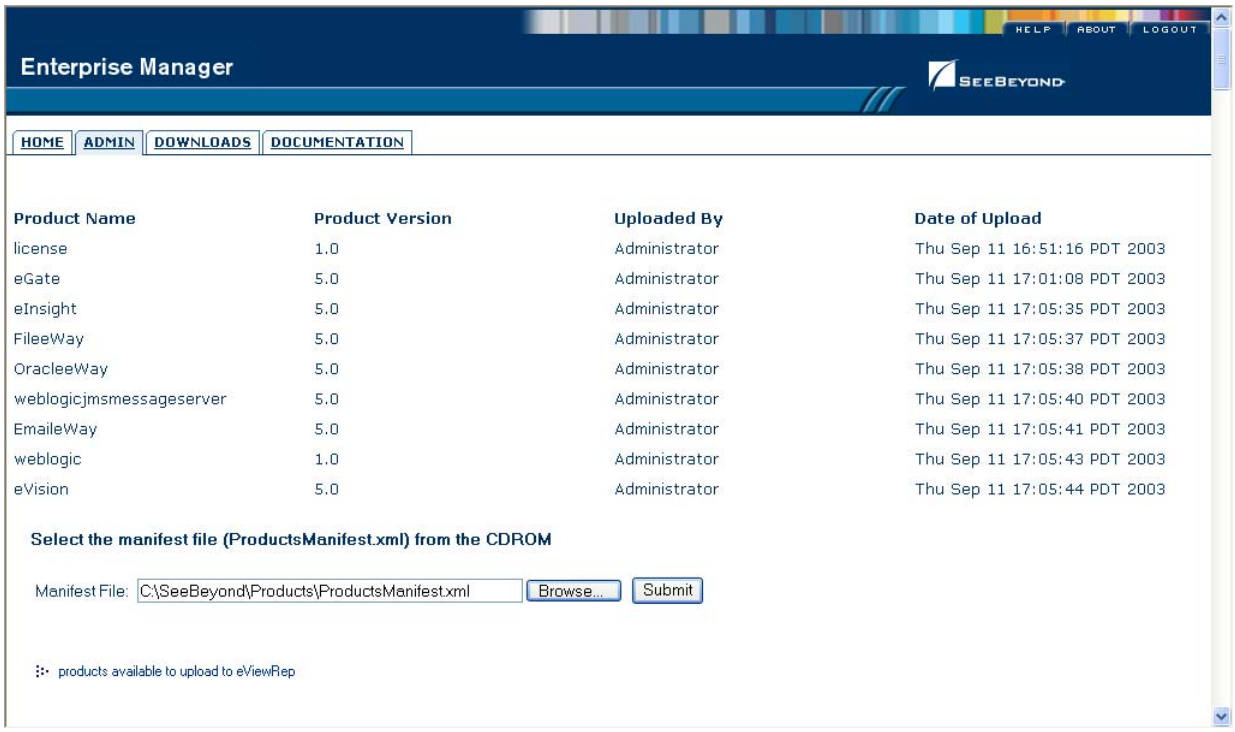
4 Enter your **username** and **password**.

5 When ready, click **Login**.

The **Upload System Component Manifest** window appears with the **HOME** tab active.

6 Click the **ADMIN** tab (see Figure 4).

Figure 4 ADMIN Page



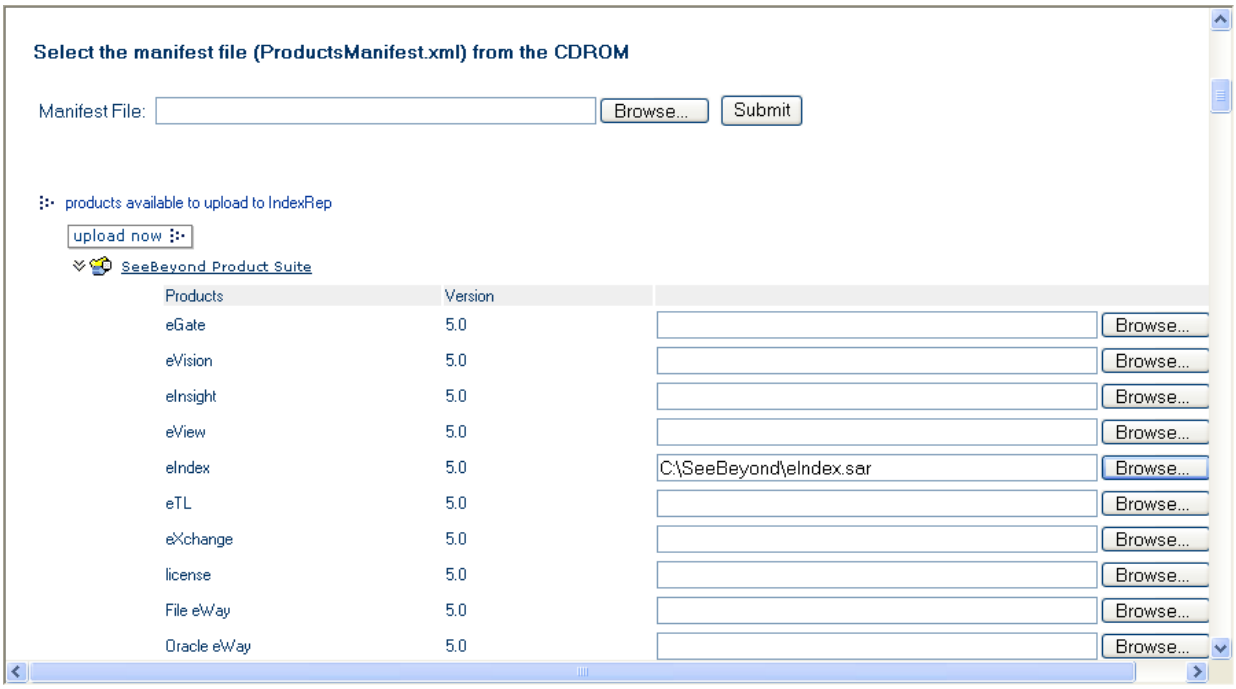
7 On the **ADMIN** page, do the following:

A Enter the products manifest file for eIndex (**ProductsManifest.xml**), located in the **Products** folder on the installation CD-ROM, or click **Browse** to navigate to **ProductsManifest.xml**, select the file, and then click **Open**.

B On the **ADMIN** page, click **Submit**.

A list of products you can install appears in the **products available to upload** section of the page.

Figure 5 Products Available to Upload Section



8 To upload the eIndex files to the Repository, under **SeeBeyond Product Suite**, click **Browse** (for eIndex).

A Navigate to the **Products** folder on the installation CD-ROM, select **eIndex.sar**, and then click **Open**.

B When you return to the **products available to upload to <*Repository_name*>** window, the **Products** box for eIndex is populated. Click **upload now**.

9 If you are installing documentation or the INTEGRITY add-on, leave the Enterprise Manager open and continue to **"Uploading the eIndex Documentation" on page 37** or **"Uploading the INTEGRITY Add-on" on page 38**.

Uploading the eIndex Documentation

This optional step uploads a complete eIndex documentation set, including Javadocs. It creates links to these components on the Enterprise Manager's DOCUMENTATION tab.

To upload the eIndex documentation

1 Complete the procedure described under **"Uploading eIndex" on page 34**.

2 On the Enterprise Manager, click the **ADMIN** tab.

3 To upload documentation files to the Repository, under **SeeBeyond Product Suite**, click **Browse** (for eIndex).

A Navigate to the **Products** folder on the installation CD-ROM, select **eIndexDoc.sar**, and then click **Open**.

B When you return to the **products available to upload to <*Repository_name*>** window, the **Products** box for eIndex is populated. Click **upload now**.

Note: *You can install additional documentation if needed. The* ***.sar*** *files for non-eWay documentation are located in the* ***\Documentation*** *directory on the Products - Disc 1 CD-ROM. The* ***.sar*** *files for eWay documentation are located in the* ***\Documentation*** *directory on the Products - Disc 2 CD-ROM.*

4 Leave the Enterprise Manager open and perform any of the following procedures:

- **To access the documentation files** on page 38
- **To download the Javadoc files** on page 38
- **To upload the INTEGRITY Add-on** on page 38

To access the documentation files

1 In the Enterprise Manager, click the DOCUMENTATION tab.

2 Click **eIndex Global Identifier**.

3 In the frame on the right side of the window, select any document title to view the file in Acrobat Reader.

To download the Javadoc files

1 In the Enterprise Manager, click the DOCUMENTATION tab.

2 Click **eIndex Global Identifier**.

3 In the frame on the right side of the windows, scroll to, and then click, **Download Javadoc**.

4 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 files.

5 To access the documents, navigate to the extract directory and then to **\eView_Javadocs\html**.

6 Double-click **index.html**. This page provides links to all Javadoc pages.

Uploading the INTEGRITY Add-on

This step is required only if you are using eIndex with the INTEGRITY match engine.

To upload the INTEGRITY Add-on

1 Complete the procedure described under **"Uploading eIndex" on page 34**.

2 On the Enterprise Manager, click the **ADMIN** tab.

3 To upload the INTEGRITY files to the Repository, under **SeeBeyond Product Suite**, click **Browse** (for eIndex).

A Insert the eView INTEGRITY Add-on CD-ROM.

B Navigate to the eView INTEGRITY Add-on CD-ROM, select **eViewIntegrityAddon.sar**, and then click **Open**.

Note: *The **.sar** files for the INTEGRITY add-on are located on the INTEGRITY Add-on CD-ROM. The **.sar** files for INTEGRITY documentation are located on the same CD-ROM.*

C When you return to the **products available to upload to <*Repository_name*>** window, the **Products** box for eIndex is populated. Click **upload now**.

4 To upload the INTEGRITY documents to the Repository, under **SeeBeyond Product Suite**, click **Browse** (for eIndex).

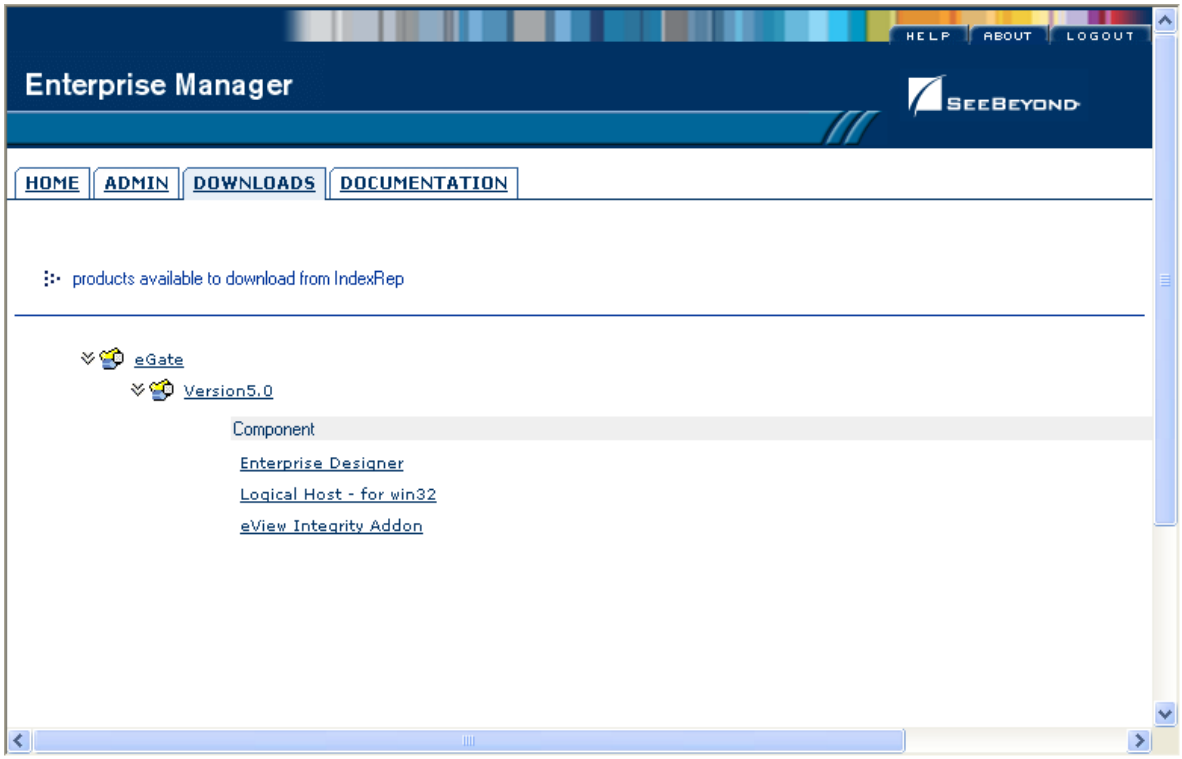
A Navigate to the eView INTEGRITY Add-on CD-ROM, select **eViewIntegrityAddonDoc.sar**, and then click **Open**.

B When you return to the **products available to upload to <*Repository_name*>** window, the **Products** box for eIndex is populated. Click **upload now**.

C The documents can be accessed from the DOCUMENTATION tab under "Add-ons".

5 When the upload is complete, click the **DOWNLOADS** tab. The **products available to download from <*Repository_name*>** window appears (see Figure 6).

Figure 6 DOWNLOADS Page (for INTEGRITY Addon)



6 Click **eView Integrity Addon**. You can either open the file and extract the INTEGRITY files, or you can save the file to disk and extract the files at a later time.

Note: *The extracted files must reside on the application or integration server for the eIndex system. Make sure to note the location to which you extract the files. You will need*

to set this as a variable in the server configuration in the Enterprise Designer, and you will need to copy one of the extracted files into the eIndex Project.

3.5.2. Installing eIndex in the Enterprise Designer

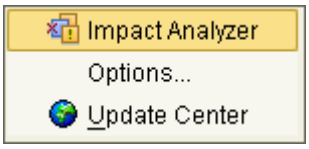
The final step in installing eIndex is updating the Enterprise Designer with eIndex. The Enterprise Designer must be installed on the machine on which you are installing eIndex. This step is performed using the Update Center, which is a tool in Enterprise Designer that allows you to install add-on modules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

Note: *The eIndex Project may be visible on the Enterprise Designer before you perform these steps. If it is 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 in the Enterprise Designer

1 Navigate to **<*c:\eGate50*>\edesigner\bin** and double-click **runed.bat**. The SeeBeyond Enterprise Designer GUI opens.

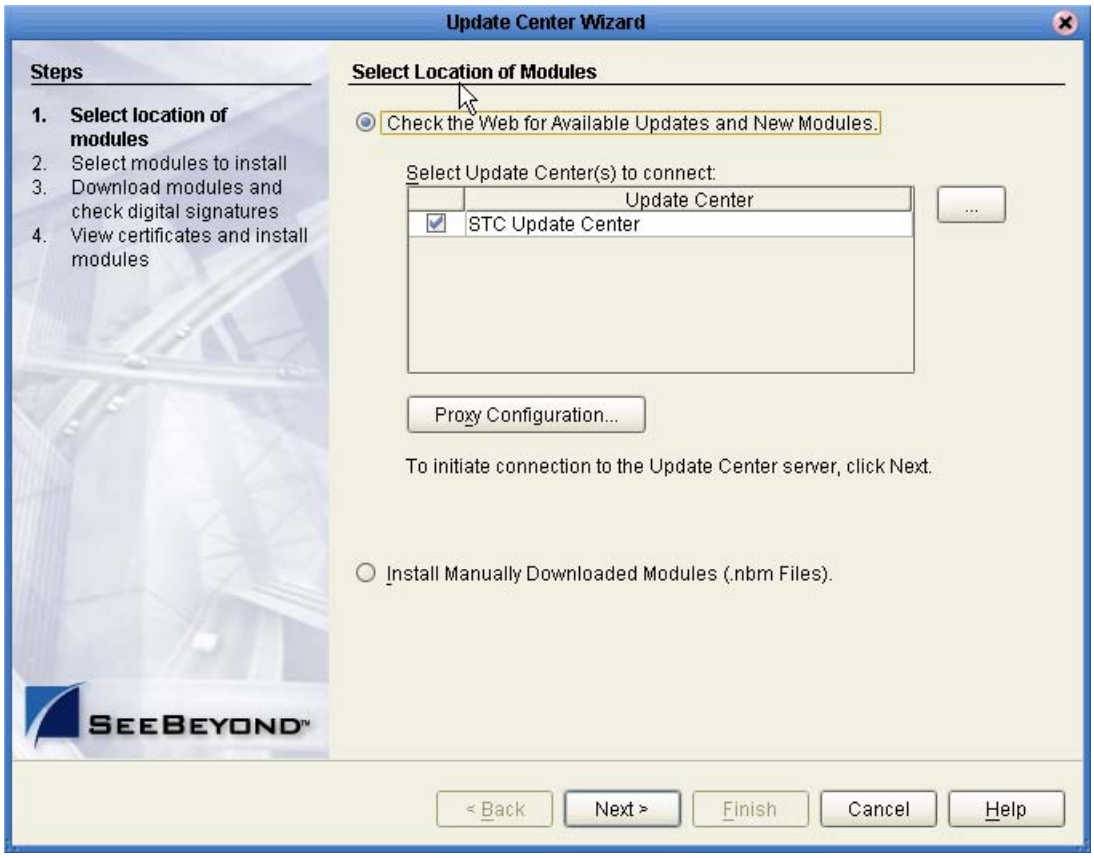
Figure 7 Tools Menu for Enterprise Designer



2 Select the **Tools** menu and click **Update Center**.

The **Update Center Wizard** appears (see Figure 8).

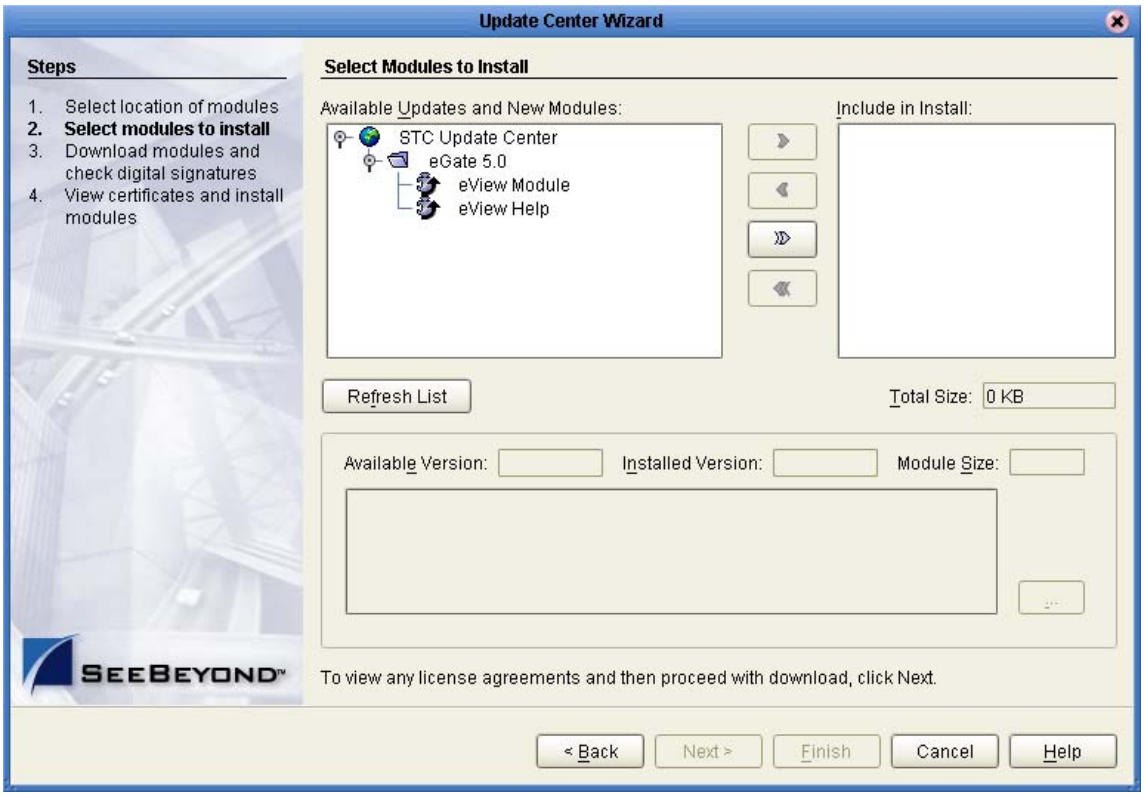
Figure 8 Update Center Wizard - Select Location of Modules



3 Accept the default values, and click **Next**.

The **Select Modules to Install** page appears.(see **Figure 9 on page 42**).

Figure 9 Update Center Wizard - Select Modules to Install



4 Do one of the following:

- In the **Available Updates and New Modules** box, select **eView Module** and **eView Help**, and then click the **Add** button (single-arrow button at the top).

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

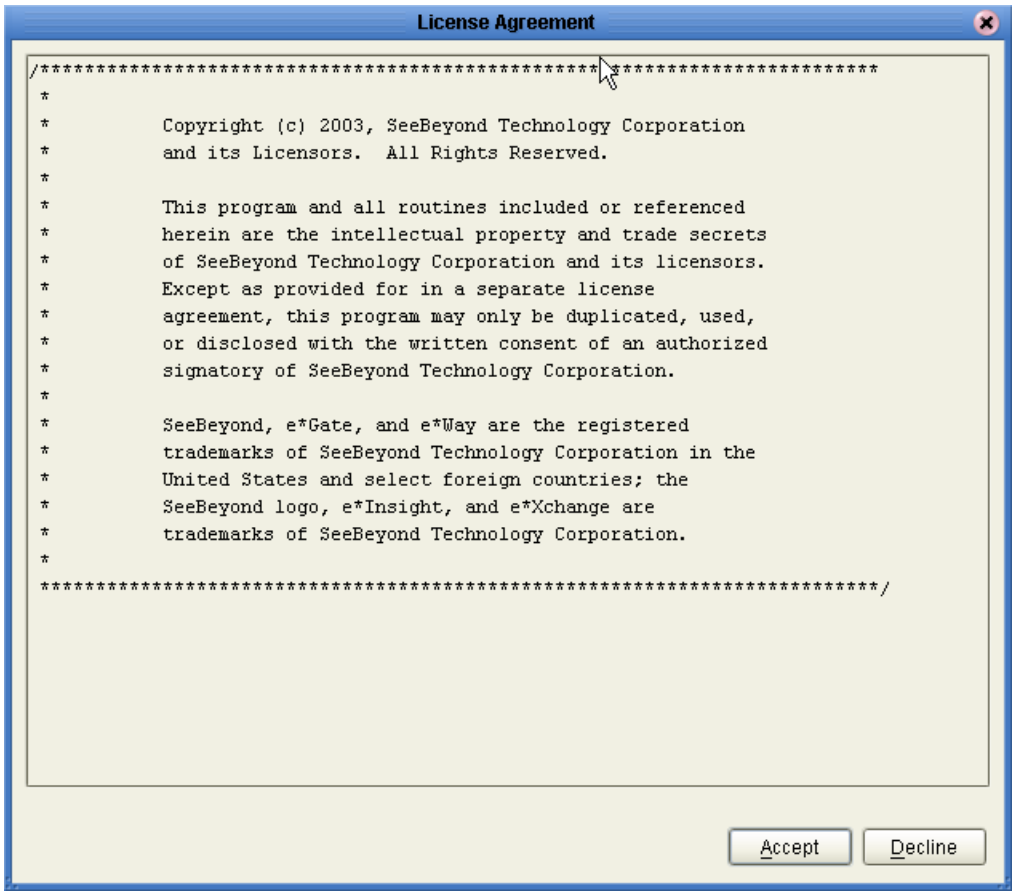
- To move all items in the **Available Updates and New Modules** box, click the **Add All** button (double-arrow button between the two panes).

This moves the eIndex files to the **Include in Install** list.

5 Click **Next**.

The License Agreement window appears (see **Figure 10 on page 43**).

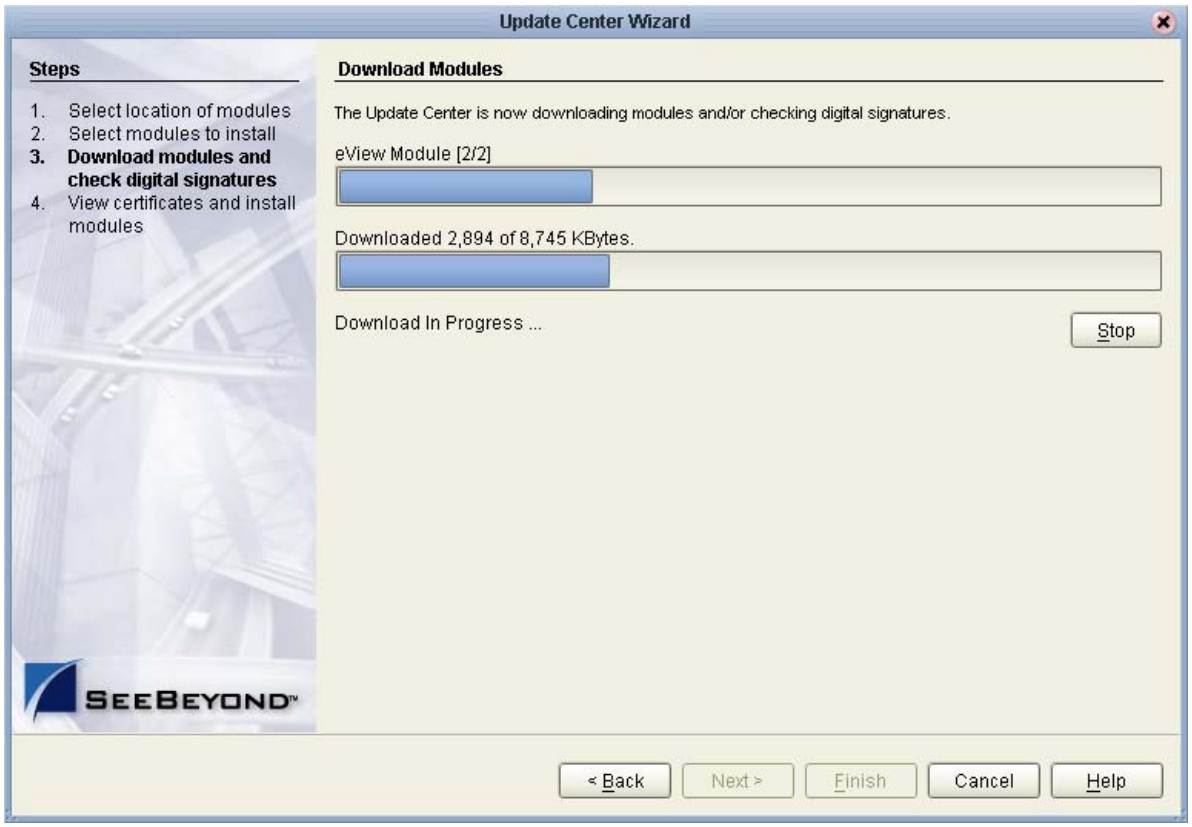
Figure 10 Update Center Wizard - License Agreement



6 On the **License Agreement** window, click **Accept**.

The **Download Modules** page appears (see Figure 11).

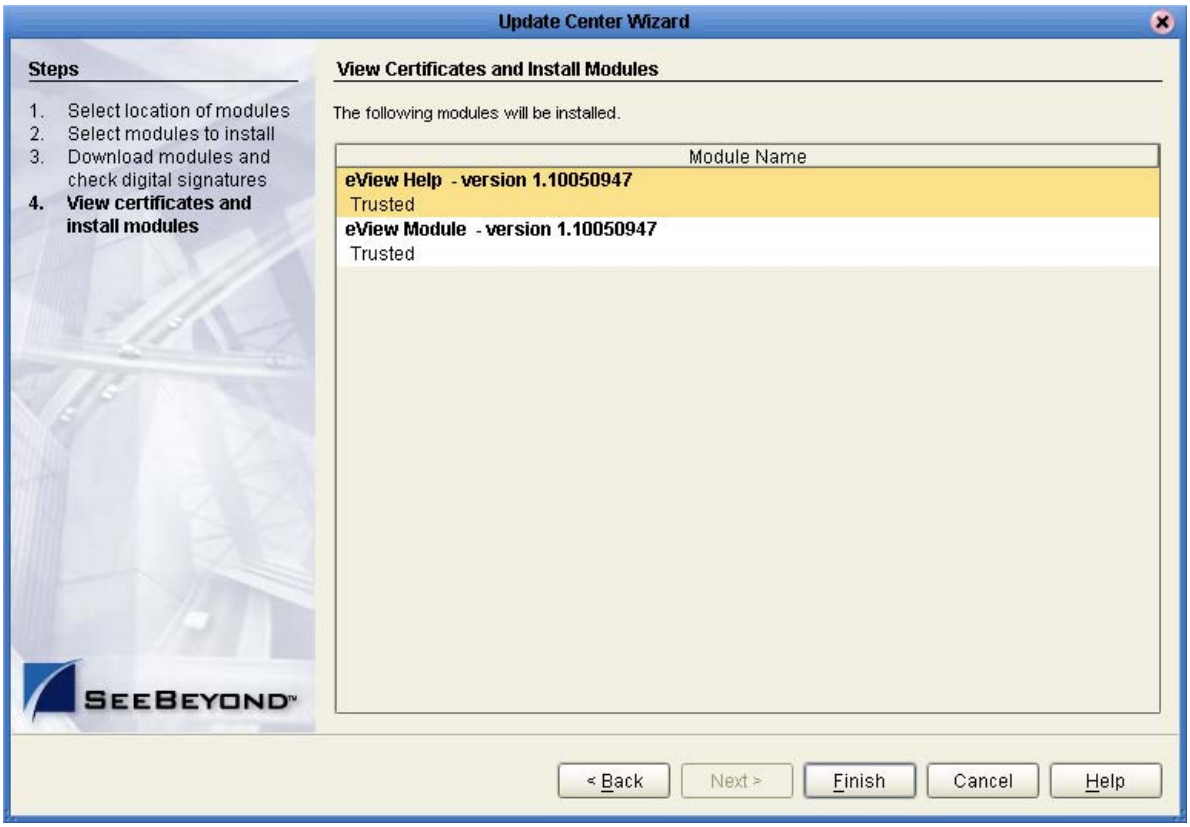
Figure 11 Update Center Wizard - Download Modules



7 After the progress bar reaches 100 percent, click **Next**.

The **View Certificates and Install Modules** page appears (see Figure 12).

Figure 12 Update Center Wizard - View Certificates and Install Modules

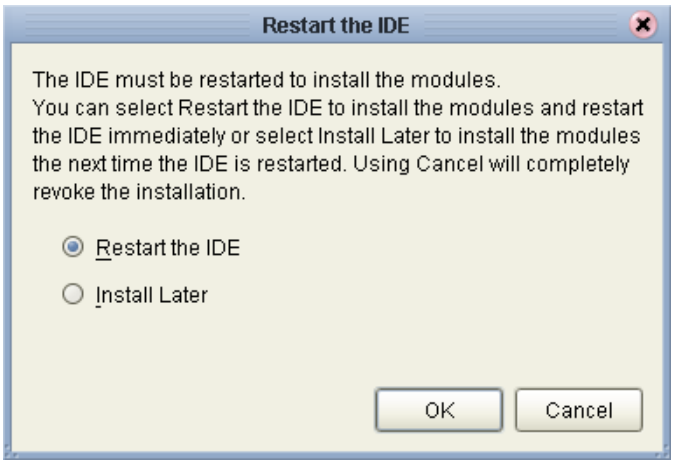


8 Click **Finish**.

The **Restart the IDE** dialog box appears (see Figure 13).

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

Figure 13 Update Center Wizard - Restart the IDE



10 To begin working with the eIndex module, restart the Enterprise Designer.

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

Chapter 4

Configuring eIndex

Several files are included in the eIndex Project that configure the structure and processing logic of eIndex. 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 *eIndex Global Identifier Configuration Guide*.

4.1 Configurable Options

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

Important: *If you are using the Ascential INTEGRITY matching algorithm you must replace the Match Field file. To do this, check out and open the Match Field file in Enterprise Designer, and then replace the text with that of the* ***mefa.xml*** *file provided in the eView INTEGRITY download extract. This file is located in the* ***eIndex50*** *folder in the directory where the add-on was downloaded.*

Object Definition

By customizing the Object Definition file, you can configure the structure of the data stored in eIndex. This file contains the configuration of each object in eIndex 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 Project.

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 attributes 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, results fields, and so on.

Query Definitions

The queries used in eIndex are all defined in the Candidate Select file. You can customize this file by configuring the types of queries used by eIndex, 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 blocks 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).

Standardization and Matching Rules

Standardization and matching are configured in two files: Threshold and Match Field. You can specify which match and standardization engines to use, and then 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. Finally, you can define certain match parameters that define weight thresholds and how assumed matches are processed.

Survivor Calculator

The logic that determines the data included 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 the fields from each system that 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 may 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.

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.

Field Validations

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

EUID Configuration

The configuration of the EUIDs assigned by eIndex is defined in the Threshold file. You can specify an EUID length, whether a checksum value will be 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 Configuration Files

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

The configuration files include the following:

- **Object Definition** on page 49
- **Candidate Select** on page 49
- **Match Field** on page 50
- **Threshold** on page 50
- **Best Record** on page 50
- **Field Validation** on page 50
- **Enterprise Data Manager** on page 51

These files can only be modified in the XML Editor provided with eIndex. This editor is described in the *eIndex Global Identifier Configuration Guide*.

4.2.1. Object Definition

The Object Definition file defines each object in eIndex 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 defines 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

This file configures the Query Builder component of eIndex, and defines the queries available for eIndex. 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 searches*. You can also define *blocking queries*, which define blocks of criteria fields for the

match process. eIndex 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

This file configures the Matching Service by defining standardization and matching fields for eIndex. It also specifies which 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. You must also define the data string to be passed to the match engine. The rules you define for standardization and matching are highly dependent on the standardization and match engines in use.

4.2.4. Threshold

This file configures the eIndex Manager Service, and defines attributes of the match process. You can specify the match and duplicate thresholds in this file, and define 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 the query from the Query Builder to use for matching queries.

This file also configures the EUIDs assigned by eIndex. You can specify an EUID length, whether a checksum value will be 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

This file defines the logic for determining the data to be included 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 a given 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

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 eIndex database, and then specify those classes in this file.

4.2.7. Security

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

4.2.8. Enterprise Data Manager

The appearance of the EDM is defined in the Enterprise Data Manager file. In this file, you define each field that appears on the EDM, along with the attributes 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.

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, and the results fields that appear. You can also specify 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 integration server in use, debugging options, and security details.

4.3 Modifying the eIndex Configuration Files

You may 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 using the XML verification function of the XML editor and the XSD validation function of the eIndex application. 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 may place other requirements on customizations. Be sure to review *Implementing the SeeBeyond Match Engine with eIndex Global Identifier* or *Implementing Ascential INTEGRITY with eIndex Global Identifier* 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 before using the eIndex application.

4.4 Match Engine Configuration Files

If you chose to implement the 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 provides a text editor for this purpose. For information about the structure of these files and how they can be modified, see *Implementing the SeeBeyond Match Engine with eIndex Global Identifier*.

Chapter 5

Generating the Project

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

5.1 Generated Application Components

Generating the eIndex 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 uses the Object Definition to update these components based on the configuration you defined. These components include:

- **Database scripts**—The scripts for creating and dropping tables and indexes are updated based on the Object Definition file.
- **Custom Plug-ins module**—This component allows you to define additional customized processing rules for eIndex to perform when certain transactions occur.
- **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 eIndex. These are customized for your application based on the Object Definition file.
- **eInsight methods**—eInsight methods can be used when processing data through eInsight rather than a Collaboration. They include a subset of the method OTDs, and are based on the Object Definition file.
- **Application JAR files**—These files are used by the web-based interface (EDM) and any client Projects that access eIndex.

5.2 Generating the Project

Once all modifications to the configuration files are complete, you can generate eIndex to update the components listed above. If you modify the configuration files after you

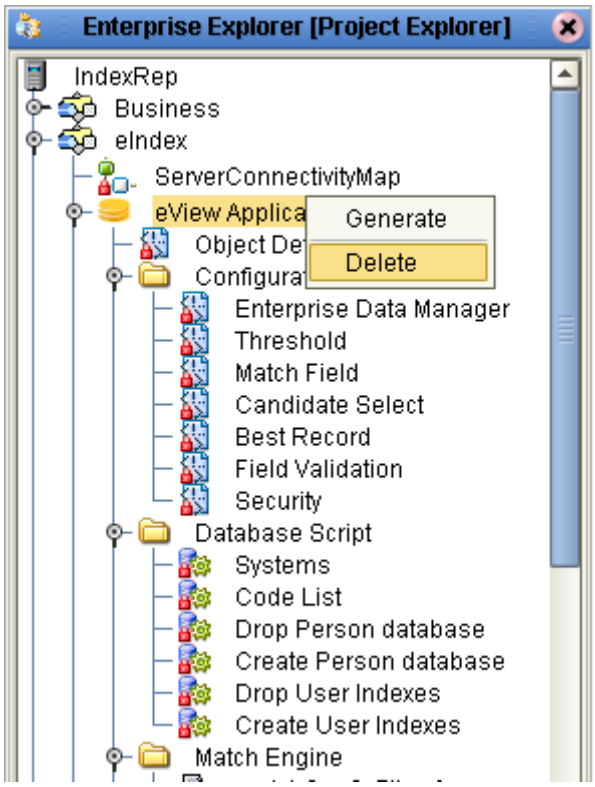
generate the application, you can regenerate the application to update the custom components.

To generate the Project

1 Save all configuration changes to the Repository.

2 Right-click the eIndex application in the **Project Explorer** pane to display the **Generate** context menu (shown in Figure 14).

Figure 14 Generate Context Menu



3 Select **Generate**. eIndex updates the Project components. This may take a few minutes.

4 Save the new components to the Repository.

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.*

Chapter 6

Creating Custom Plug-ins

eIndex provides the ability to create custom update procedures to be performed on a record once standard eIndex processing is carried out. 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 to use the custom procedures.

6.1 About Custom Plug-ins

You can add custom processing toeIndexusing the Custom Plug-ins module of an eIndex Project. You can use these plug-ins for field validations, update policies, and to create custom components for eIndex, such as a custom phonetic encoders, block pickers, or query builders. The components listed below are explained more fully in the *eIndex Global Identifier Configuration Guide*.

6.1.1. Update Policies

For the primary transactions performed by eIndex you can define additional custom processing to perform against the record that results from the 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 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 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. You can define several different types of update policies. Each policy modifies an enterprise object (class **com.stc.eindex.objects.EnterpriseObject**), and must implement **com.stc.eindex.update.UpdatePolicy**. This class contains one method, **applyUpdatePolicy**. The syntax is as follows:

```
public EnterpriseObject applyUpdatePolicy(EnterpriseObject
beforeImage, EnterpriseObject afterImage)
```

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

Enterprise Merge Policy

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

Enterprise Unmerge Policy

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

Enterprise Update Policy

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

Enterprise Create Policy

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

System Merge Policy

The system merge policy defines additional processing to perform when 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). Specify the fully qualified name of this custom plug-in in the **SystemMergePolicy** element of the Best Record file.

System Unmerge Policy

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

UndoAssumeMatchPolicy

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

6.1.2. 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, you can specify 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 by default to validate information about system and local ID fields before processing the data. This is described in the *eIndex Global Identifer Configuration Guide*.

6.1.3. Custom eIndex Components

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

Query Builder

The query builder defines the different types of queries that can be used in eIndex. You can create custom queries to implement. To add a new query builder, you must define a class that extends the base abstract **com.stc.eindex.querybuilder.QueryBuilder**, and then define that class in a **query-builder** element in the Candidate Select file. The exception thrown is **com.stc.eindex.querybuilder.QueryBuilderException**.

Three methods must be implemented.

- **init—**This method receives the XML elements after the config tag 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 in the call to **getApplicableQueryIds**.
- **buildQueryObject—**This method constructs the query object based on one of the applicable query IDs provided as an input argument.

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 as you define. Specify the fully qualified name of this custom plug-in in 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.

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.The class you define is specified 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.

Match Engine

You can define classes to connect to a custom match engine instead of the SeeBeyond Match Engine or INTEGRITY. The classes you define are specified 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.

Standardization Engine

You can define classes to connect to a custom standardization engine instead of the SeeBeyond Match Engine or INTEGRITY. The classes you define are specified 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.

Phonetic Encoders

Two phonetic encoders, NYSIIS and Soundex, are predefined for eIndex. You can define custom classes to implement additional phonetic encoders if needed. These classes are specified 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.

6.2 Implementing Custom Plug-ins

eIndex provides a simple method of incorporating custom Java code into an eIndex 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 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 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"**.

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 you would 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 application, you must build the plug-ins. This compiles the Java code and incorporates it into the application files.

To build custom plug-ins

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

Chapter 7

Creating the Database

The eIndex Project includes several database scripts to create the eIndex 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.

7.1 Database Scripts

The eIndex 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** node of the eIndex Project, and are named **Systems**, **Code List**, **Create User Indexes**, **Drop User Indexes**, **Create User Code Data, Create Person database**, **Drop Person database**. You can modify these scripts as needed to customize the tables, indexes, startup data, and distribution of the database.

7.2 Requirements

When configuring the eIndex 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 eIndex database and with your data processing requirements.

Database Platforms

The eIndex database can be run on either an Oracle 8.1.7 or an Oracle 9*i* platform. You must have this software installed before beginning the database installation.

Operating Systems

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

Hardware Requirements

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 NT, 2000, or XP database server, the following configuration is recommended as a *minimal* installation:
  - Windows NT 4.0 with SP4 or later
  - Pentium 300
  - 256 MB RAM (increase this based on the number of users, connections to the database, and volume)
  - 2.5 GB disk space plus an additional 2 KB for each 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)
  - 256-color video
  - CD-ROM device
- For a Unix database server, the following configuration is recommended as a *minimal* installation:
  - 256 MB RAM (increase this based on the number of users and connections to the database)
  - Swap space should be a minimum of twice the amount of RAM
  - 1 GB disk space plus an additional 2 KB for each 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)
  - CD-ROM device

Important: *Disk space recommendations do not take into account the volume and processing requirements or the number of users. These are minimal requirements to install a generic database. At a minimum, the empty database and the database software will require 2.5 GB of disk space.*

7.3 Database Structure

The eIndex database contains some tables that are created for all implementations. These 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 tables that store information about the enterprise object are named 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. sbyn_<*parent_object*> stores parent object information from each local system, and sbyn_<*parent_object*>sbr stores parent object information 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 4, "The Database Structure", in the *eIndex Global Identifier 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.

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 Reference* includes 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 eIndex 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 eIndex 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 also help you define the processing codes and the descriptions to define in the common tables.

7.4.3. Common Table Data

This part of the 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 members 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 Project includes a script to help you load the processing codes into the database.

When an enterprise object appears on the EDM, eIndex 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

This part of the 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 eIndex 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 may want to install these tables into different tablespaces, depending on the original size and expected volume of the database.

Sizing

To begin the database installation, you first create an Oracle database instance using Oracle configuration tools. You can use this tool to define the tablespace and extent sizing for the database.

Distribution

The Oracle configuration tools also allow you to define the distribution of your system tables, data tables, rollback logs, dump files, control files, and so on.

Indexes

By default, indexes are defined for the following tables: sbyn_appl, sbyn_common_header, sbyn_common_detail tables, 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 fields used for blocking or matching be indexed.

7.5 Creating the Database

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

7.5.1. Step 1: Analyze the Database Requirements

Before you begin to create the eIndex 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 eIndex. 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

Before beginning this step, be sure that the correct version of Oracle is installed on the database server. eIndex supports both Oracle 8.1.7 and Oracle 9*i*. 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 be defining tablespace 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 an administrator user, granting DBA with the admin option. Also give this user permission to select any table and to create users.

7.5.3. Step 3: Customize the Database Scripts

The database script that installs the database components specific to eIndex 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 eIndex, you can define additional indexes for the database tables that store object data. SeeBeyond 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** 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 Project defines one default source system for eIndex. 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** 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.
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	The length of the local identifiers assigned by the system. This length does not include any additional characters added by the input mask. ***Note:*** *The 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.*
format	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 J2SDK.
input_mask	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. ▪ **D** - indicates a numeric character. ▪ **L** - indicates an alphabetic character. ▪ **A** - indicates an alphanumeric character.
value_mask	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 **DDxDDDxDDD**. This strips the hyphens before storing the ID.
create_date	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.
create_userid	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.

Defining Code Lists

The **Code List** file of the eIndex 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** node and then double-click the **Code List** file.

   The file opens in the text editor.

2. In the **Code List** file, scroll to the following line.

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

   The statements following this line can be customized.

3. 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.*

4. 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')
   ```

5. 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 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).

Save and close the **Systems** 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. 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 Java 2Software Development Kit (SDK).
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. These character types can be used. ▪ **D**—Numeric character ▪ **L**—Alphabetic character ▪ **A**—Alphanumeric character
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** node. For information about the structure of the eIndex database, see the *eIndex Global Identifier Reference Guide*.

To create a custom script

1. In the eIndex Project, right-click the **Database** 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** 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

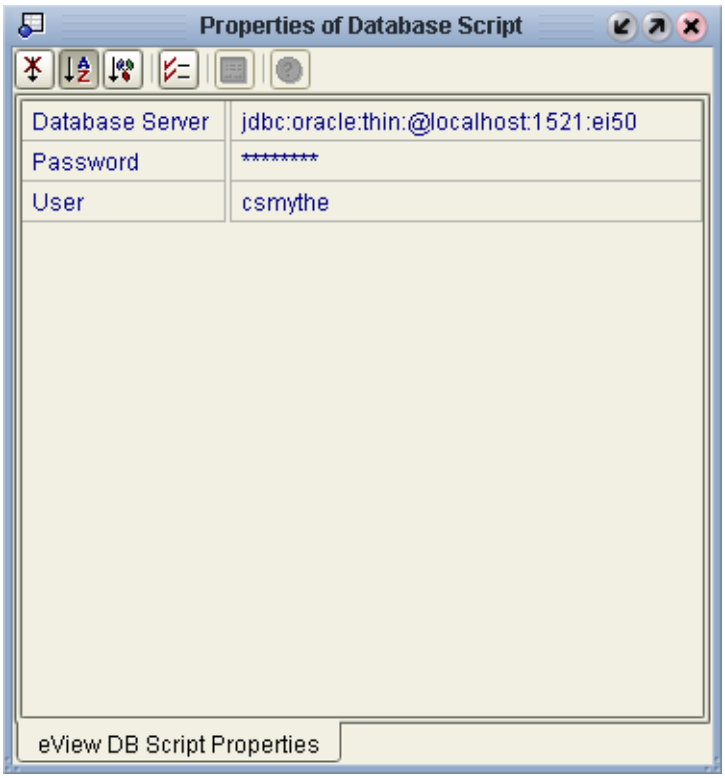
After you create the database instance and customize the database scripts, you can create the eIndex tables and insert the custom data.

To modify the database

1. In the eIndex Project, right-click the **Database** node, and then select **Properties** from the **Database** context menu.

   The **Properties of Database Script** dialog appears as shown in **Figure 15 on page 71**.

Figure 15 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"**.
- In the **Password** field, enter the password of the administrator user you created when you created the database.
- In the **User** field, enter the administrator user's logon ID.

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 eIndex start with "1", with padded zeroes added to the left to make the EUID number the correct length (for more information, see "Threshold Configuration" in the *eIndex Global Identifier 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 eIndex Tables or Indexes

Scripts are provided to drop the default database tables and indexes created in **Step 4: Modify the Database** on page 70. This is useful while testing eIndex implementation.

To delete eIndex tables or indexes

1. To drop the indexes created by the **Create User Indexes** script, right-click **Drop User Indexes**, and then select **Run**. On the confirmation dialog, click **OK**.
2. To drop the eIndex database tables, right-click **Drop Person database**, and then select **Run**. On the confirmation dialog, click **OK**.

Chapter 8

Defining Connectivity Components

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

8.1 Overview

Data can be processed by eIndex in four ways. First, data is processed through the EDM. This process is defined by the Connectivity Map in the eIndex server Project. Second, data is processed from the external systems that share information with eIndex. This process is defined by the Connectivity Maps in the external system Projects. Next, you can incorporate eIndex methods in an eInsight Business Process and develop eVision Web pages to access data from the eIndex database. This process is defined by the Connectivity Map in an eInsight integration Project. Finally, you can define a JMS Topic to which eIndex publishes messages to broadcast to external systems. This topic is included in the eIndex server Project and also in Projects for the external systems.

8.1.1. Connectivity Components

The connectivity components of an eIndex server Project include the eIndex application files and an optional JMS Topic. The client Projects that connect to eIndex use standard connectivity components of an eGate Project, with the addition of an eIndex method OTD or Java methods for eInsight integration.

eIndex Server Project Connectivity Components

The connectivity components in an eIndex server Project include:

- **Connectivity Map**—Graphically describes the relationship between the web application files and the eIndex application files.
- **Application file**—Contains the logic used by eIndex to process data into and out of the eIndex database. This file is automatically created when you generate the eIndex 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 Project.
- **JMS Topic**—A message destination conforming to the *publish-and-subscribe* (pub/sub) messaging paradigm. In this case, eIndex publishes to the topic to broadcast messages to external systems.

Client Project Connectivity Components

The connectivity components in an eIndex client Project can include any of the following:

- **Connectivity Map**—Graphically describes the relationship between the External Systems, Queues and Topics, Services, Web Connectors, and eIndex 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 external system or Web connector in the eIndex system must include the eIndex application in its Connectivity Map in order for those components to exchange information with eIndex.
- **Service**—Provides a framework for a process or a Collaboration, which 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 describing external objects including both data structure and methods. A custom method OTD is created in the eIndex Project for use in the client Projects to define how data is processed between eIndex and external systems or eInsight. A custom OTD is also created to publish messages from eIndex 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 eIndex to connect with external systems via eGate. These 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. External system Project can include a JMS Topic to which eIndex publishes, giving the external system 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 eWay is an application-specific adapter linking an external application with eGate.
- **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 *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. These maps do not include a JMS Topic to which eIndex publishes. To publish eIndex transactions to external systems, a JMS Topic must be added to the Connectivity Maps. This section describes how to add a JMS Topic to the predefined Connectivity Map and to define connectivity components for external system Projects and Projects integrating eIndex with eInsight. Perform the following tasks to configure connectivity for eIndex and connected systems.

- **Adding a JMS Topic to the eIndex Connectivity Map** on page 75
- **Defining External System Connectivity Components** on page 77
- **Defining eInsight Integration Connectivity Components** on page 83

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

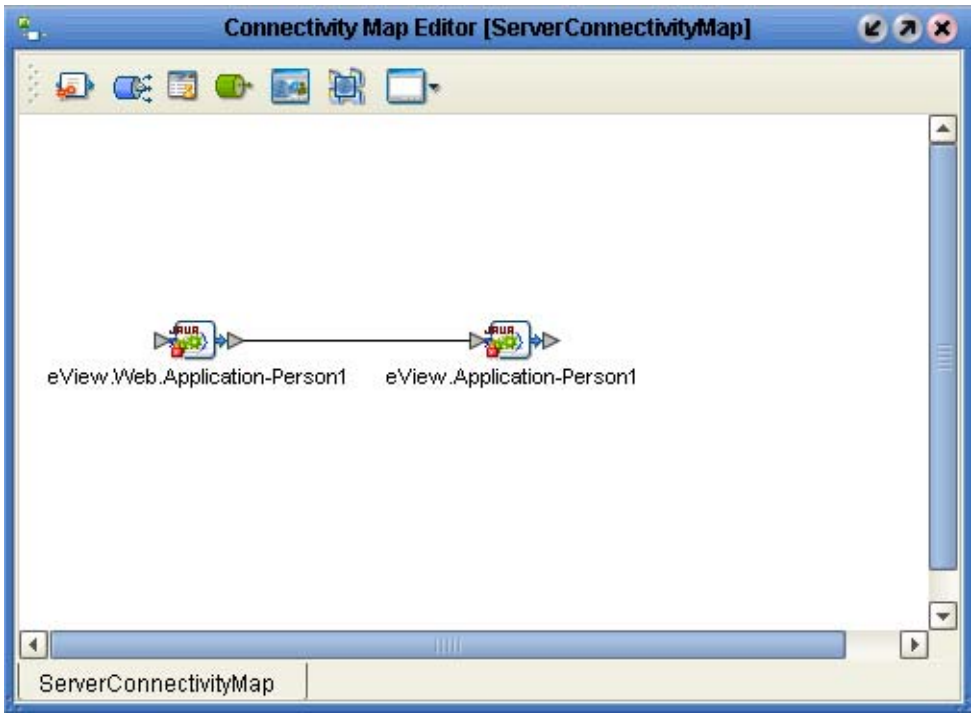
8.2.1. Adding a JMS Topic to the eIndex Connectivity Map

The Connectivity Map in the eIndex server Project is already created. The default map contains only the application files, but you can add a JMS Topic to publish eIndex messages. This section describes how add a JMS Topic to the eIndex 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 Enterprise Explorer, check out the Connectivity Map to which you want to add the new topic.
2. Open the Connectivity Map in the Connectivity Map Editor, as shown in Figure 16.

Figure 16 eIndex Connectivity Map



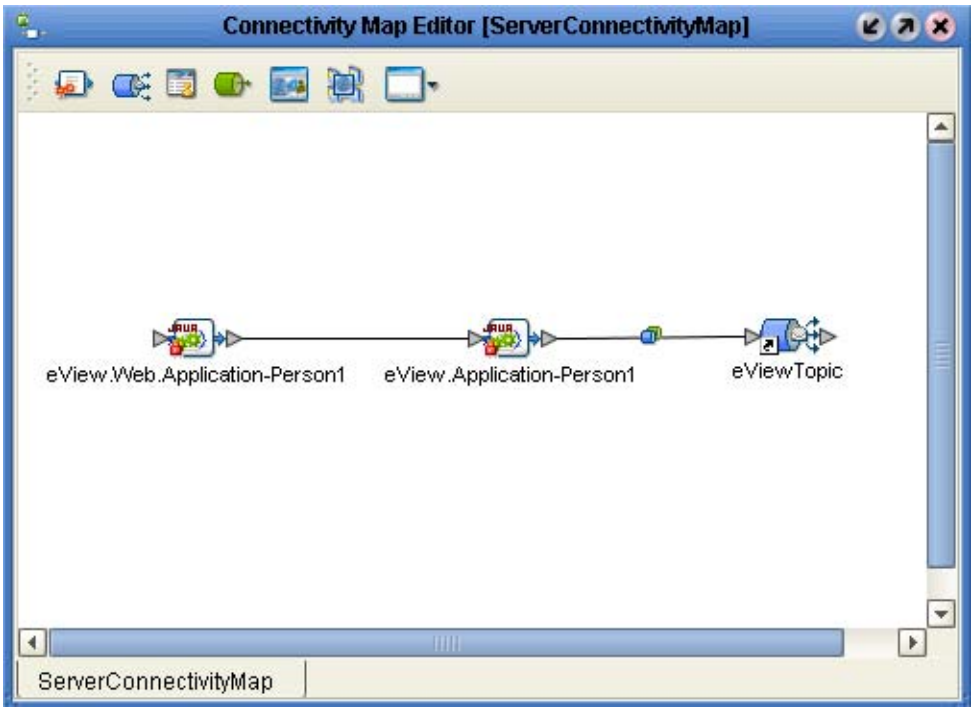
3 In the Connectivity Map toolbar, select the **Topic** icon, and drag it to the right of the **eView.Application-Person** icon.

4 In the Connectivity Map Editor, place the cursor over the arrow to the right of the **eView.Application-Person** icon until the cursor turns into a hand.

5 Click the arrow and drag it to the **Topic** icon.

The Connectivity Map should look similar to Figure 17.

Figure 17 eIndex Connectivity Map with Topic



6 If necessary, configure the JMS Client Connection (for more information, see the *eGate Integrator User's Guide*).

8.2.2. Defining External System Connectivity Components

In the client Projects for external systems sharing data with eIndex, the Connectivity Map contains business logic and information about how data is transferred between eIndex 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 this map by adding a JMS Topic or by changing the Java Collaboration. Perform these tasks to configure the connectivity components.

- **Modifying the Java Collaboration** on page 77
- **Modifying the External System Project Connectivity Map** on page 78
- **Configuring the Outbound Collaboration** on page 82

Modifying the Java Collaboration

This section describes how the eIndex method OTD is used in Java Collaborations for external systems. For a complete reference of the methods included in the eIndex OTD, see the *eIndex Global Identifier Reference Guide*. For information about working with the Java Collaboration Editor, see the *eGate Integrator User's Guide*.

To modify the Java Collaboration

1 In the eIndexClient Project, check out the **ProcessPerson** Collaboration, and then open it in the Java Collaboration Editor.

2 To use eIndex methods in the Collaboration, do the following:

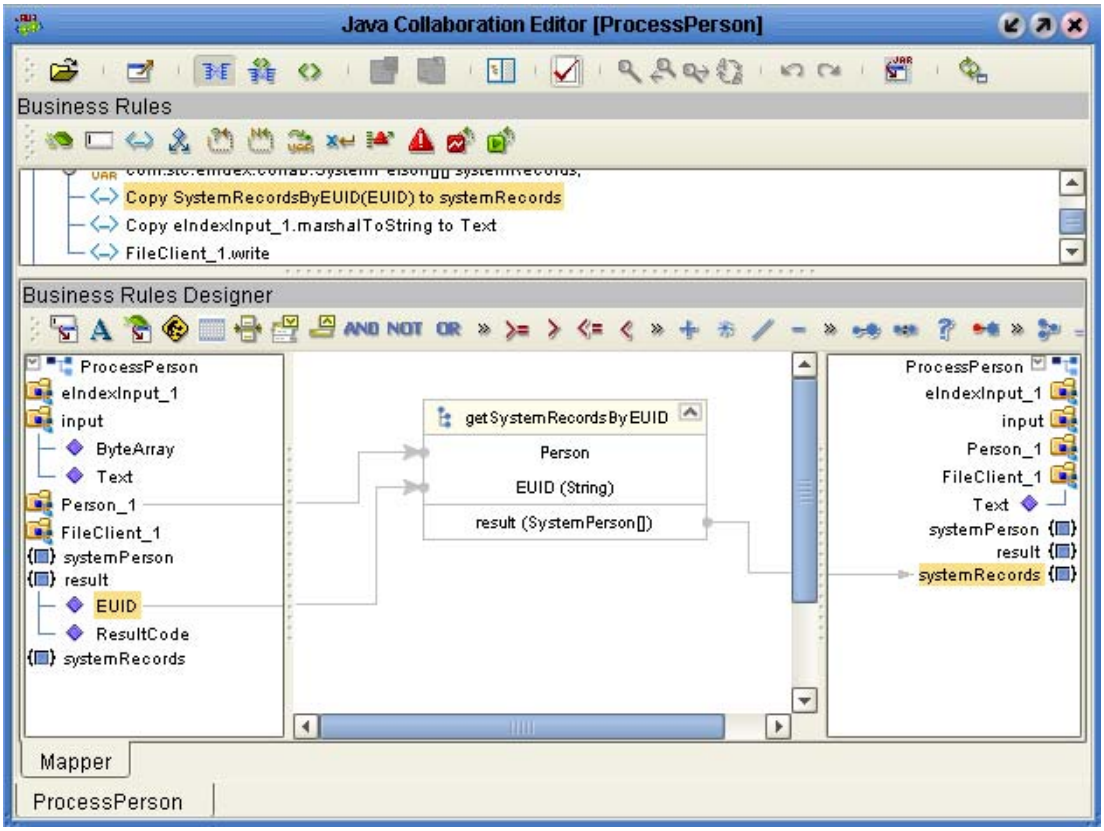
A In the left pane of the Transformation Designer, right-click **Person_1**. A list of available methods appears.

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 18 illustrates a sample of the **getSystemRecordsbyEUID** method in the Java Collaboration Editor.

Figure 18 eIndex Method OTD in Java Collaboration Editor



3 You can delete or modify existing information as needed. When you are done defining the processing rules, save the Collaboration.

Modifying the External System Project Connectivity Map

Connectivity between the eIndex 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.

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 messages to external systems.

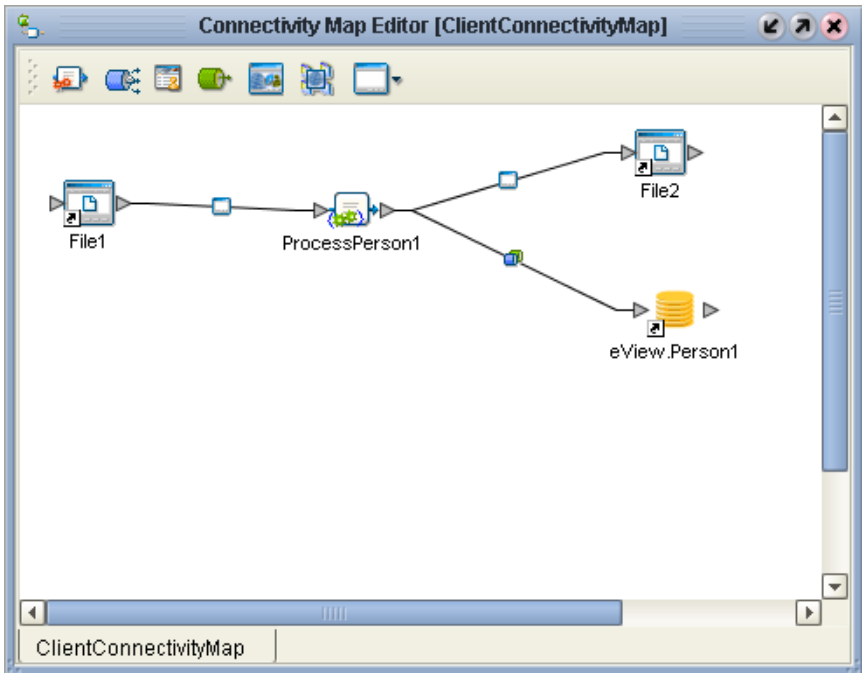
Note: *The eIndex application icon in this Connectivity Map comes from the External Applications menu on the Connectivity Map Editor toolbar.*

To configure the File External Applications in the sample

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

The predefined map appears, as shown in Figure 19.

Figure 19 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.

4 Repeat steps 2 and 3 for the **File2** eWay.

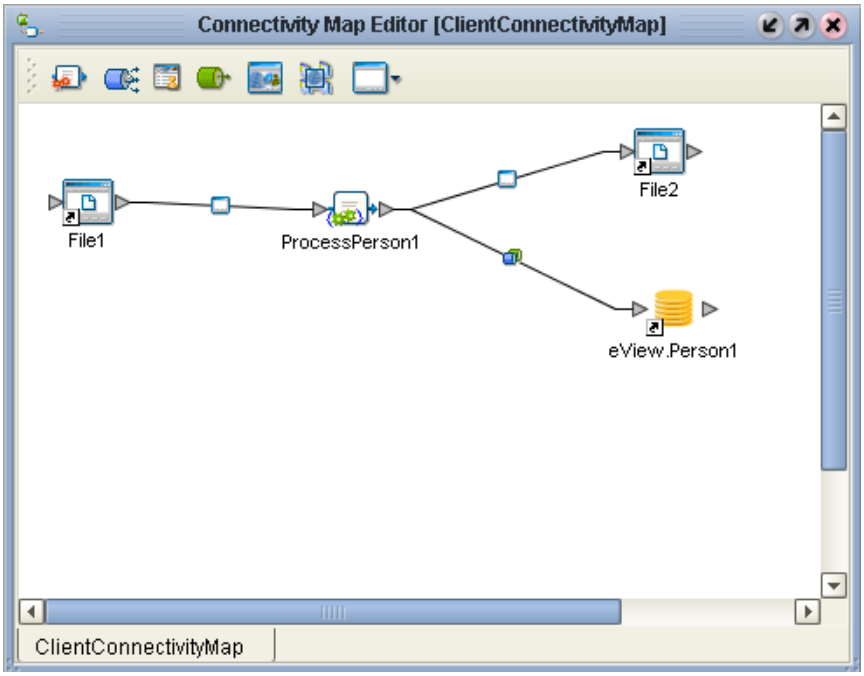
5 Save and close the Connectivity Map.

To add the JMS Topic to the Connectivity Map

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

The predefined map appears, as shown in Figure 20.

Figure 20 eIndexClient Connectivity Map



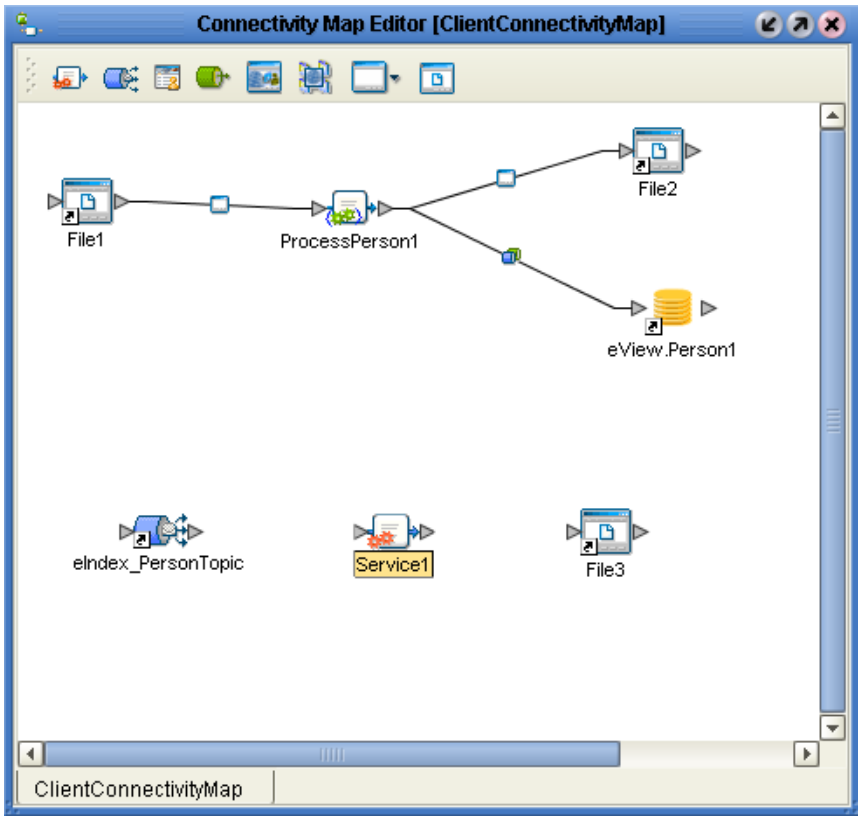
2 Drag the JMS Topic from the eIndex Project to the Connectivity Map Editor, below the existing connectivity diagram (this is the topic you created in **"Adding a JMS Topic to the eIndex Connectivity Map" on page 75**).

3 Drag a Service from the Connectivity Map Editor toolbar to the right of the JMS Topic on the canvas.

4 Drag an external source system of the appropriate type from the Connectivity Map Editor toolbar to the right of the new Service on the canvas (for testing purposes, use a File External Application).

The Connectivity Map should now look similar to Figure 21.

Figure 21 eIndexClient Connectivity Map with JMS Topic

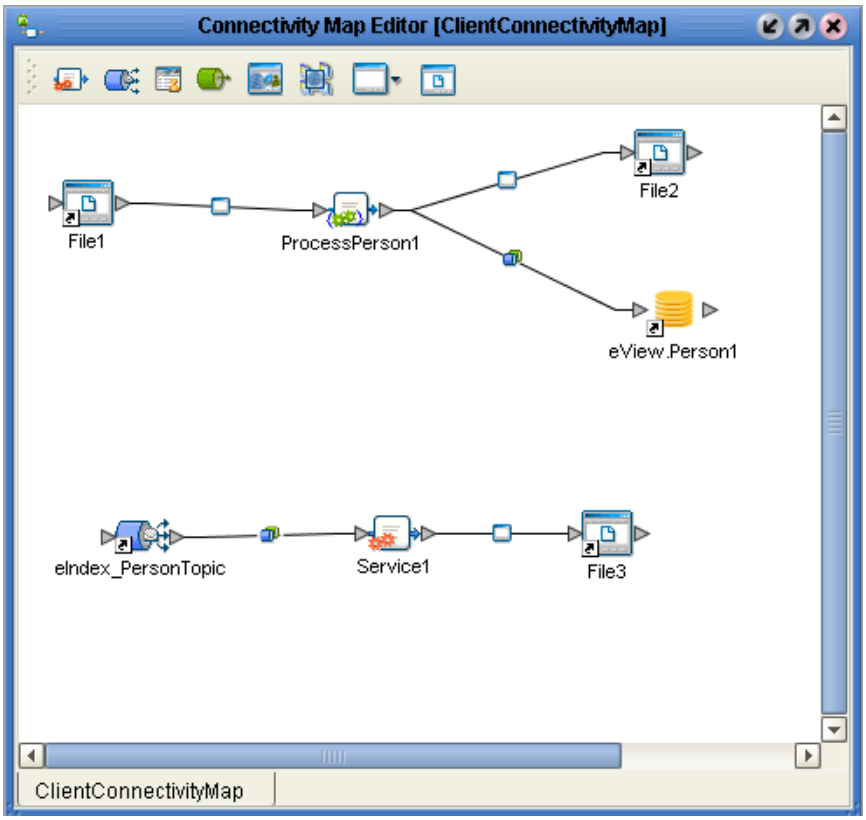


5 In the Connectivity Map Editor, place the cursor over the arrow to the right of the **Topic** icon until the cursor turns into a hand, and then drag it into the Service to connect the two objects.

6 Repeat step 5 to connect the Service to the external system.

The Connectivity Map should now look similar to Figure 22.

Figure 22 eIndexClient Connectivity Map with Connections



7 If necessary, configure the JMS Client Connection (for more information, see the *eGate Integrator User's Guide*).

8 Double-click the **File3** eWay to configure the location and parameter settings for the File 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 process messages from the eIndex JMS Topic. Before you begin, make sure you have completed all of the steps in **"Modifying the External System Project Connectivity Map" on page 78**.

To configure the outbound Collaboration

1 In the Project Explorer, right-click the **eIndexClient** Project.

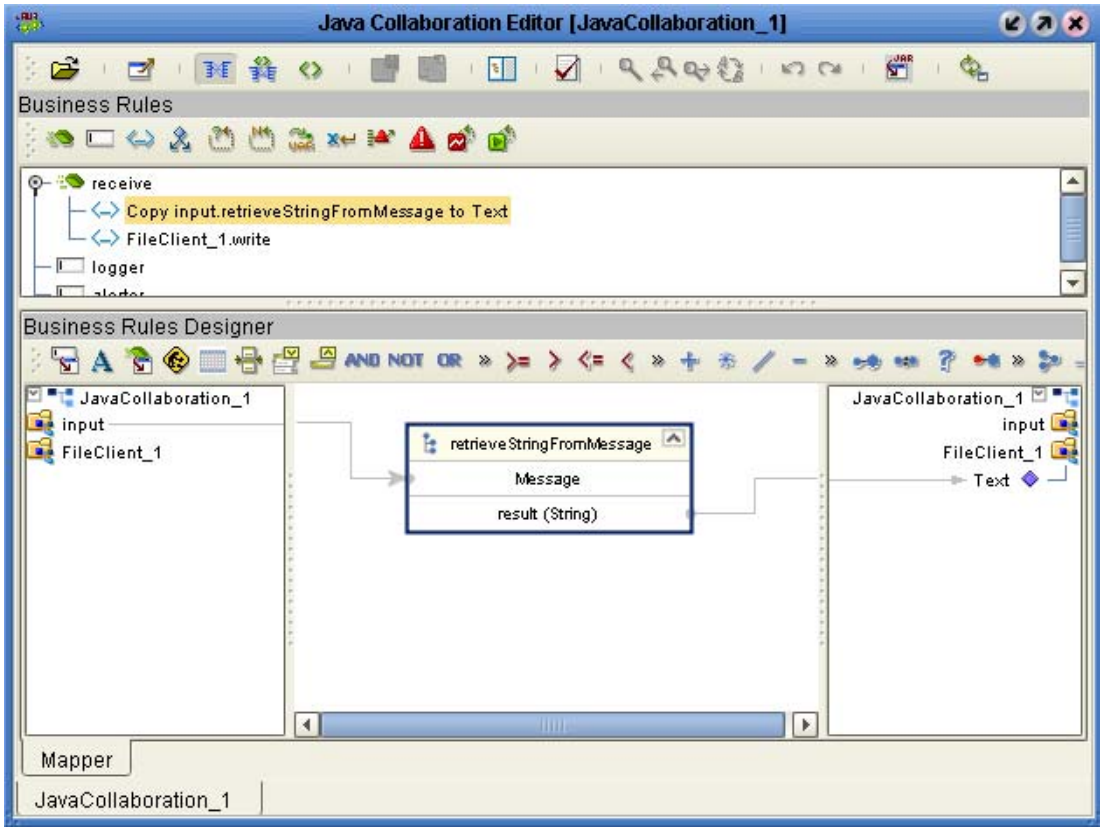
2 In the **Project** context menu, select **New**, and then select **Java Collaboration Definition**.

3 Enter information into the Java Collaboration Definition Wizard, with the following guidelines:

- For the **Web Service Type**, select the existing JMS receive type (navigate to **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 23.)

Figure 23 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. Defining eInsight Integration Connectivity Components

In the eInsight integration Projects, the Connectivity Map contains business logic and information about how data is transferred between eIndex and a Business Process for viewing on eVision Web pages. This section describes how to incorporate eIndex methods into a Business Process, create the Connectivity Map, and then add and connect the connectivity components.

- **"Including eIndex Methods in a Business Process" on page 84**

- "Creating the eInsight Integration Connectivity Map" on page 86
- "Connecting Connectivity Map Components" on page 88

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

Including eIndex Methods in a Business Process

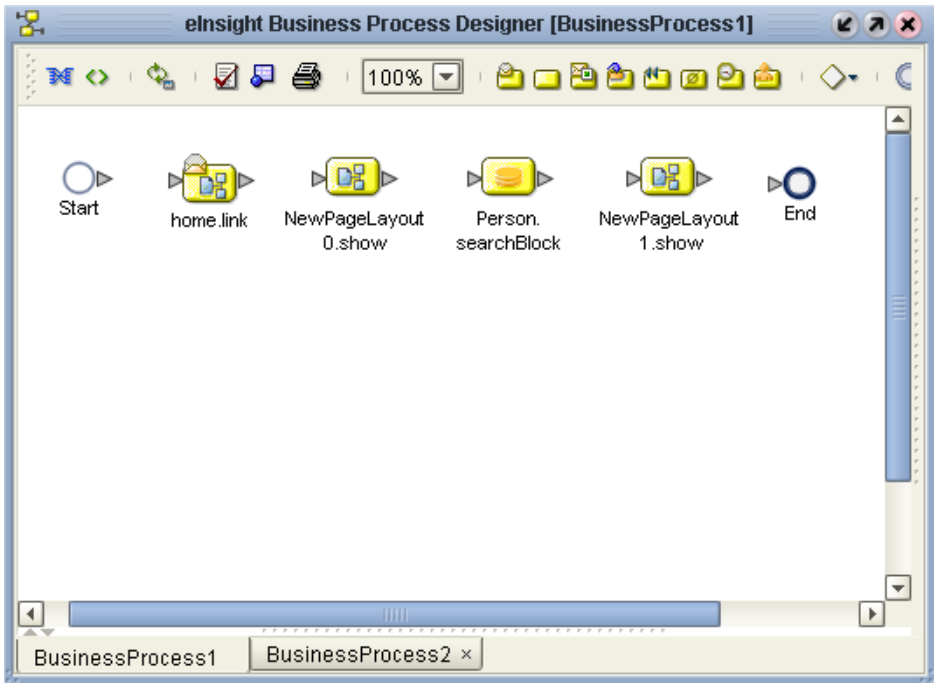
Before including the eIndex application in the Connectivity Map for eInsight integration, you must add eIndex methods to a Business Process. This section provides instructions for adding an eIndex method to an eVision Web page in a Business Process. For more information about the available methods, see the *eIndex Global Identifier Reference Guide*.

To include eIndex methods in a Business Process

1. In the client Project, create two page layouts: one for the user input web page and one for the eIndex output Web page.
2. Create a new page link for the Web pages.
3. Create a new Business Process.
4. Drag the page link icon from the Project Explorer onto the Business Process Editor to the right of the **Start** icon.
5. Drag the input Web page icon from the Project Explorer onto the Business Process Editor to the right of the page link activity.
6. Drag the output Web page icon from the Project Explorer onto the Business Process Editor to the left of the **End** icon.
7. In the eIndex Project, expand the method OTD folder to display the method list and then drag the method you want to use into the Business Process Editor to the right of the input web page activity (so it is between the input and output web page activities, as shown in Figure 24).

Note: *The method OTD is the folder in the eIndex Project with the same name as the eIndex application.*

Figure 24 eInsight Business Process with eIndex Activity



8 Save the Business Process.

Connecting the Business Process Components

This section describes how to connect the components of a Business Process that applies eIndex methods to eVision Web pages. Make sure you have completed all of the steps in **"Including eIndex Methods in a Business Process" on page 84**.

To connect the Business Process Components

1 In the Connectivity Map Editor, place the cursor over the arrow to the right of the **Start** icon until the cursor turns into a hand.

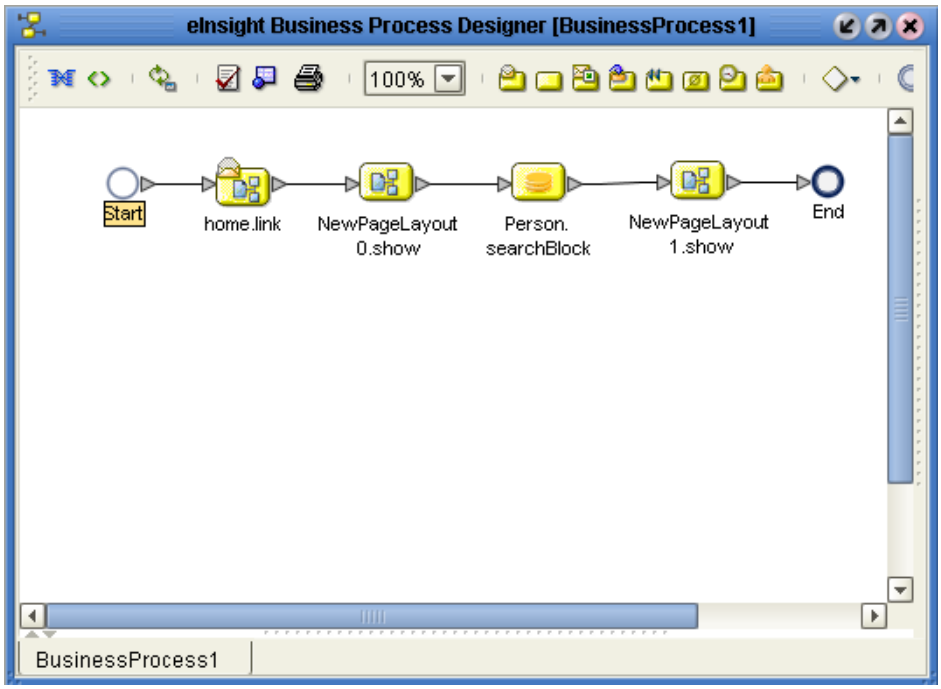
2 Click the arrow and drag it to the **Page Link** activity.

3 Follow the same procedure to link the following activities:

- Link the **Page Link** activity to the input web page activity.
- Link the input web page activity to the **eIndex** activity.
- Link the **eIndex** activity to the output web page activity.
- Link the output web page activity to the **End** icon.

The business process should look similar to Figure 25.

Figure 25 eInsight Business Process with Connections



4 For each link you created in step 3, 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 *eInsight Business Process Manager User's Guide*.)

5 Save the Business Process to the Repository.

Creating the eInsight Integration Connectivity Map

This section describes how to create, and add components to, the eInsight integration Connectivity Map.

To create the eInsight Integration Connectivity Map

1 In Enterprise Explorer, select the Project to which you want to add the Connectivity Map.

2 Right-click to display the **Project** context menu.

3 Select **New > Connectivity Map** to add a **Connectivity Map** icon to the Project and display the Connectivity Map Editor window.

4 Click the **Connectivity Map** icon in the Project Explorer and change the default name to the name you want to use.

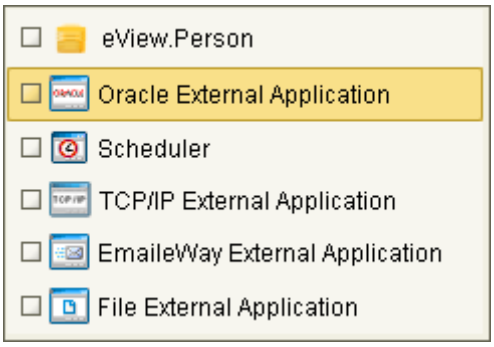
5 Drag a **Web Connector** icon from the Connectivity Map Editor toolbar onto the canvas.

6 Drag a **Service** icon from the Connectivity Map Editor toolbar onto the canvas to the right of the **Web Connector** icon.

7 Drag the Business Process created in **"Including eIndex Methods in a Business Process"** into the Service.

8 On the Connectivity Map Editor toolbar, click the down arrow next to the **External Systems** icon and select the check box next to the name of the eIndex application you want to integrate.

Figure 26 External System Menu

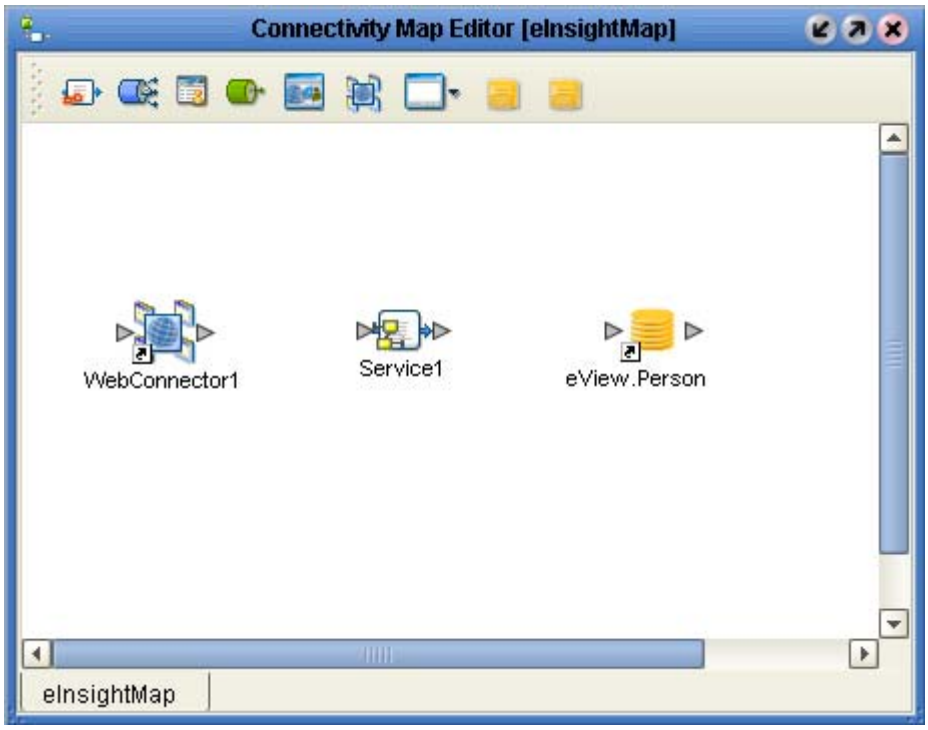


The eIndex application icon appears in the Connectivity Map Editor toolbar.

9 Drag the eIndex application icon from the Connectivity Map Editor toolbar onto the Connectivity Map Editor to the right of the **Service** icon.

The Connectivity Map should now look similar to Figure 27.

Figure 27 eInsight Integration Connectivity Map



10 Save the Connectivity Map to the Repository.

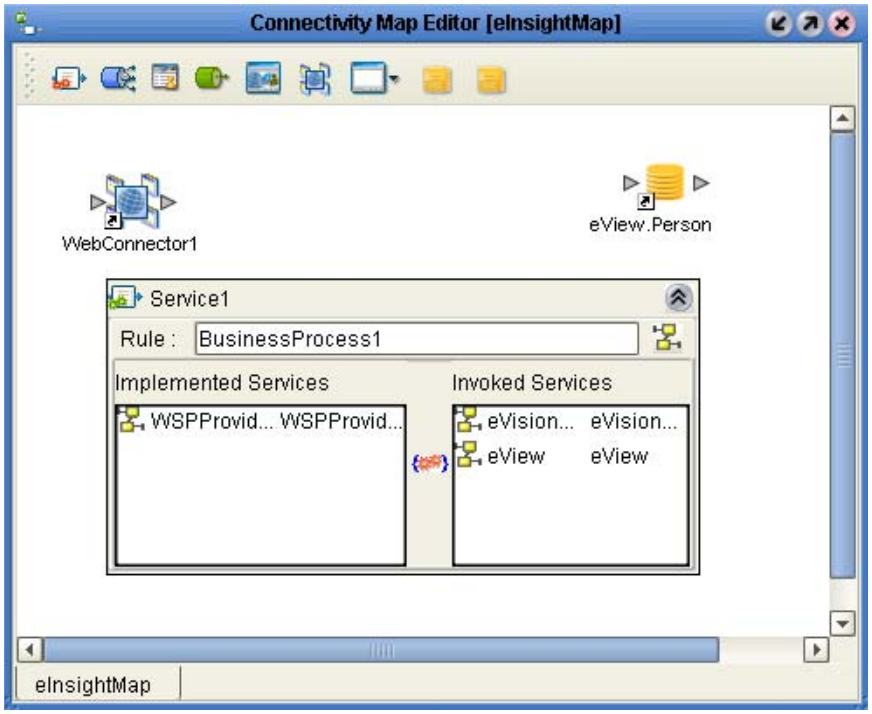
Connecting Connectivity Map Components

Once you create the components of a Connectivity Map, you must link them to define the flow of data through the system. Before you connect the components, make sure you have completed all of the steps in **"Creating the eInsight Integration Connectivity Map" on page 86**.

To connect Connectivity Map components

1 In the Connectivity Map, double-click the **Service** icon to display the Collaboration Binding window, as shown in Figure 28.

Figure 28 Collaboration Binding Window, eInsight



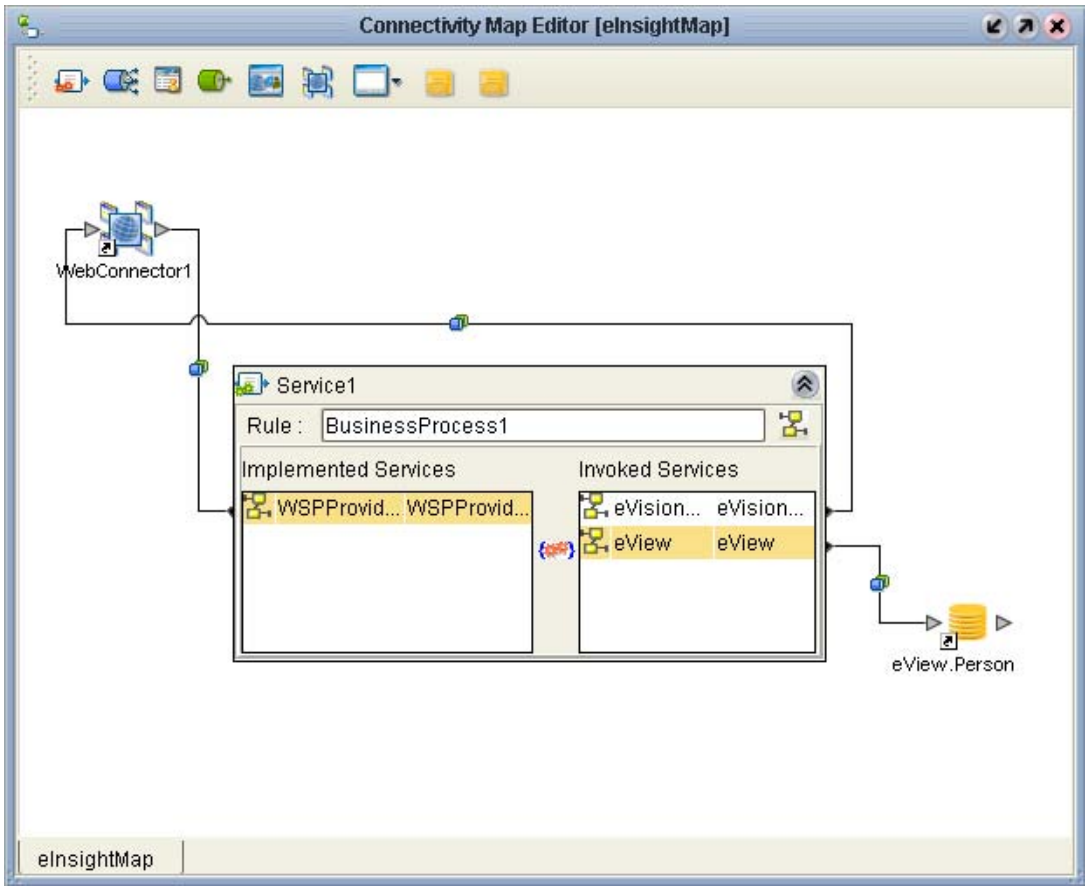
2 Drag **eVision** from the **Invoked Services** box in the Collaboration Binding window to the **Web Connector** icon on the Connectivity Map Editor.

3 Drag the eIndex application from the **Invoked Services** box in the Collaboration Binding window to the **eIndex application** icon on the Connectivity Map Editor.

4 Drag the Partner (in the illustrations, **WSPProvider**) from the **Implemented Services box** in the Collaboration Binding window to the **Web Connector** icon on the Connectivity Map Editor.

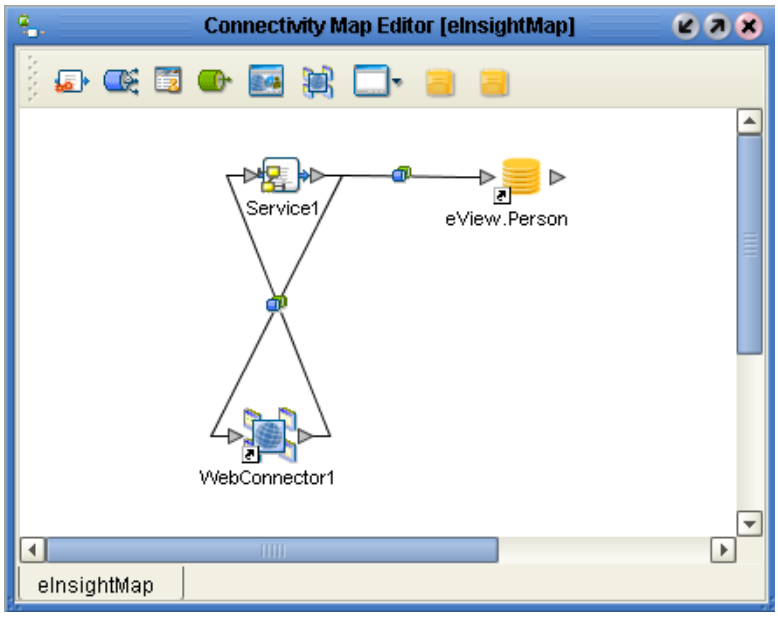
Figure 29 illustrates the Connectivity Map with the connections in place.

Figure 29 Collaboration Binding Window Connections, eInsight



5 Close the Collaboration Binding window. The Connectivity Map window should now look similar to Figure 30.

Figure 30 eInsight Connectivity Map With Connections



6 Configure the JMS Client Connection between the Web Connector and the Business Process Service (for more information, see the *eGate Integrator User's Guide*).

7 Save the Connectivity Map to the Repository.

Chapter 9

Defining the Environment

The eIndex Environment defines the configuration of the physical environment of eIndex, including the Logical Host, integration server, JMS IQ Manager, constants, and external systems. This chapter describes building a generic environment for an eIndex application. For more information about Environments and Environment components, see the *eGate Integrator User's Guide*.

9.1 Environment Components

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

Environment Components

The Environments configured to support the eIndex application Project include the following components:

- **Logical Hosts**—Each Environment contains one or more Logical Hosts, which are instances of the eGate runtime environment installed on a host hardware platform.
- **Integration Servers**—The Logical Host contains one or more Integration Servers, which are the engines that run eGate Services and eWays. It provides services for security, transactions, business rules execution, and database connectivity management.
- **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 Projects connecting external systems with the eIndex application.
- **eVision External Systems**—An eVision external system is a representation of an eVision Web application. This component is required for Projects integrating eIndex with eInsight.

- **Environmental Constants**—You can define constants for a specific Environment. Environmental constants are name/value pairs that are visible across the Environment.

9.2 Building an Environment

Each Environment represents a unit of software that implements one or more eIndex applications. You must define and configure at least one Environment for eIndex before you can deploy the application. These tasks you can perform to build an Environment for the eIndex application are described on the following pages.

- **"Creating an eIndex Environment" on page 92**
- **"Adding a Logical Host" on page 93**
- **"Adding Servers" on page 94**
- **"Adding an External System" on page 95**
- **"Configuring the Integration Server" on page 97**

9.2.1. Creating an eIndex Environment

This section describes how to create an Environment for an eIndex 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. Enter the name of the new Environment as shown in Figure 31.

Figure 31 New Environment

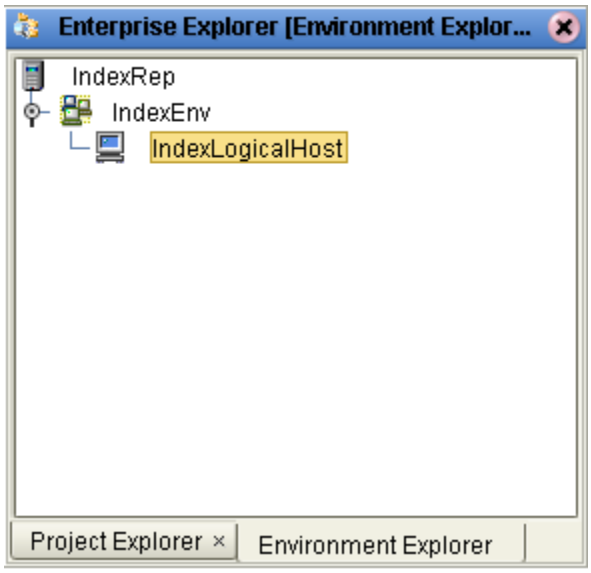
9.2.2. Adding a Logical Host

This section describes how to add a logical host to the eIndex Environment. This is a required step for all eIndex implementations.

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. Select **New Logical Host** to add a **Logical Host** icon to the **Environment Explorer** tab.
4. Enter the name of the new Logical Host as shown in Figure 32.

Figure 32 eIndex Logical Host



9.2.3. Adding Servers

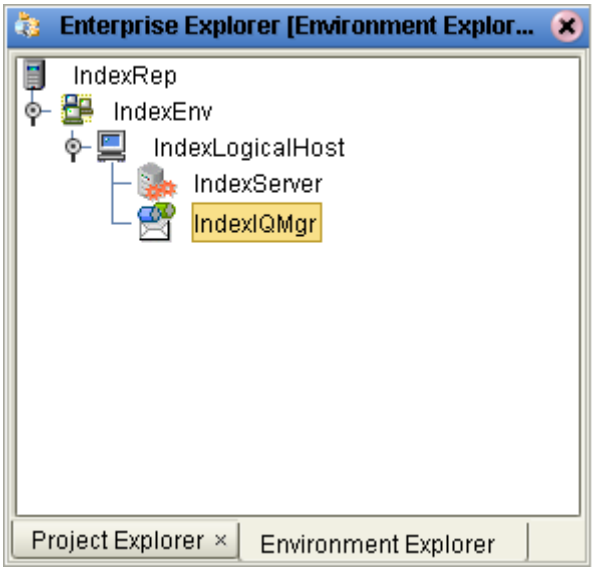
This section describes how to add integration servers and JMS IQ Managers to the eIndex Logical Host. You must add an integration server to the Environment.

To add integration servers and JMS IQ Managers

1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the Logical Host icon of the eIndex Logical Host.
3. Right-click the Logical Host icon to display the **Logical Host** context menu.
4. Select **New SeeBeyond Integration Server**.
5. Right-click the mouse to redisplay the **Logical Host** context menu.
6. Select **New SeeBeyond JMS IQ Manager**.

   Figure 33 illustrates an Environment with a Logical Host, SeeBeyond Integration Server, and SeeBeyond JMS IQ Manager.

Figure 33 Integration and JMS Servers



7 Configure the connection to the database, as described in **"Configuring the Integration Server" on page 97**.

9.2.4. Adding an External System

This section describes how to add an external system to the eIndex Environment. This is only required for Projects connecting an external system to the eIndex master index.

To add an external system

1 In Enterprise Explorer, click the **Environment Explorer** tab.

2 Select the eIndex Environment icon.

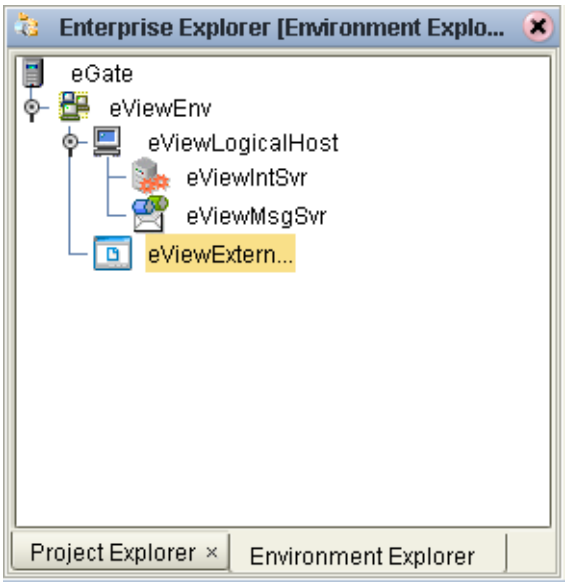
3 Right-click the eIndex Environment icon to display the **Environment** context menu.

4 Select **New <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 Enter the name of the new external system as shown in Figure 34.

6 Repeat these steps for each external system defined in the Projects that will be using this Environment.

Figure 34 File External System



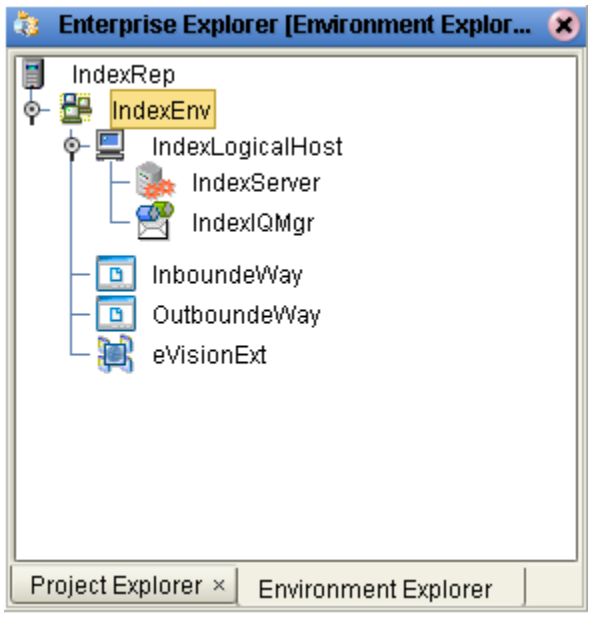
9.2.5. Adding an eVision External System

This section describes how to add an eVision external system to the eIndex Environment. This is only required for Projects connecting a Business Process for eVision Web pages to eIndex.

To add an eVision external system

1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the eIndex Environment icon.
3. Right-click the eIndex Environment icon to display the **Environment** context menu.
4. Select **New eVision External System** to add the system to the Environment.
5. Enter the name of the new eVision external system as shown in Figure 35.
6. Configure the eVision external system properties.

Figure 35 eVision External System



9.2.6. Configuring the Integration Server

The configuration of the integration server varies depending on the type of integration server you are using. For both servers, you must define the data source and specify environment variables.

Defining the Data Source

In order to connect to the database through the SeeBeyond Integration Server, you must configure the JNDI data source for the server. This supplies the information necessary to establish a connection to the database.

Note: *To access the EDM for eIndex, you need to know the host name and port number of the server. This information is found on the SIS* ***Properties*** *page in the* ***Web Connection Container*** *section.*

To define the data source

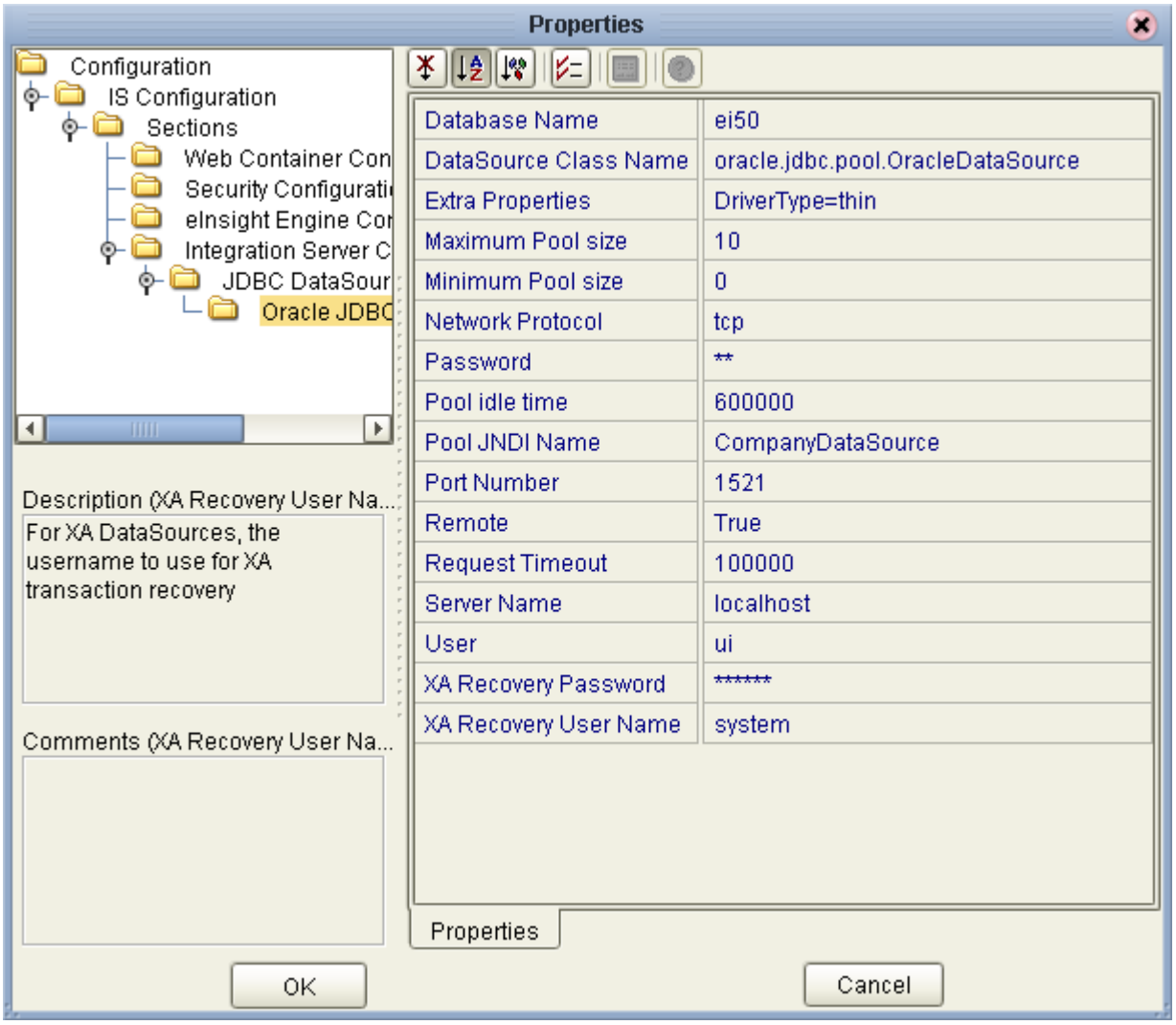
1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the Integration Server icon in the Environment you want to configure.
3. Right-click the icon to display the **Integration Server** context menu.
4. Select **Properties** to display the **Properties** window.
5. Expand the configuration list in the left pane until **JDBC DataSource Connection Pools** is visible.
6. Right-click **JDBC DataSource Connection Pools** and select **Create New Section**.

7 Rename the new section.

8 Define each property in the right portion of the window with information specific to eIndex database you created. For more information about the properties on this window, see **Table 4 on page 99**.

Note: *Creating the database is described in* **Chapter 7** *of this guide. Use the information for that database for the properties on this window. You may need to close and re-open this window in order for the fields to display as illustrated and described below.*

Figure 36 Integration Server Properties



9 Click **OK** to close the **Properties** dialog.

Important: *For the* ***Pool JNDI Name*** *property, you must enter* ***PersonDataSource****. To see a description of each property you can configure, select a property and then view the description in the* ***Description*** *box in the left portion of the window.*

Table 4 Data Source Properties

Property	Description
Database Name	The SID name of the eIndex database.
DataSource Class Name	The Java class in the JDBC driver that is used to implement the ConnectionPoolDataSource interface. Default: **oracle.jdbc.pool.OracleConnectionPoolDataSource**
Extra Properties	Use this property field to define additional data source properties. You must specify the driver type for the application (for example, "DriverType=thin").
Maximum Pool size	The maximum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there is no maximum.
Minimum Pool size	The minimum number of physical connections the pool should keep available at all times. 0 (zero) indicates that there should be no physical connections in the pool and new connections should be created as needed.
Network Protocol	The network protocol used to communicate with the server.
Password	The logon password associated with the user logon ID specified in the **user** parameter below.
Pool idle time	The maximum number of seconds that a physical connection may remain unused before it is closed. 0 (zero) indicates that there is no limit.
Pool JNDI Name	The JNDI name for the data source. This must be **<*Object*>DataSource**, where <*Object*> is the name of the eIndex application (Person).
Port Number	The port number used to connect to the database. Typically, for Oracle this is **1521**.
Remote	An indicator of whether the database can be accessed remotely. Specify **true** for remote access; specify **false** if you do not want to allow remote access.
Request Timeout	The number of seconds driver will wait before attempting to log in to the database before timing out.
Server Name	The name of the server on which the eIndex database resides. Specify **localhost** only if the database resides on the integration server.
User	The user logon ID that will be used to connect to the database from the EDM and external connections.
XA Recovery Password	The database administrator password.
XA Recovery User	The database administrator logon ID.

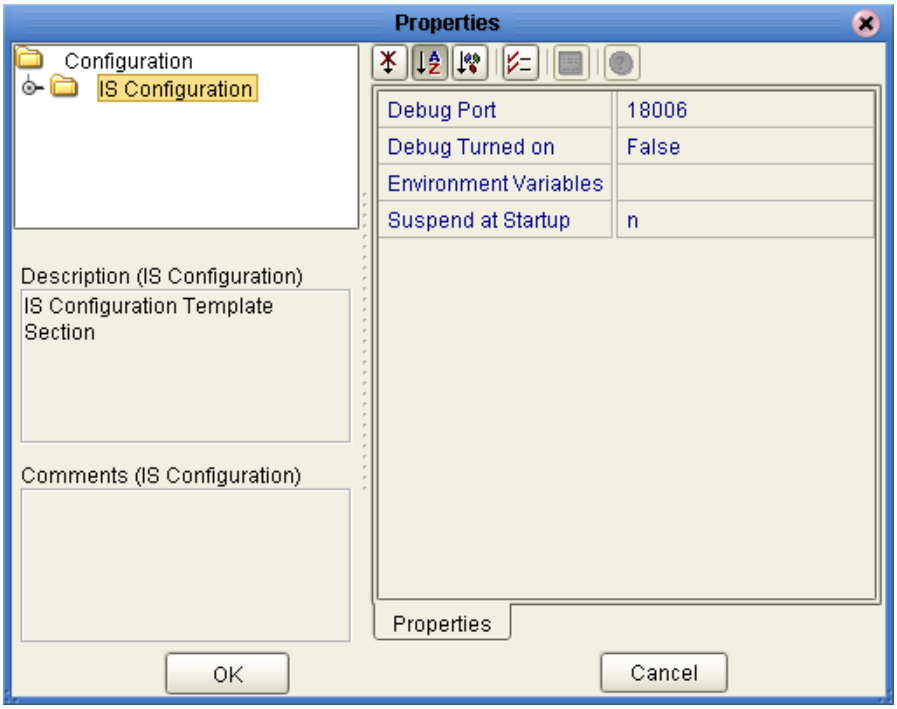
Defining Environment Variables for INTEGRITY

If you are implementing the INTEGRITY matching algorithm instead of the SeeBeyond Match Engine, you must define certain variables to help the application use the INTEGRITY rule set and library files.

To define environment variables

1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the SeeBeyond Integration Server icon in the Environment you want to configure.
3. Right-click the icon to display the **Integration Server** context menu.
4. Select **Properties** to display the **Properties** window.
5. In the configuration list in the left pane, select IS Configuration to display the configuration properties, as shown in Figure 37.

Figure 37 Integration Server Configuration Properties

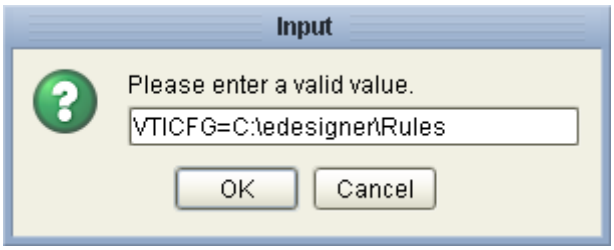


6. Click in the **Environment Variables** field, and then click the ellipses to the right of the field.

   The **Environment Variables** properties dialog appears.

7. For each environment variable listed in **Table 5 on page 102**, do the following:
   - A On the **Environment Variables** properties dialog, click **Add**.
   - B In the **Input** dialog, enter the name of the environment variable and the definition, as shown in Figure 38.

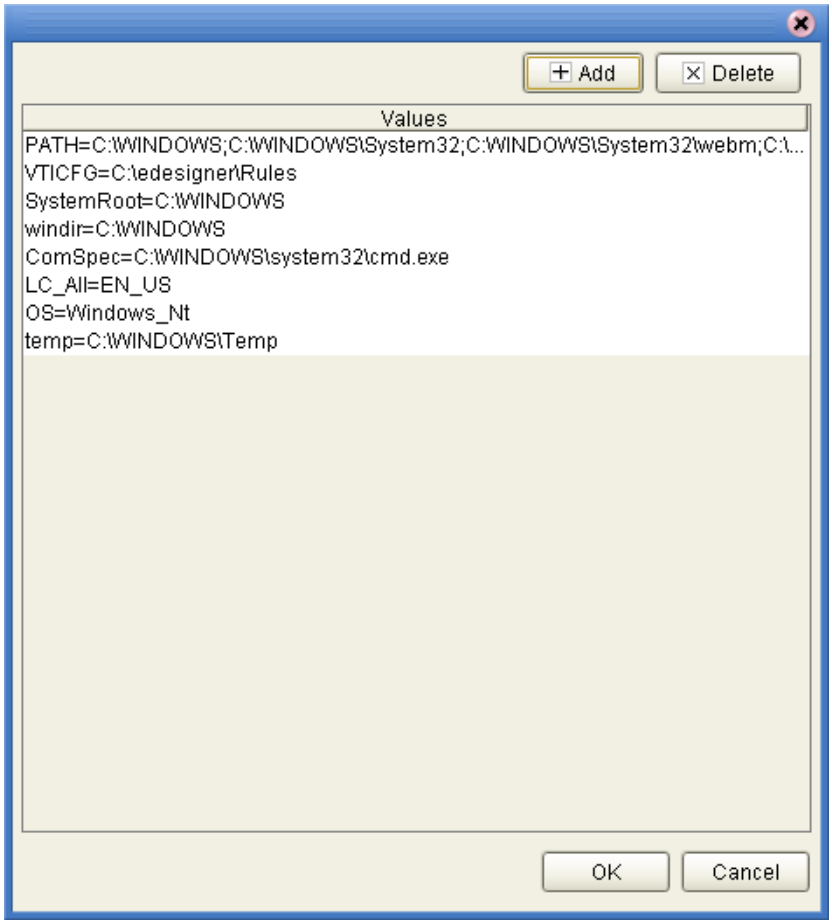
Figure 38 Input dialog



C On the Input dialog, click **OK**.

After you defined all variables, the window should look similar to Figure 39.

Figure 39 Defined Environment Variables



8 When you have defined all environment variables, click **OK** on the **Environment Variables** properties dialog.

9 On the IS Configuration **Properties** window, click **OK** to close the window.

Note: *In the following table,* ***<OS>*** *represents the operating system directory (such as Windows or WINNT),* ***<edesigner>*** *represents the Enterprise Designer home*

directory, ***<logicalhost>*** *represents the Logical Host home directory, and* ***<integrity>*** *represents the directory to which the INTEGRITY add-on files were extracted). Make sure to include a drive designation for Windows paths.*

Table 5 INTEGRITY Environment Variable Definitions

Set this environment variable …	to this value …
PATH	**Windows:** <*OS*>;<*OS*>\system32; <*OS*>\system32\webm;<*integrity*>\lib\win32; <*logicalhost*>\jre\bin **UNIX:**<*logicalhost*>/jre/bin
LD_LIBRARY_PATH	**Sparc Solaris:** <*integrity*>/lib/solaris
SHLIB_PATH	**HP UNIX:** <*integrity*>/lib/hpux
LIBPATH	**AIX:** <*integrity*>/lib/aix
VTICFG	**Windows and UNIX:** <*integrity*>\Rules
LC_All	**Windows and UNIX:** EN_US
OS	**Windows and UNIX:** Set this variable to the operating system of the server.
temp	**Windows and UNIX:** <*location of the temporary directory on the server* >
SystemRoot	**Windows only:** <*OS*>
windir	**Windows only:** <*OS*>
ComSpec	**Windows only:** <*OS*>\system32\cmd.exe

9.2.7. Defining Security

A secure user name and password must be defined for the integration server in order to connect to the eIndex database. For each user you define, you must also specify a security role or roles in order for that user to be able to perform any functions in the Enterprise Data Manager.

To define security

1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the eIndex Environment icon.
3. Right-click the eIndex Environment icon to display the **Environment** context menu.
4. Select **User Management** to display the **User Management** dialog as shown in Figure 40.
5. Enter the user name, password, and confirmation password of the eIndex user.

Figure 40 User Management

6 On the **User Management** dialog, click **Add Role**.

7 In the **Role** dialog, do the following:

A Click **Create Role**, enter the user role name, and then click **OK**.

B Select the newly created user role, and then click **OK**.

eIndex user roles are listed and described in Table 6.

8 Repeat step 7 for each role to which you want to assign the user.

9 Click **OK** and then **Close** to close the **User Management** dialog.

Table 6 User Roles and Descriptions

User Role	Description
eView.Admin	Gives access permission to all functions of the Enterprise Data Manager.
eView.User	Gives access to the EDM. This role must be assigned to each user except those assigned the eView.Admin role.
eView.VIP	Gives permission to view fields masked by the VIP feature.
AL.View	Gives access permission to view audit log entries.

Table 6 User Roles and Descriptions

User Role	Description
Duplicate.All	Gives access permission to all potential duplicate functions.
Duplicate.SearchAndView	Gives access permission to search for and view potential duplicate records.
Duplicate.Print	Reserved for future functionality.
Duplicate.Unresolve	Gives access permission to unresolve potential duplicate records that were previously resolved.
Duplicate.Resolve	Gives access permission to resolve potential duplicate records.
Duplicate.AutoResolve	Gives access permission to permanently resolve potential duplicate records.
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.
EO.Unmerge	Gives access permission to unmerge enterprise records.
EO.ViewMergeTree	Gives access permission to view a merge history of an enterprise object.
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.

Table 6 User Roles and Descriptions

User Role	Description
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.Print	Reserved for future functionality.
SO.Merge	Gives access permission to merge system records.
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.

Chapter 10

Deploying the Project

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

10.1 Overview

The Deployment Profile binds the eIndex Project attributes to the Environment that defines where each component runs. For example, the Deployment Profile defines which Integration Server runs eIndex. The Deployment Profiles for the client Projects that use eIndex Components define which JMS IQ Managers host which topics, which External Systems are connected toeIndex via which Ways, and so on.

10.2 Deploying a Project

To deploy a project, you must perform the following steps.

- **"Defining a Deployment Profile" on page 106**
- **"Activating the Project" on page 114**
- **"Running the Bootstrap" on page 115**

10.2.1. Defining a Deployment Profile

A Deployment Profile maps Project components to the Environment. You need to create a Deployment Profile for the eIndex application Project and for any client Projects that connect to eIndex. You can use the same Environment components for each Deployment Profile.

Defining an eIndex Application Deployment Profile

This section describes how to add a Deployment Profile to the eIndex application Project. This consists of two primary steps: creating the Deployment Profile and mapping the Project components to the Environment components.

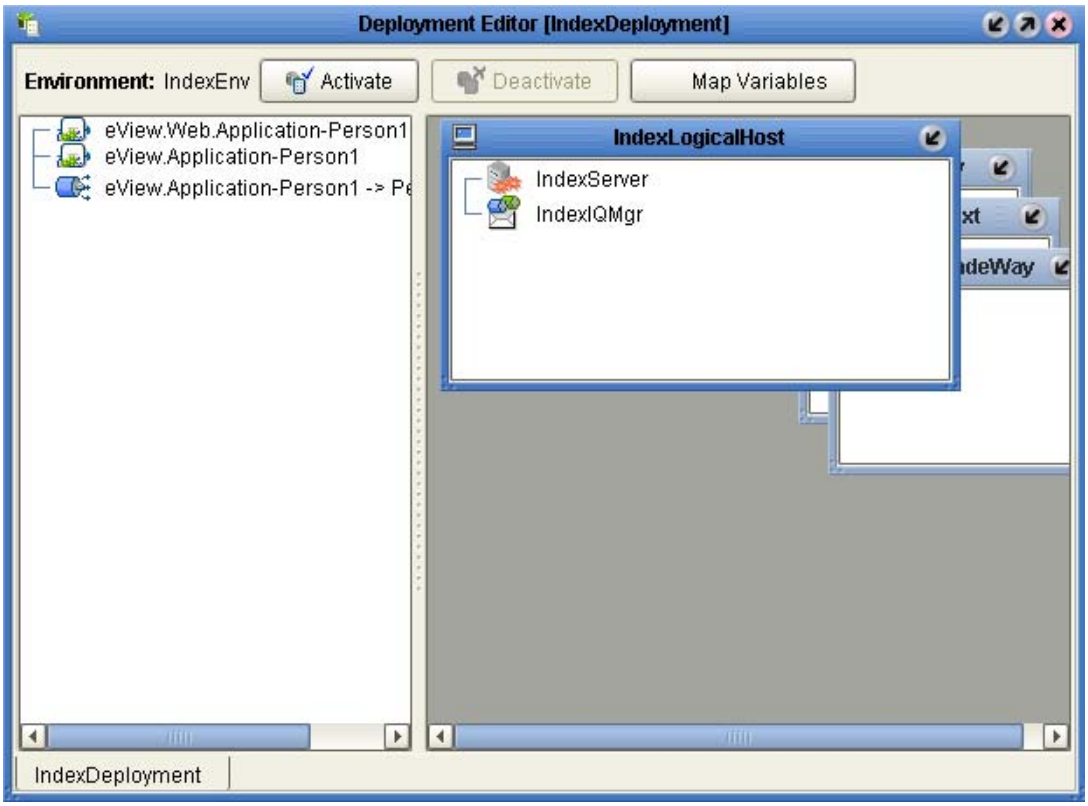
To create an eIndex application Deployment Profile

1. In Enterprise Explorer, click the **Project Explorer** tab.
2. Select the eIndex 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 41.

Figure 41 Create Deployment Profile Dialog Box

5. In the **Deployment Profile Name** field, type a name for the Deployment Profile.
6. In the **Environments** drop-down list, select the Environment you created for the eIndex Project.
7. Click **OK** to add a **Deployment Profile** icon to the eIndex Project and display the Deployment Editor window shown in Figure 42.

Figure 42 Deployment Editor Window - Server Project



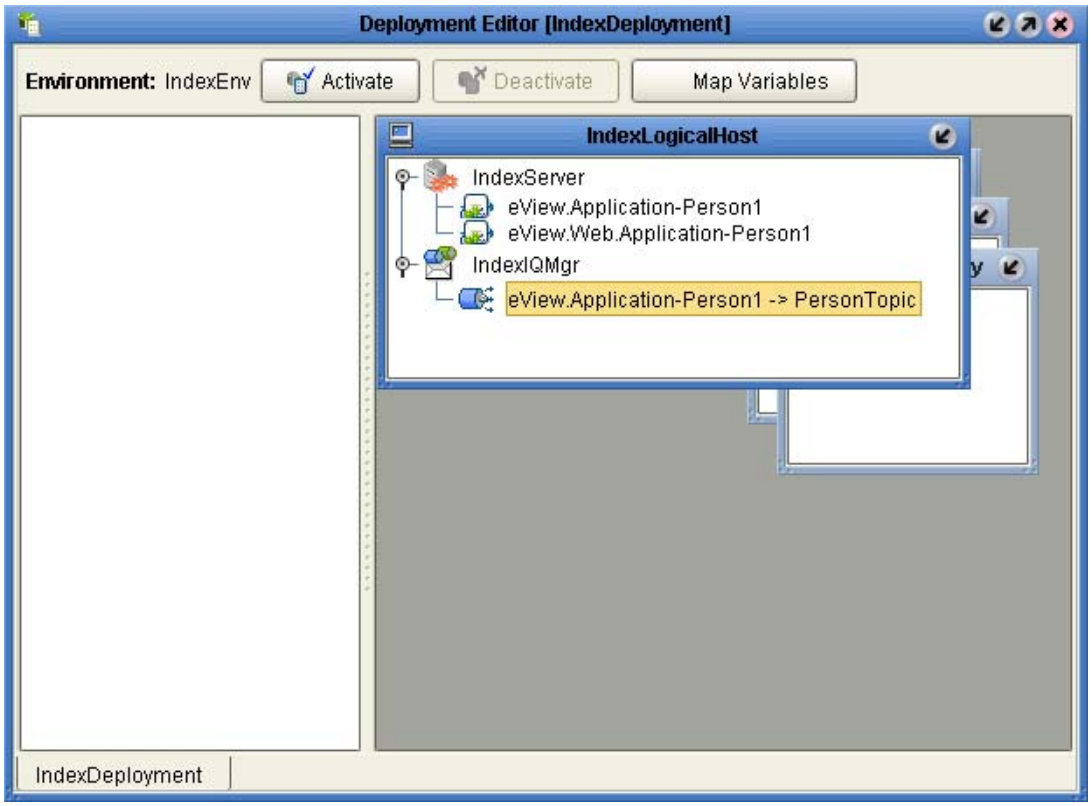
To map eIndex Project components

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.

1. With the eIndex Project Deployment Profile open in the Deployment Editor, drag the **eView.Web.Application-Person** and **eView.Application-Person** icons onto the SeeBeyond Integration Server in the Deployment Editor (the servers appear in the Logical Host).
2. If you defined a JMS Topic to publish eIndex messages, drag the JMS Topic icon to the JMS IQ Manager in the Logical Host.

   Figure 43 illustrates the updated Deployment Profile.

Figure 43 Mapped eIndex Components in the Deployment Editor



3 Deploy the application, as described in **"Activating the Project" on page 114**.

Creating an eIndex Client Deployment Profile

This section describes how to create a Deployment Profile for the Projects defining external system connections to the eIndex application. This consists of two primary steps: creating the Deployment Profile and mapping the Project components to the Environment components.

To create an eIndex client Deployment Profile

1 In Enterprise Explorer, click the **Project Explorer** tab.

2 Select the eIndex 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 44.

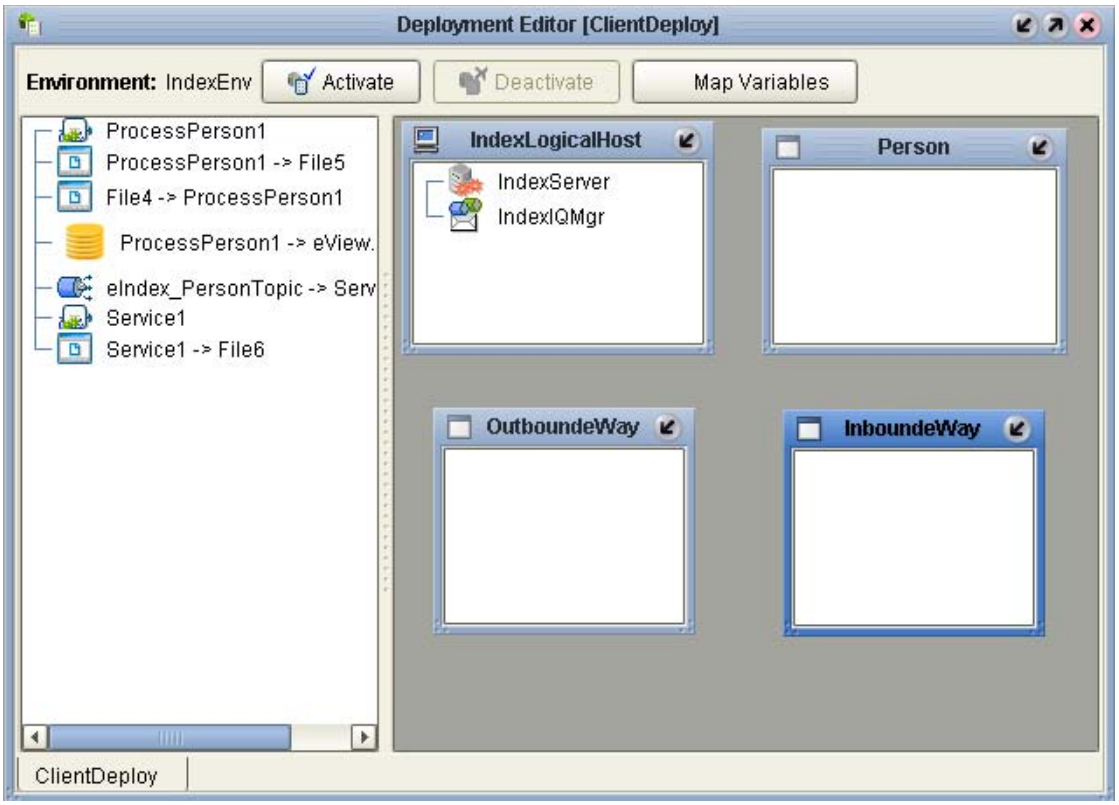
Figure 44 Create Deployment Profile Dialog Box

5 In the **Deployment Profile Name** field, type a name for the Deployment Profile.

6 In the **Environments** drop-down list, select the Environment you created for the eIndex Project.

7 Click **OK** to add a **Deployment Profile** icon to the eIndex Project and display the Deployment Editor window shown in Figure 45.

Figure 45 Deployment Editor Window - Client Project



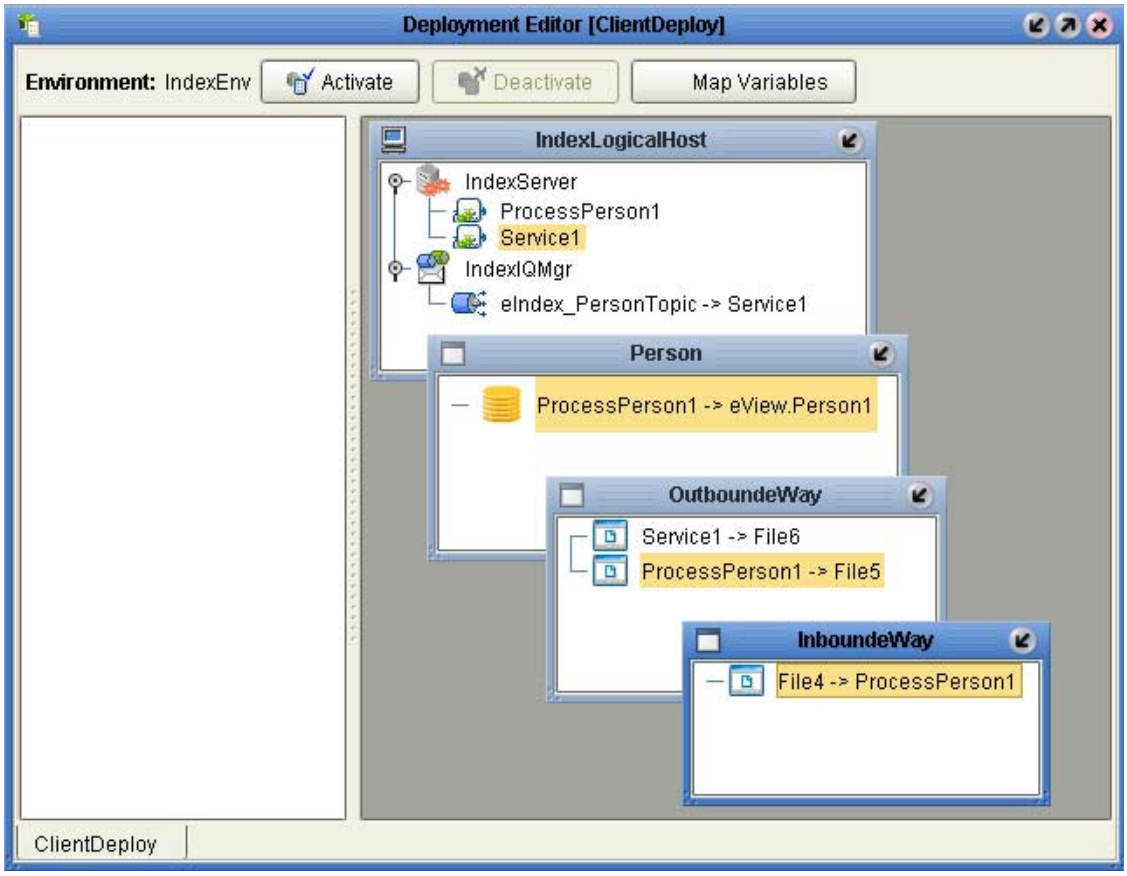
Note: *Your Deployment Editor may differ from the above illustration depending on whether you implementing a JMS Topic for processing outbound messages.*

To map eIndex 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.

1. With eIndex Project Deployment Profile open in the Deployment Editor, drag the Service icon(s) onto the SeeBeyond Integration Server in the Deployment Editor (the servers appear in the Logical Host).
2. Drag the eIndex application icon onto the eIndex application deployment component (this is named after the eIndex application).
3. Drag the external system eWays to the appropriate inbound and outbound deployment components (in Figure 46, the inbound File eWay is placed in **InboundeWay** and the outbound File eWays are placed in **OutboundeWay**).
4. If you implemented a JMS Topic, drag the topic to the JMS IQ Manager in the Logical Host.

Figure 46 Mapped Client Components in the Deployment Editor



5. Deploy the application, as described in **"Activating the Project" on page 114**.

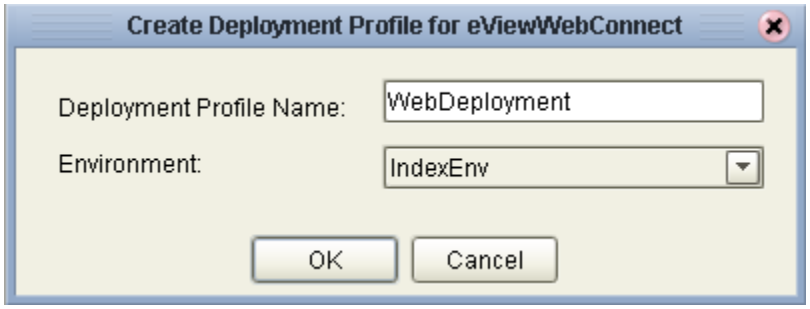
Creating an eInsight Integration Deployment Profile

This section describes how to create a Deployment Profile for the Projects defining Web pages for eIndex using eInsight integration. This consists of two primary steps: creating the Deployment Profile and mapping the Project components to the Environment components.

To create a Deployment Profile for eInsight Integration

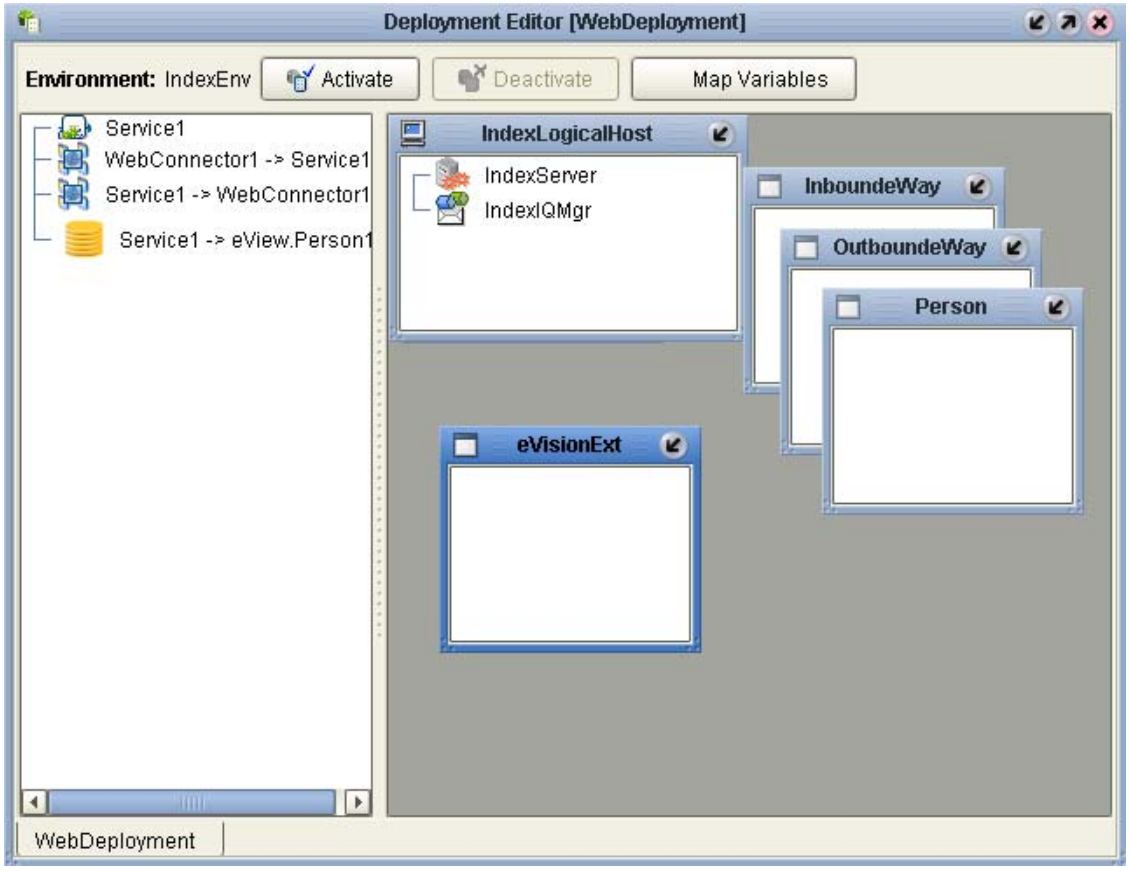
1. In Enterprise Explorer, click the **Project Explorer** tab.
2. Select the eIndex Project folder.
3. Right-click the mouse to launch the **Project** context menu.
4. Select **New > Deployment Profile** to display the **Create Deployment Profile** dialog shown in Figure 47.

Figure 47 Create Deployment Profile Dialog Box



5. In the **Deployment Profile Name** field, type a name for the Deployment Profile.
6. In the **Environments** drop-down list, select the Environment you created for the eIndex Project.
7. Click **OK** to add a **Deployment Profile** icon to the eIndex Project and display the Deployment Editor window shown in Figure 48.

Figure 48 Deployment Editor Window



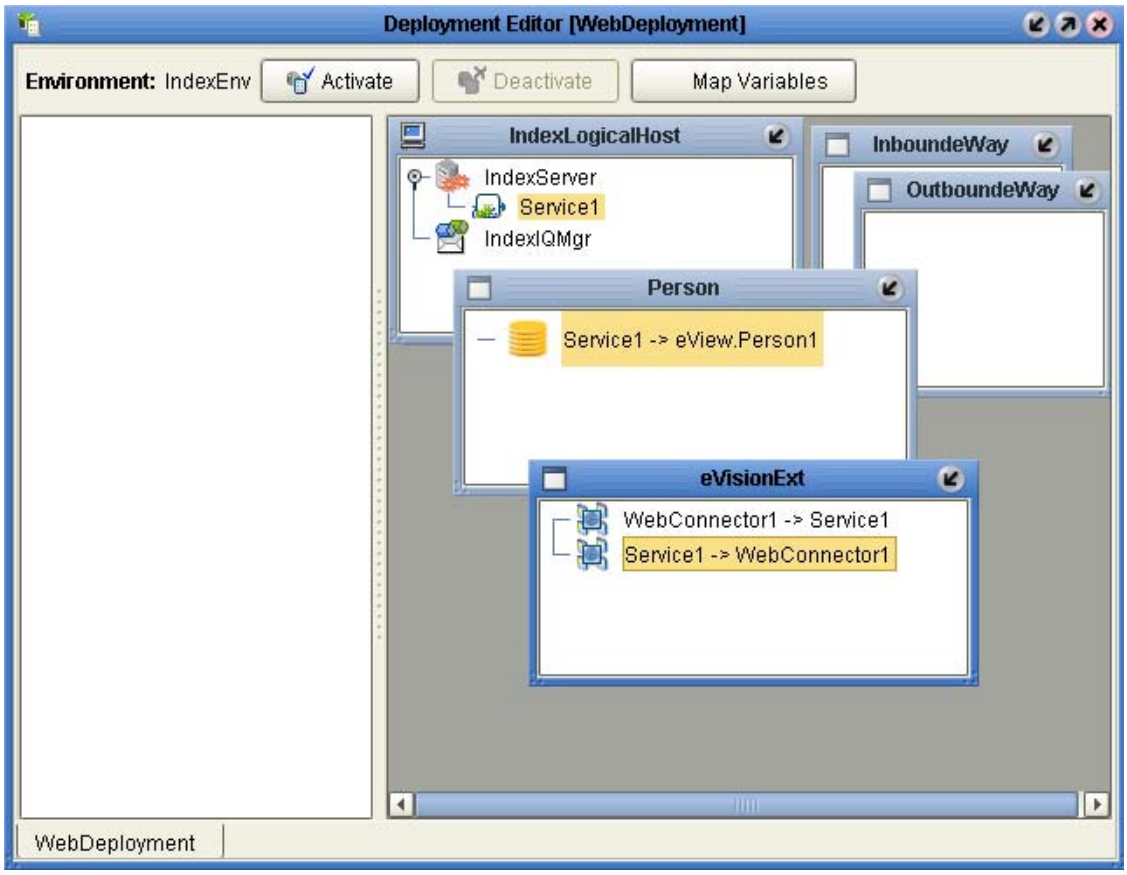
To map eInsight Integration Project components

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.

1. With the eInsight integration Project Deployment Profile open in the Deployment Editor, drag the Service icon onto the SeeBeyond Integration Server in the Deployment Editor (the servers appear in the Logical Host).
2. Drag the eIndex application icon onto the eIndex application deployment component (this is named after the eIndex application).
3. Drag the eVision Web service(s) to the eVision External Application in the Environment.

   Figure 49 illustrates the updated Deployment Profile.

Figure 49 Mapped Client Components in the Deployment Editor



4 Deploy the application, as described in **"Activating the Project" on page 114**.

10.2.2. Activating the Project

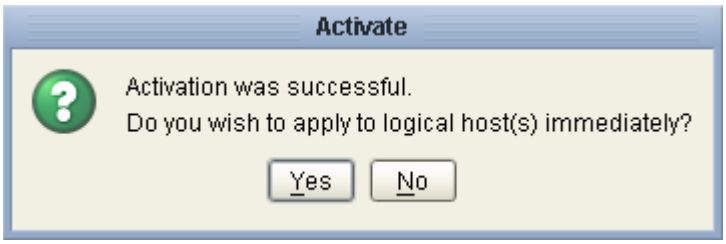
With Project components mapped in the Deployment Profile, you are now ready to activate the Project.

To activate a Project

1 In the Project, select the Deployment Profile you wish to activate.

2 Click the **Activate** button. After the activation is successful, the Activate dialog appears as shown in Figure 50.

Figure 50 Activate Dialog



3 Answer the question appropriately.

- If you wish to apply the changes immediately, click **Yes**.
- If you wish to apply the changes at a later time, click **No**. (To apply the changes at a later time, right-click the Logical Host and click **Apply**. This will apply all of the changes for that Logical Host.)

Note: *The Project must be properly configured for successful activation. Deploying the eIndex Project before creating client Project Deployment Profiles makes the eIndex application available to those Profiles.*

10.2.3. Running the Bootstrap

To run eIndex, you must run the bootstrap for the Logical Host to which the application was deployed. If any client Projects connected to eIndex are deployed on a different Logical Host, you must run a bootstrap for that Logical Host as well. For information about the bootstrap process and instructions for running the bootstrap, see the *eGate Integrator User's Guide*.

Appendix A

Field Notations

The configuration files use specific notations to define a specific field in an enterprise or system object. This appendix describes each type of notation used.

1.3 Defining Field Locations

There are three different type of notations used to specify a specific field or group of fields in the eIndex configuration files. They are ePath, qualified field name, and simple field name.

1.3.1. 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**
  An *ObjectField* represents a field defined in the eIndex object structure.
- **ObjectNode**
  An *ObjectNode* represents a parent or child object defined in the eIndex object structure.
- **ObjectField List**
  An *ObjectField List* is a list of references to certain ObjectFields in the eIndex object structure.
- **ObjectNode List**
  An *ObjectNode List* is 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
```

- **Nodes**
  The 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. "node.*" is used to address a node rather than a field.
- **Qualifiers**
  Qualifiers restrict 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 (','). For example, **[@key1=value1,@key2=value2]**.
  - **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]**.
- **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 listed that refer to various elements of this object structure. A description of the data in the sample object structure referred by the ePath is also included.

```
Enterprise
    SystemObject - A 111
        Person
            FirstName
            LastName
            -Address
                AddressType = Home
                Street = 404 E. Huntington 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 Marine Parkway
                City = Redwood Shores
                State = CA
                PostalCode = 94065
```

- **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”.
- **Person.Address[City=Monrovia,State=CA].Street**
  Returns a list of ObjectField references: “404 E. Huntington Dr.”, “181 E. Huntington Dr.”. Note that a reference to the **Billing** address is not returned.
- **Person.Address[*].Street**
  Returns a list of ObjectField references: “404 E. Huntington 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 ObjectField.

1.3.2. 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**—This type 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**—This type 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**

1.3.3. 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 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**

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, (depending on the configuration of matching parameters) the objects are an assumed match and are merged automatically (see "Automatic Merge").

automatic merge
When two records are assumed to be matches of one another (see "Assumed Match"), the system performs an automatic merge to join the records rather than flagging them as potential duplicates.

Blocking Query
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. The criteria is defined in the Candidate Select file.

Candidate Select file
The eIndex 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.

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 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 Manager Service
An eIndex component that provides an interface to all eIndex 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 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 configuration file that specifies any default or custom Java classes that perform field validations when data is processed.

local ID
A unique identification code assigned to an member in a specific local system. A member profile may have several local IDs in different systems.

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 configuration file that defines normalization, parsing, phonetic encoding, and the match string for an instance of eIndex. 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.

match type
An indicator specified in the **MatchingConfig** section of the Match Field configuration file that tells the match engine which rules to use to match information.

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.

Matching Service
An eIndex 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.

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 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. 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 fields from which source systems should be 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 fields 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 "source 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 EDM tab on the Define Enterprise Object window to display the EDM attributes.

Threshold file
An eIndex 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

B

C

D

E

F

G

I

J

L

M

O

P

Q

R

S

T

U

W

X