SeeBeyond ICAN Suite

eView Studio User's Guide

Release 5.0

SEEBEYOND®

Version 20031014184140.

Contents

List of Figures

Chapter 1

Introduction

This guide provides comprehensive information on using the SeeBeyond® eView Studio (eView) to design, configure, and create a customized enterprise-wide master index. As a component of SeeBeyond's Integrated Composite Application Network (ICAN) Suite, eView helps you integrate information from disparate systems throughout your organization. This guide explains how to install eView and to create and configure the components of a master index, including eGate project files, the index database, the runtime environment, and the Enterprise Data Manager (EDM). It also includes information about the eView and master index structure. This guide is intended to be used with the *eView Studio Configuration Guide* and *Implementing the SeeBeyond Match Engine with eView Studio* or *Implementing Ascential INTEGRITY with eView Studio*.

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 eView and creating a master index application. It includes navigational information, functional instructions, and background information where required. A summary of activities for creating a master index is provided in **Chapter 2** of this guide.

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 15**).

1.1.1. Intended Audience

Any user who installs any component of eView, or designs and creates a master index using eView, should read this guide. A thorough knowledge of eView is not needed to understand this guide. It is presumed that the reader of this guide is familiar with the eGate environment and GUIs, eGate projects, Oracle database administration, and the operating system(s) on which eGate and the index database run. Readers who will configure the master index should also be familiar with XML documents, the SQL scripting language, and Java.

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

1.1.2. 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 twelve chapters and two appendixes 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 **"eView Studio Overview"** provides information about the architecture of eView and the master index, and describes how a master index is created.
- Chapter 3 **"Installation"** gives instructions for installing the eView files and setting up the environment for eView and the master index.
- Chapter 4 **"Creating the Master Index Framework"** describes how to create the object definition file and configure the definition of the primary object you are storing in the master index.
- Chapter 5 **"Configuring the Master Index"** gives a summary of the configuration files in the eView Project, and provides information about the XML Editor.
- Chapter 6 **"Generating the Project"** explains how to generate the eView application to create a master index from the files you created and customized.
- Chapter 7 **"Creating Custom Plug-ins"** describes how to implement custom processing code in the master index.
- Chapter 8 **"Creating the Database"** describes how to design, install, and configure the Oracle database for the master index.
- Chapter 9 **"Defining Connectivity Components"** describes how to work with the connectivity components of the eView Project to share and process data through the master index system.
- Chapter 10 **"Defining the Environment"** describes how to set up the physical environment of the master index.
- Chapter 11 **"Deploying the Project"** explains how to create the Deployment Profile for the master index and how to activate the profile.
- Chapter 12 **"Implementing the eView Sample"** explains how to work with the eView sample to run data into the database through a Service, and view the information in the EDM.
- Appendix A **"Field Notations"** describes the different notations used in the configuration files to define different fields in the messages being processed and stored.

- Appendix B **"eView Wizard Match Types"** describes the different match types you can select from the eView Wizard and how they define matching and standardization processing logic.

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	*eView Studio 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 14.
File Names and Paths	**Bold** text	To install eView, upload the **eView.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 eView 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 eView. 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 eView 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 eView Java API is provided in Javadoc format.

- *Enterprise Data Manager User's Guide*
- *eView Studio Configuration Guide*
- *eView Studio Reference Guide*
- *Implementing the SeeBeyond Match Engine with eView Studio*
- *Implementing Ascential INTEGRITY with eView Studio*
- *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

eView Studio Overview

This chapter provides information about eView, how it is used to create a master index, and the master index applications you create with eView. It also includes a description of the files stored in the eGate Repository and the XML files that define the structure and configuration of the master index environment.

2.1 Learning about eView

2.1.1. Overview

eView provides a flexible framework to allow you to create matching and indexing applications called enterprise-wide master indexes (or just *master indexes*). It is an application building tool to help you design, configure, and create a master index that will uniquely identify and cross-reference the business objects stored in your system databases. Business objects can be any type of entity for which you store information, such as customers, members, vendors, businesses, hardware parts, and so on. In eView, you define the data structure of the business objects to be stored and cross-referenced. In addition, you define the logic that determines how data is updated, standardized, weighted, and matched in the master index database.

The structure and logic you define is located in a group of XML configuration files that you create using the eView Wizard. These files are created within the context of an eGate Project, and can be further customized using the XML editor provided in the Enterprise Designer.

2.1.2. eView Features and Functions

eView provides features and functions to allow you to create and configure an enterprise-wide master index for any type of data. The primary function of eView is to automate the creation of a highly configurable master index application. eView provides a wizard to guide you through the initial setup steps, and various editors so you can further customize the configuration of the master index. eView automatically generates the components you need to implement a master index.

eView provides the following features:

- **Rapid Development**—eView allows for rapid and intuitive development of a master index using a wizard to create the master index configuration, and using

XML documents to configure the attributes of the index. Templates are provided for quick development of person and company object structures.

- **Automated Component Generation**—eView automatically creates the eView configuration files that define the primary attributes of the master index, including the configuration of the Enterprise Data Manager (EDM). eView also generates scripts that create the appropriate database schemas and an eGate Object Type Definition (OTD) based on the object definition you create and configure.
- **Configurable Survivor Calculator**—eView provides predefined strategies for determining which field values to populate in the single best record (SBR). You can define different survivor rules for each field, and you can create a custom survivor strategy to implement in the master index.
- **Flexible Architecture**—eView provides a flexible platform that allows you to create a master index for any business object. You can customize the object structure so the master index can match and store any type of data, allowing you to design an application that specifically meets your data processing needs.
- **Configurable Matching Algorithm**—eView provides support for both the SeeBeyond Match Engine and the Ascential™ INTEGRITY™ matching algorithm. In addition, you can plug in a custom matching algorithm to the master index.
- **Configurable Server Support**—The master index applications generated by eView support the SeeBeyond Integration Server (SIS).
- **Custom Java API**—eView generates a Java API that is customized to the object structure you define. You can call the methods in this API in the Collaborations of the eGate Project.

2.1.3. eView and the SeeBeyond ICAN Suite

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

eGate Integrator

eView 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 master index database and external systems by adding the eView 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 eView Project in Business Processes for eInsight, enabling access to the master index 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 master index databases.

2.2 eView Components

The components of eView are designed to work within the eGate Enterprise Designer to create and configure the master index, and to define connectivity between external systems and the master index. The primary components of eView are:

- eView Wizard
- Editors
- Project Components
- Environment Components

2.2.1. eView Wizard

The eView Wizard takes you through each step of the master index setup process, and creates the XML files that define the configuration of the application. The eView Wizard allows you to define the name of the master index, the objects to store, the fields in each object and their attributes, the EDM configuration, and the database and match engine platforms to use. The eView Wizard generates a set of configuration files and database scripts based on the information you specify. You can further customize these files as needed.

2.2.2. Editors

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

- **XML Editor**—allows you to review and customize the XML configuration files created by the eView Wizard. This editor provides verification services for XML syntax (schema validation is provided by eView). The XML editor is automatically launched when you open an eView configuration file.
- **Text Editor**—allows you to review and customize the database scripts created by the eView Wizard. This editor is very similar to the XML editor but without the verification services. The text editor is automatically launched when you open an eView database script.
- **Java Source Editor**—allows you to create and customize custom plug-in classes for the master index. 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.

2.2.3. Project Components

An eView master index is implemented within a Project in Enterprise Designer. When you create an eView application, a set of configuration files and a set of database files are generated based on the information you specified in the eView Wizard. When you generate the Project, additional components are created, including a method OTD, an outbound OTD, eInsight web page methods, necessary **.jar** files, and a Custom Plug-in function that allows you to define additional, custom processing for the index. To complete the Project, you create a Connectivity Map and Deployment Profile.

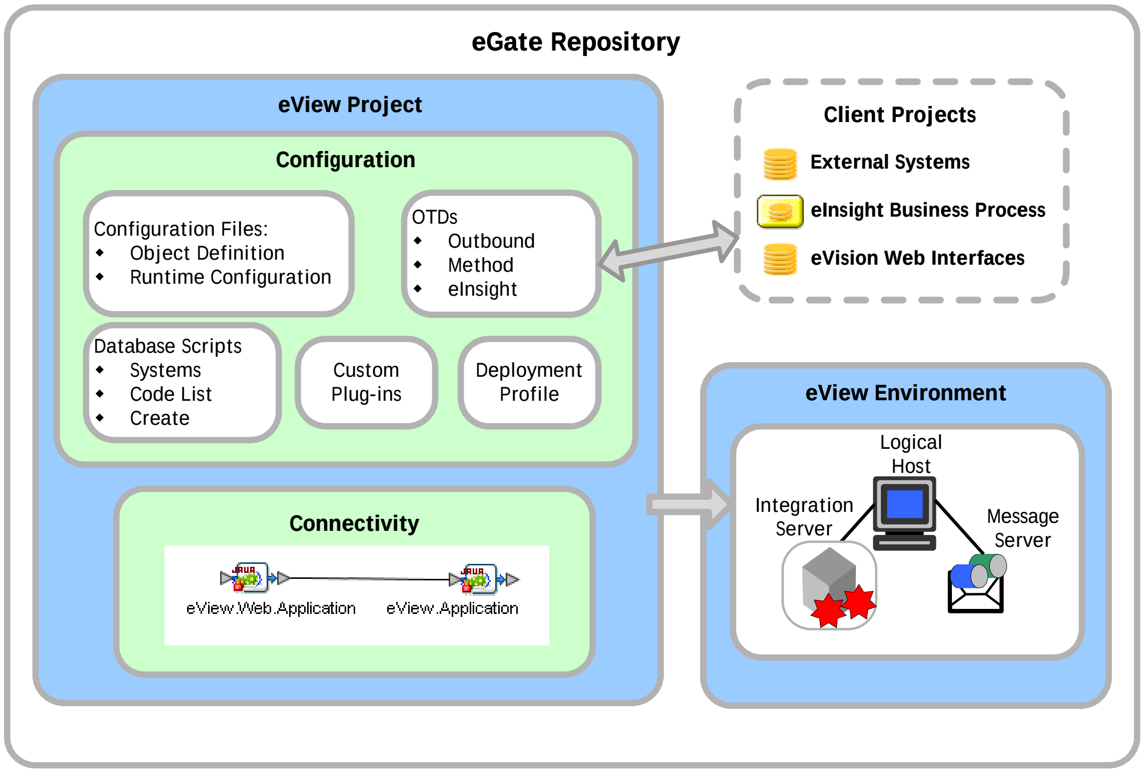
Additional eGate components must be added to the client Projects accessing the eView master index, including Services, Collaborations, OTDs, Web Connectors, eWays, JMS Queues, JMS Topics, and so on. You can use the standard Enterprise Designer editors, such as the OTD or Collaboration editors, to create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

Following is a list of eView 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 21 illustrates the Project and Environment components of eView Studio.

Figure 1 eView Project and Environment Components



Configuration Files

Several XML files together determine certain characteristics of the master index, such as how data is processed, queried, and matched. These files configure runtime components of the master index, which are listed in **"Master Index Components" on page 27**.

- **Object Definition**—Defines the data structure of the object being indexed in a master index.
- **Enterprise Data Manager**—Configures the search functions and appearance of the EDM, along with debug information and security information for authorization.
- **Candidate Select**—Configures the Query Builder component of the master index, and defines the queries available for the index.
- **Match Field**—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**—Configures the eView 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**—Configures the Update Manager, and defines the strategies used by the survivor calculator to determine the field values for the SBR. It also allows you to define custom update procedures.
- **Field Validation**—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

Two database scripts are generated by the eView Wizard: **Systems** and **Code List**. The additional scripts are created when you generate the Project (or by the wizard if you choose to create all Project files at once).

- **Systems**—Contains the SQL insert statements that add the external systems you specified in the eView Wizard to the database. You can define additional systems in this file. This file is executed after the create script is run.
- **Code List**—Contains the SQL statements to insert processing codes and drop-down list values into the database. You must define these elements in this file to make them available to the master index system.
- **Create database script**—Defines the structure of the master index database based on the object structure defined in the eView Wizard. You can customize this file, and then run it against an Oracle database to create a customized master index database. This file is named the same name as was specified for the eView application in the eView Wizard.
- **Drop database script**—Used primarily in testing, when you need to drop existing database tables and create new ones. The delete script removes all tables related to the master index so you can recreate a fresh database for your Project.

You can also create custom scripts to store in the eView Project and run against the master index database.

Custom Plug-ins

eView provides a method by which you can create custom processing logic for the master index. To do this, you need to define and name a custom plug-in, which is a Java class that performs the required functions. Once you create a custom plug-in, you incorporate it into the index by adding it to the appropriate configuration file. You can create custom update procedures and field validations, as well as define custom eView components. Update procedures must be referenced in the update policies of the Best Record file; field validations must be referenced in the Field Validation file; and custom components must be referenced in the configuration file for that component.

Match Engine Configuration Files

If you specified the SeeBeyond Match Engine in the eView Wizard, several configuration files for the engine are created in the eView 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 eView Studio*.

Outbound Object Type Definition (OTD)

eView generates 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 the master index to the external systems that share data with the master index. 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 the master index data available to external systems, you must define a JMS Topic in the eView Connectivity Map to which the master index can publish transactions.

Dynamic Java API

Due to the flexibility of the object structure, eView generates 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 one set is provided for Web services. The names, parameter types, and return types of these methods vary based on the objects you defined in the object structure. These methods are described in the *eView Studio Reference Guide*.

Method OTD

Generating the eView instance creates a method OTD containing Java functions you can use to define data processing rules in Collaborations. 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, eView creates 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 master index database from eInsight Web pages.

Connectivity Components

The eView Project Connectivity Map only consists of two components: the Web application file and the application file. However, in client Project Connectivity Maps you can use any of the standard Project components to define connectivity and data flow to and from the master index. Client Projects includes those created for the external systems sharing data with the index and for eVision Web pages. The eView

Connectivity Map may include one additional component, a JMS Topic, to which the master index can publish all processed messages for broadcasting to external systems.

For the client Projects, you can use connectivity components from the eView server Project and create any standard eGate connectivity components, such as OTDs, Services, Collaborations, JMS Queues, JMS Topics, and eWays. Client Project components transform and route incoming data into the master index 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 the master index. It contains information about the assignment of Services and message destinations to integration servers and JMS IQ Managers within the eView system. Each eView 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 custom master index you created using eView.

2.2.4. Environment Components

The eView Environments define configuration of the deployment environment of the master index, including the Logical Host and application server. If eView 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 one or more eView applications. You must define and configure at least one Environment for the master index before you can deploy the application. The integration server hosting the eView application 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 Learning about the Master Index

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. The master indexes you create from eView 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 structure of the business object and configure the runtime environment for the master index, the final product will include much of the same functions and features. The master index 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 Master Index

The master index 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 object by recording all changes to each object'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 an enterprise object.
- **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 object or a set of objects. 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 master index system is its ability to match records and identify possible duplicates. Using matching algorithm logic, the index 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 if you find them to be actual duplicates of one another. You can merge records at either the EUID or system record level. 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. Features of the Master Index

The components of the master index 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 index suited to your specific data processing needs. Primary features of the master index include:

- **Centralized Information**—The master index maintains a centralized database, enabling the integration of data records throughout the enterprise while allowing local systems to continue operating independently. The index stores copies of local system records and of SBRs, which represent the most accurate and complete data for each object. This database is the central location of information and identifiers, and is accessible throughout the enterprise.
- **Configurability**—Before deploying the master index, 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**—The master index is a global cross-indexing application that automates record-matching across disparate source systems, simplifying the process of sharing data between systems. The master index uses the local identifiers assigned by your existing systems as a reference for cross-indexing, allowing you to maintain your current systems and practices. The index maintains the most current information by providing accurate identification and cross-referencing of business objects.
- **Data Cleansing**—The master index uses configurable matching algorithm logic to uniquely identify object records, and to identify duplicate and potential duplicate records. The index provides the functionality to easily merge or resolve duplicates. The index can be configured to automatically merge records that are found to be duplicates of one another.
- **Data Updates**—The master index 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. Data updates from external systems can occur in real time or as batch processes.
- **Identification**—The master index 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 objects within an enterprise.
- **Integration**—Relying on the eGate Integrator, the master index 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**—The master index 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 the index to use the match engine of your choice.
- **Shared Information**—Each time a record is updated, added, merged, or unmerged from the user interface, the master index generates a message that can be transmitted to external systems. It also receives, processes, and broadcasts messages, containing information about the objects in your index.
- **Unique Identifier**—Records from various systems are cross-referenced using an enterprise-wide unique identifier, known as an EUID, that the index assigns to each object record. The index uses the EUID to cross-reference the local IDs assigned to each object by the various computer systems throughout the enterprise.

2.4 Master Index Components

The master index created by eView is made up of several components that work together to form the complete indexing system. The primary components of the master index are:

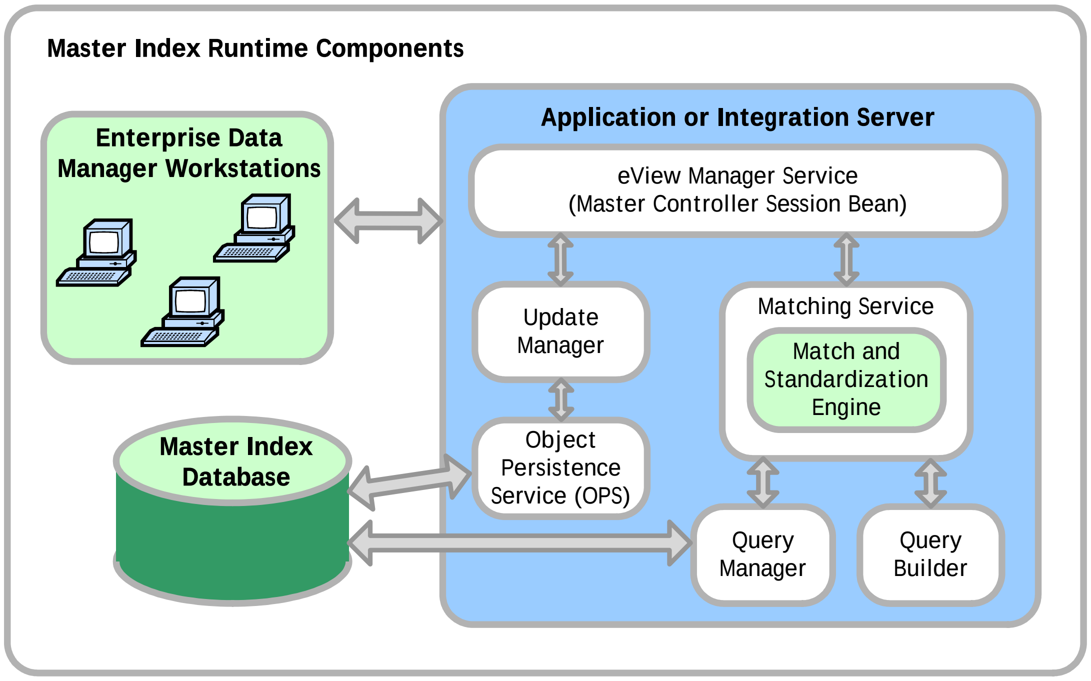
- eView Manager Service
- Matching Service
- Query Builder
- Query Manager
- Update Manager
- Object Persistence Service
- Database
- Enterprise Data Manager

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

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

The components of an eView master index are illustrated in Figure 2.

Figure 2 eView Master Index Architecture


Master Index Runtime Components
Enterprise Data Manager Workstations
Master Index Database
Application or Integration Server
eView Manager Service (Master Controller Session Bean)
Update Manager
Matching Service
Match and Standardization Engine
Object Persistence Service (OPS)
Query Manager
Query Builder

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. eView Manager Service

The eView Manager Service provides a session bean to all components of the master index, such as the Enterprise Data Manager, Query Builder, and Update Manager. The service also provides connectivity to the master index database. The configuration of the eView 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 eView Manager Service is defined in the *Threshold* file.

2.4.3. Query Builder

The Query Builder defines all queries available to the master index. This includes the queries performed automatically by the master index 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 master index database and returns a list of objects that match or closely match the query criteria. The Query Manager uses classes specified in the *Match Field* file to determine how to perform a query for match processing. All queries performed in the master index system are executed through the Query Manager.

2.4.5. Update Manager

The Update Manager controls how updates are made to an entity’s single best record (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

The master index uses an Oracle database to store the information you specify for the business objects being cross-referenced. The database stores local system records, the single best record for each object record, and certain administrative information, such as drop-down menu lists, processing codes, and information about the systems from which data originates. The scripts that are generated to create the database tables are based on the information specified 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 your master index database. Most of the configurable attributes of the EDM are defined by information you specify in the eView Wizard, but you can further configure the EDM in the Enterprise Data Manager file after you generate the eView 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 object records.

2.5 Enterprise Records

An *enterprise record* includes all components of a record that represents one entity. The master index stores two different types of records in each enterprise record: *system records* and a *single best record* (SBR). A system record contains an enterprise record'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 an enterprise record. 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 an object 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 eView, each system record and SBR in an enterprise record typically contain a set of objects that store different types of information about the business object. A record contains a parent object and several child objects. A record can have only one parent object, but can have multiple instances of each type of child object. For example, if the business object being indexed is a person, the record can only contain one member name and social security number, which would be contained in the parent object (for example, a *person* object). However, the record could have multiple addresses, telephone numbers, and aliases, which would each be defined in different child object types (for example, in *address*, *phone*, and *alias* objects).

2.6 From eView to the Master Index

The process of creating a master index begins with a thorough analysis of the data you plan to store in the index database and to share among the systems connected to the index. 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 a master index using the eView Studio.

1. Perform a thorough analysis of the data you plan to store in the master index.
2. Create an eGate Project, and create a new eView application within that Project (**Chapter 4**).
3. Define the object structure, operating environment, and certain runtime characteristics using the eView Wizard (**Chapter 4**).
4. If necessary, customize the configuration files (**Chapter 5**).

Note: *If you customize Object Definition, you may 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.

5. Generate the master index (**Chapter 6**).
6. Define and build custom plug-ins, and specify the plug-ins in the appropriate configuration file (**Chapter 7**).
7. Create the database (**Chapter 8**).
   - Customize the scripts by defining system information, processing codes, and drop-down menu values.
   - Create the database and then create any necessary indexes.
8. Create Connectivity Maps (**Chapter 9**).
   - Create and define the components in the Connectivity Maps, such as Collaborations, Services, and External Applications.
   - Configure the Connectivity Maps.
9. Define the Environment and configure the server and security (**Chapter 10**).
10. Create the Deployment Profile and activate the Project (**Chapter 11**).

2.6.2. From XML to the Database

The master index database is created using 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 your master index. 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 you specify in the eView Wizard. 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

When you generate the eView Project, several connectivity components are created, 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 the master index 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 eView 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

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

3.1 Installation Overview

In order to work with eView, you only need to perform the eView installation described in this chapter. To work with the master index created by eView, a second component, an Oracle database, must be installed for any master indexes you create.

3.1.1. eView Installation

eView is installed by uploading the eView files to the eGate Repository using the Enterprise Manager, and then installing the eView Module, eView Wizard, and eView Help to the Enterprise Designer. eView must be uploaded via an active eGate Repository, and the eView Module, eView Wizard, and eView 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 eView, 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. eView runs on the SeeBeyond Integration Server.

3.1.2. Database Installation

All of the master index components you create using eView are stored in the eGate Repository. The only external component is the master index database. The database does not need to be installed in order to use the eView tools, but it must be installed as a part of master index implementation. For optimal performance, the database should be installed on its own server. The master index 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. After you run the eView Wizard and generate the Project, several SQL scripts are created. Running these scripts against the database creates the master index tables and inserts startup data. For complete instructions on installing the database, see Chapter 8, **"Creating the Database"**.

3.2 About the Installation

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

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

You can also install these components:

- eView documentation in PDF format
- eView Javadocs
- eView sample Projects

3.3 System Requirements

eView 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 eView.

- 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 master index database, which is only required when you deploy a master index, are included in Chapter 8, **"Creating the Database"**. The client workstations accessing the master index 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 eView. eView 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 eView

Before you install eView, 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 eView. 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 eView in the Enterprise Manager and to update the Enterprise Designer through the Update Center.
5. Make sure the File eWay is installed. The eView sample Project 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 eView

To install eView, you must complete the following tasks:

- **Uploading eView to the Repository** on page 36
- **Installing eView in the Enterprise Designer** on page 42

3.5.1. Uploading eView to the Repository

The first step in installing eView 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 eView 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 eView files.

- **Uploading eView** on page 36
- **Uploading the eView Documentation and Sample** on page 39
- **Uploading the INTEGRITY Add-on** on page 41

Important: *Make sure eGate Integrator and Enterprise Designer are installed before performing these steps.*

Uploading eView

The first step is to upload the eView product files.

To upload eView

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:

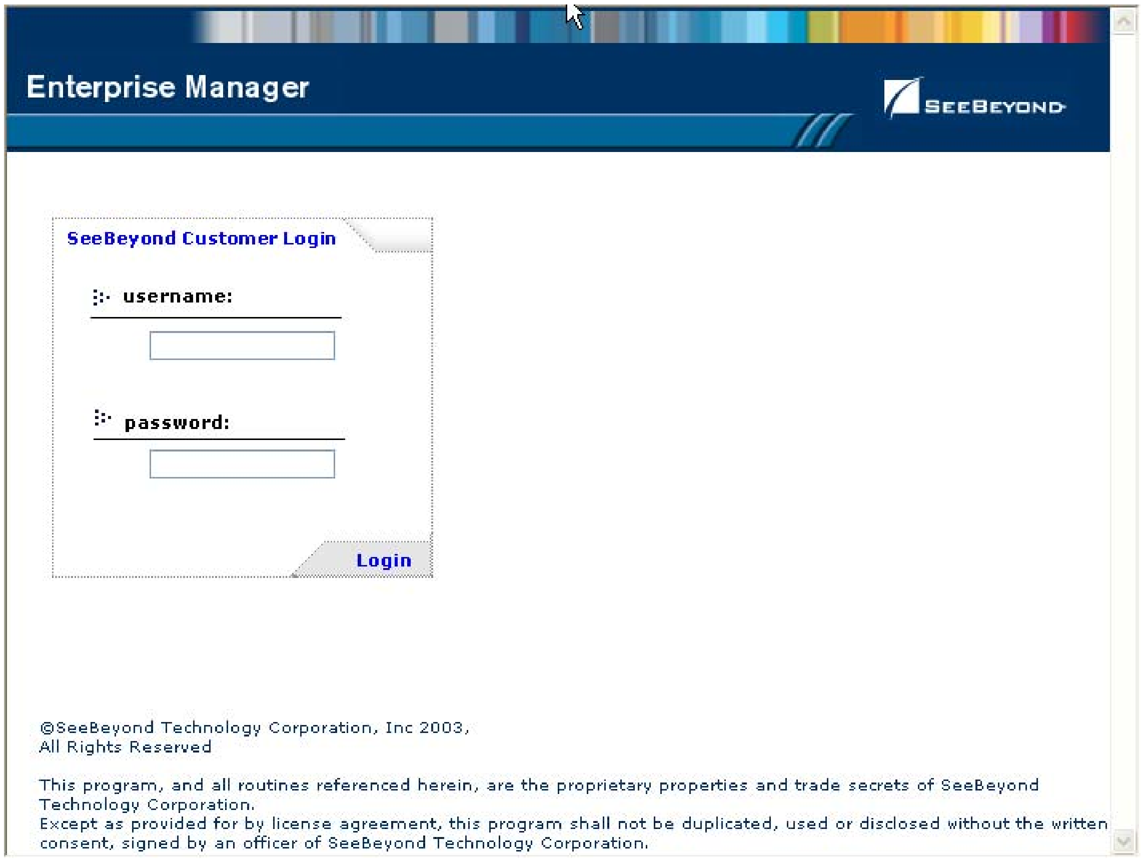
   <***hostname***> is the TCP/IP host name of the server where you installed the Repository—not the name of the Repository itself.

   <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



©SeeBeyond Technology Corporation, Inc 2003,
All Rights Reserved

This program, and all routines referenced herein, are the proprietary properties and trade secrets of SeeBeyond Technology Corporation.
Except as provided for by license agreement, this program shall not be duplicated, used or disclosed without the written consent, signed by an officer of SeeBeyond Technology Corporation.

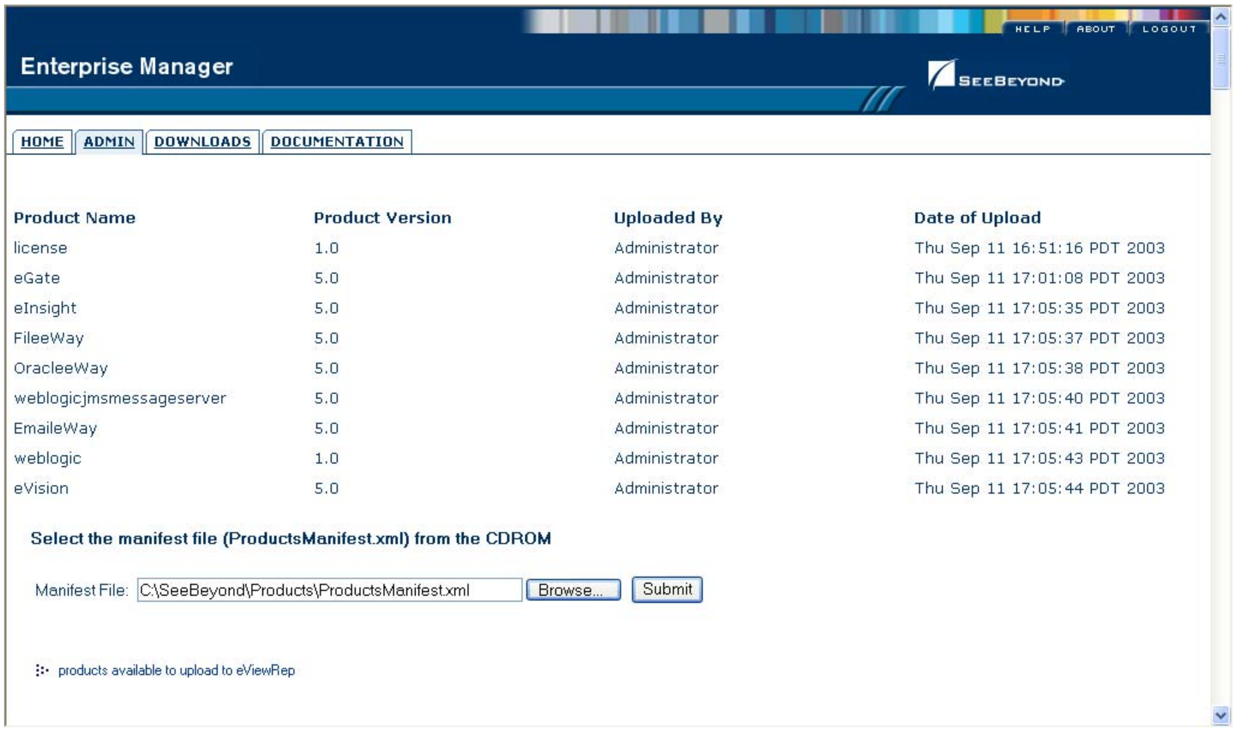
4 Enter your **username** and **password**.

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



Product Name	Product Version	Uploaded By	Date of Upload
license	1.0	Administrator	Thu Sep 11 16:51:16 PDT 2003
eGate	5.0	Administrator	Thu Sep 11 17:01:08 PDT 2003
eInsight	5.0	Administrator	Thu Sep 11 17:05:35 PDT 2003
FileeWay	5.0	Administrator	Thu Sep 11 17:05:37 PDT 2003
OracleeWay	5.0	Administrator	Thu Sep 11 17:05:38 PDT 2003
weblogicjmsmessageserver	5.0	Administrator	Thu Sep 11 17:05:40 PDT 2003
EmaileWay	5.0	Administrator	Thu Sep 11 17:05:41 PDT 2003
weblogic	1.0	Administrator	Thu Sep 11 17:05:43 PDT 2003
eVision	5.0	Administrator	Thu Sep 11 17:05:44 PDT 2003

Select the manifest file (ProductsManifest.xml) from the CDROM

Manifest File: C:\SeeBeyond\Products\ProductsManifest.xml

products available to upload to eViewRep

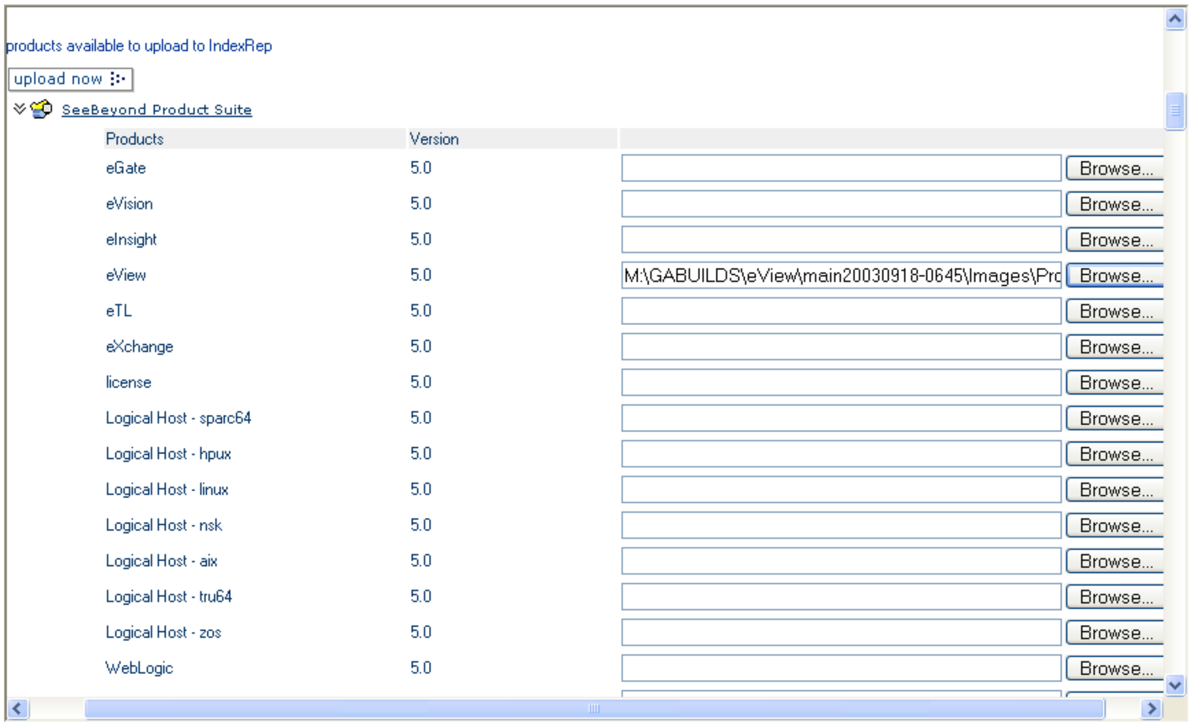
7 On the **ADMIN** page, do the following:

A Enter the products manifest file for eView (**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



products available to upload to IndexRep

upload now

SeeBeyond Product Suite

Products	Version	
eGate	5.0	
eVision	5.0	
eInsight	5.0	
eView	5.0	M:\GABUILDS\eView\main20030918-0645\Images\Prc
eTL	5.0	
eXchange	5.0	
license	5.0	
Logical Host - sparc64	5.0	
Logical Host - hpux	5.0	
Logical Host - linux	5.0	
Logical Host - nsk	5.0	
Logical Host - aix	5.0	
Logical Host - tru64	5.0	
Logical Host - zos	5.0	
WebLogic	5.0	

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

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

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

9 If you are installing documentation, samples, or the INTEGRITY add-on, leave the Enterprise Manager open and continue to **"Uploading the eView Documentation and Sample" on page 39** or **"Uploading the INTEGRITY Add-on" on page 41**.

Uploading the eView Documentation and Sample

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

To upload the eView documentation and sample

1 Complete the procedure described under **"Uploading eView" on page 36**.

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

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

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

B When you return to the **products available to upload to <*Repository_name*>** window, the **Products** box for eView 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 40
- **To download the Javadoc files** on page 40
- **To download the sample Project** on page 40
- **To upload the INTEGRITY Add-on** on page 41

To access the documentation files

1 In the Enterprise Manager, click the DOCUMENTATION tab.

2 Click **eView Studio**.

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 **eView Studio**.

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.

To download the sample Project

1 In the Enterprise Manager, click the DOCUMENTATION tab.

2 Click **eView Studio**.

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

4 A dialog appears prompting you to open the file to disk or save it to your computer. Choose **Save**, and then choose a location for the file set.

5 Extract **eView_Samples.zip** (make sure “Use folder names” is selected on the WinZip dialog).

6 To implement the sample Project, see **Chapter 12 “Implementing the eView Sample”**.

Uploading the INTEGRITY Add-on

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

To upload the INTEGRITY Add-on

1 Complete the procedure described under **“Uploading eView” on page 36**.

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 eView).

   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 eView is populated. Click **upload now**.

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

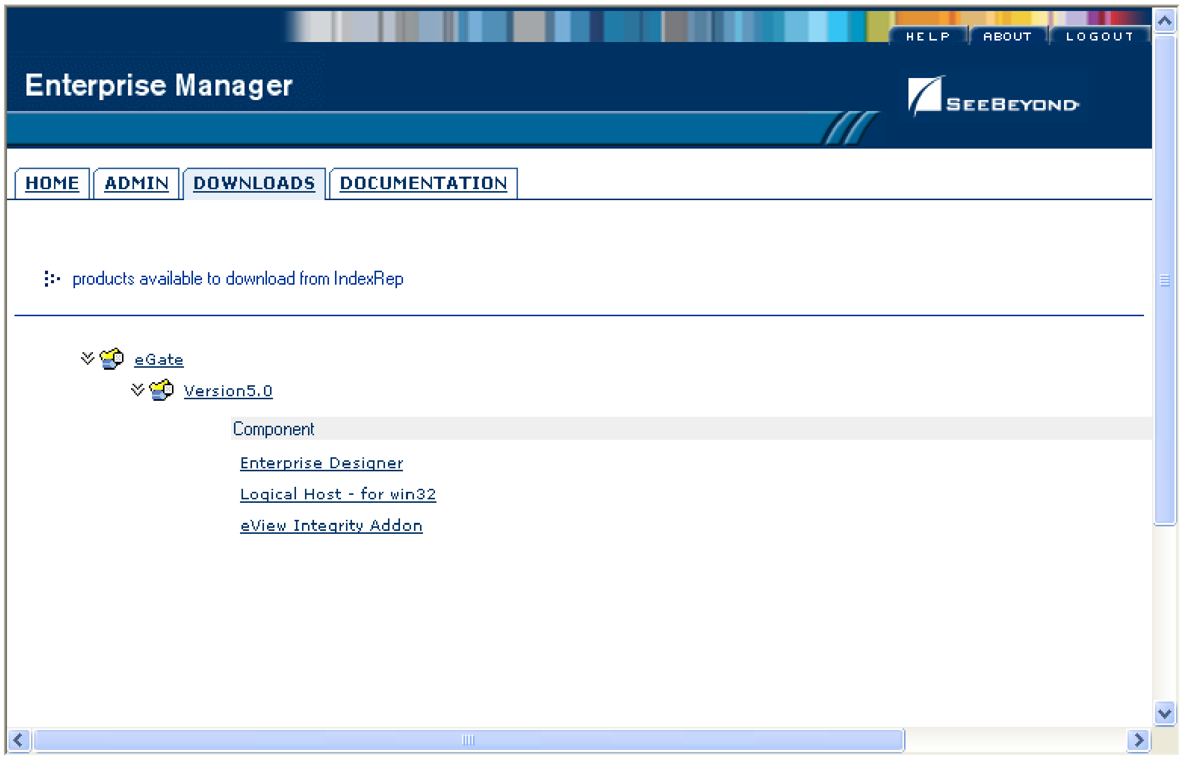
   A Navigate to the eView INTEGRITY Add-on CD-ROM, select **eViewIntegrityAddonDoc.sar**, and then click **Open**.

   B When you return to the **products available to upload to <*Repository_name*>** window, the **Products** box for eView 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 eView 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 eView Project.*

3.5.2. Installing eView in the Enterprise Designer

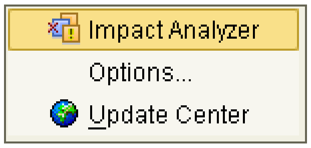
The final step in installing eView is updating the Enterprise Designer with eView. The Enterprise Designer must be installed on the machine on which you are installing eView. This step is performed using the Update Center, which is a tool in Enterprise Designer that allows you to install add-on modules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

Note: *The eView 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 eView in the Enterprise Designer

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

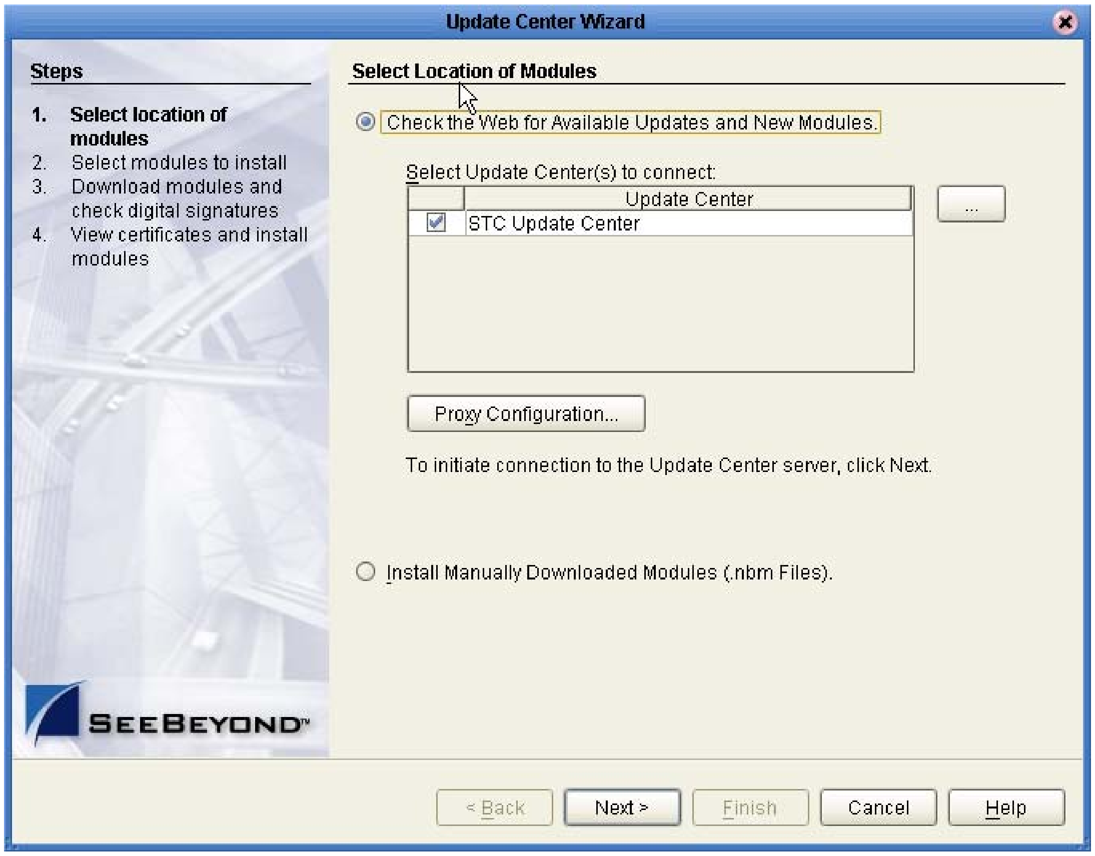
Figure 7 Tools Menu for Enterprise Designer



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

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

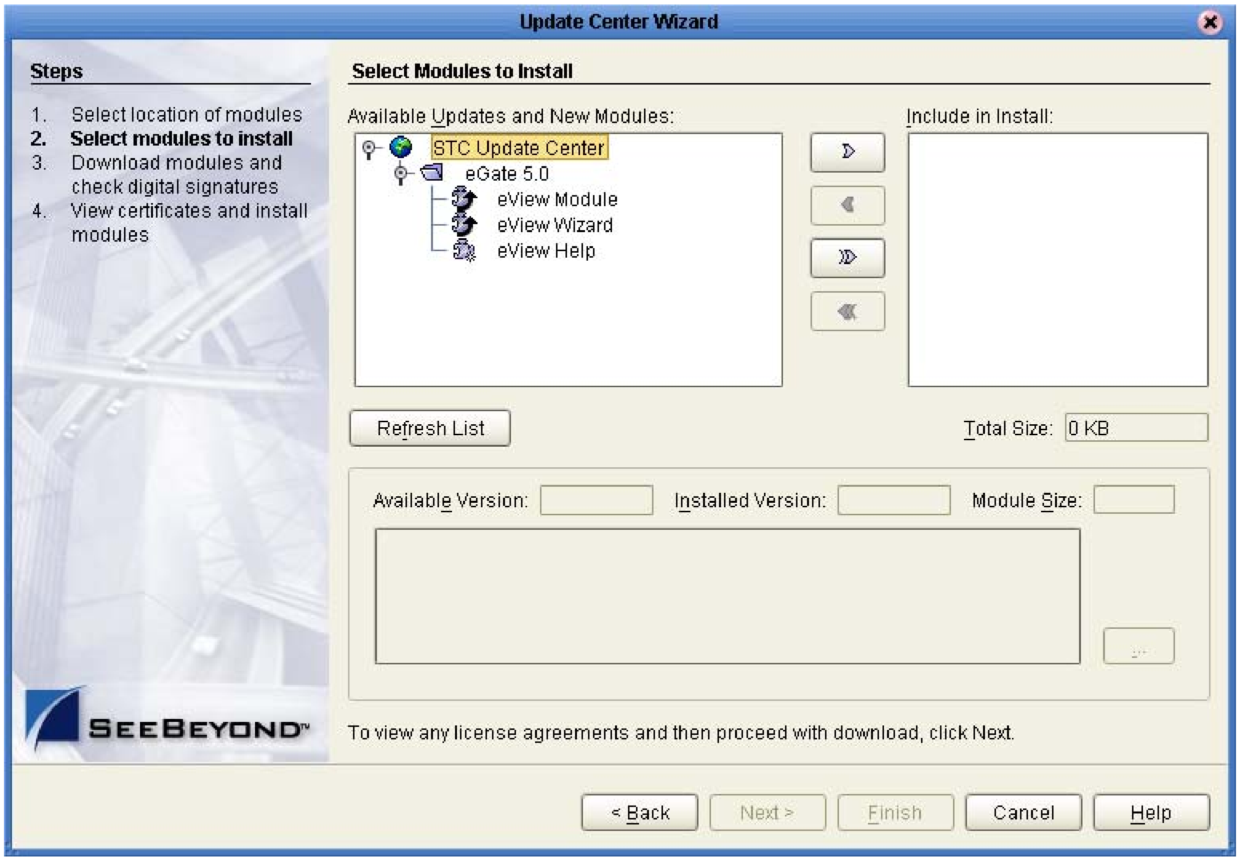
Figure 8 Update Center Wizard - Select Location of Modules


Update Center Wizard
Steps
1. Select location of modules
2. Select modules to install
3. Download modules and check digital signatures
4. View certificates and install modules
Select Location of Modules
Check the Web for Available Updates and New Modules.
Select Update Center(s) to connect:
Update Center
STC Update Center
...
Proxy Configuration...
To initiate connection to the Update Center server, click Next.
Install Manually Downloaded Modules (.nbm Files).
< Back
Next >
Finish
Cancel
Help

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

The **Select Modules to Install** page appears.(see **Figure 9 on page 44**).

Figure 9 Update Center Wizard - Select Modules to Install



To view any license agreements and then proceed with download, click Next.

4 Do one of the following:

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

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

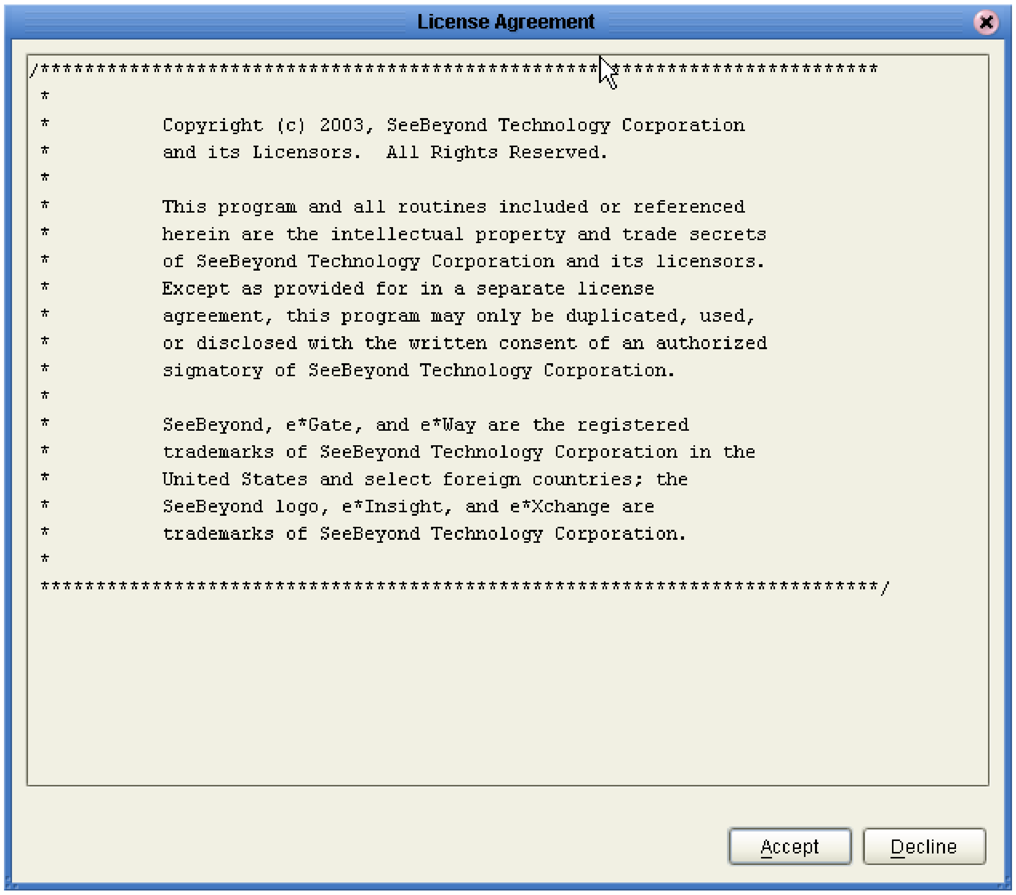
- To move all items in the **Available Updates and New Modules** box, click the **Add All** button (double-arrow button between the two panes).

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

5 Click **Next**.

The License Agreement window appears (see **Figure 10 on page 45**).

Figure 10 Update Center Wizard - License Agreement

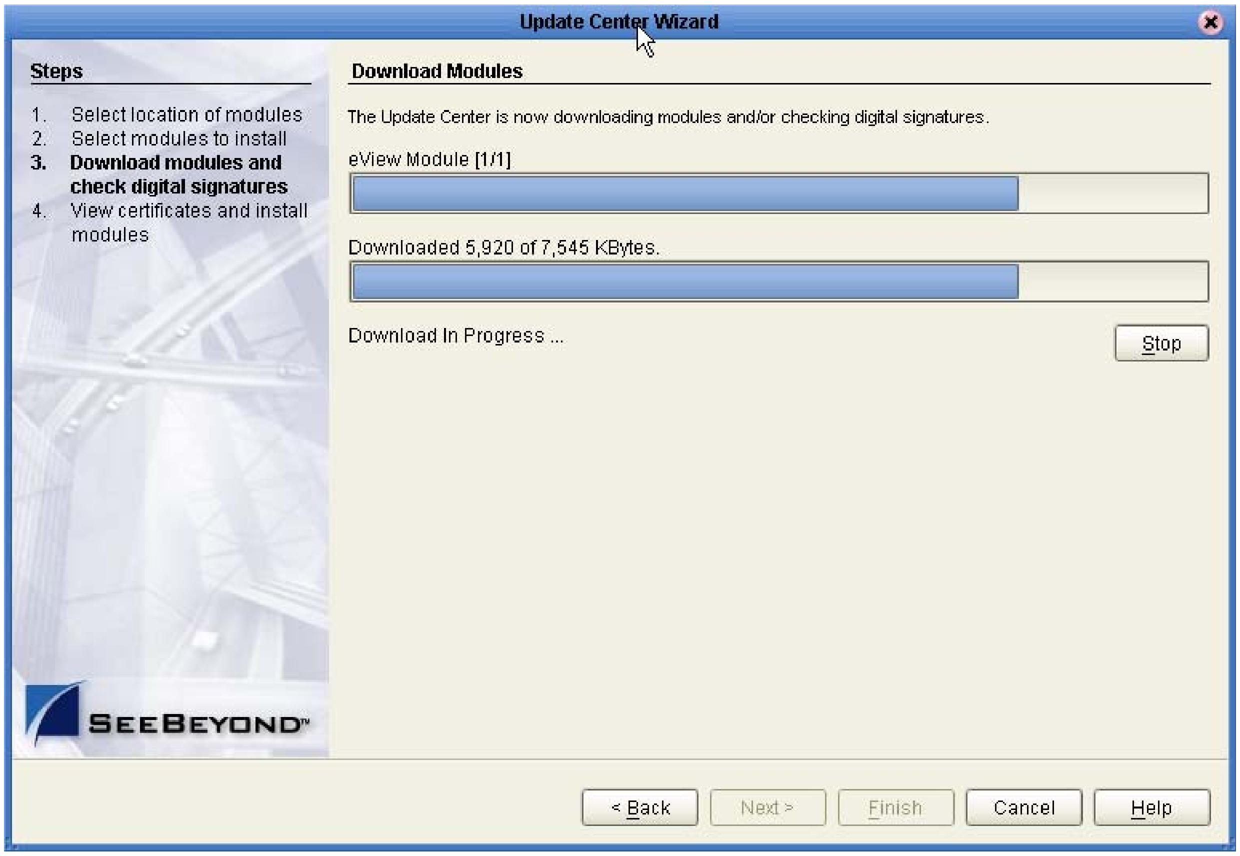


```
/****************************************************************************
 *
 *          Copyright (c) 2003, SeeBeyond Technology Corporation
 *          and its Licensors.  All Rights Reserved.
 *
 *          This program and all routines included or referenced
 *          herein are the intellectual property and trade secrets
 *          of SeeBeyond Technology Corporation and its licensors.
 *          Except as provided for in a separate license
 *          agreement, this program may only be duplicated, used,
 *          or disclosed with the written consent of an authorized
 *          signatory of SeeBeyond Technology Corporation.
 *
 *          SeeBeyond, e*Gate, and e*Way are the registered
 *          trademarks of SeeBeyond Technology Corporation in the
 *          United States and select foreign countries; the
 *          SeeBeyond logo, e*Insight, and e*Xchange are
 *          trademarks of SeeBeyond Technology Corporation.
 *
 ****************************************************************************/
```

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

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

Figure 11 Update Center Wizard - Download Modules


Update Center Wizard
Steps
1. Select location of modules
2. Select modules to install
3. Download modules and check digital signatures
4. View certificates and install modules
Download Modules
SEEBEYOND
Stop
< Back
Next >
Finish
Cancel
Help

The Update Center is now downloading modules and/or checking digital signatures.

eView Module [1/1]

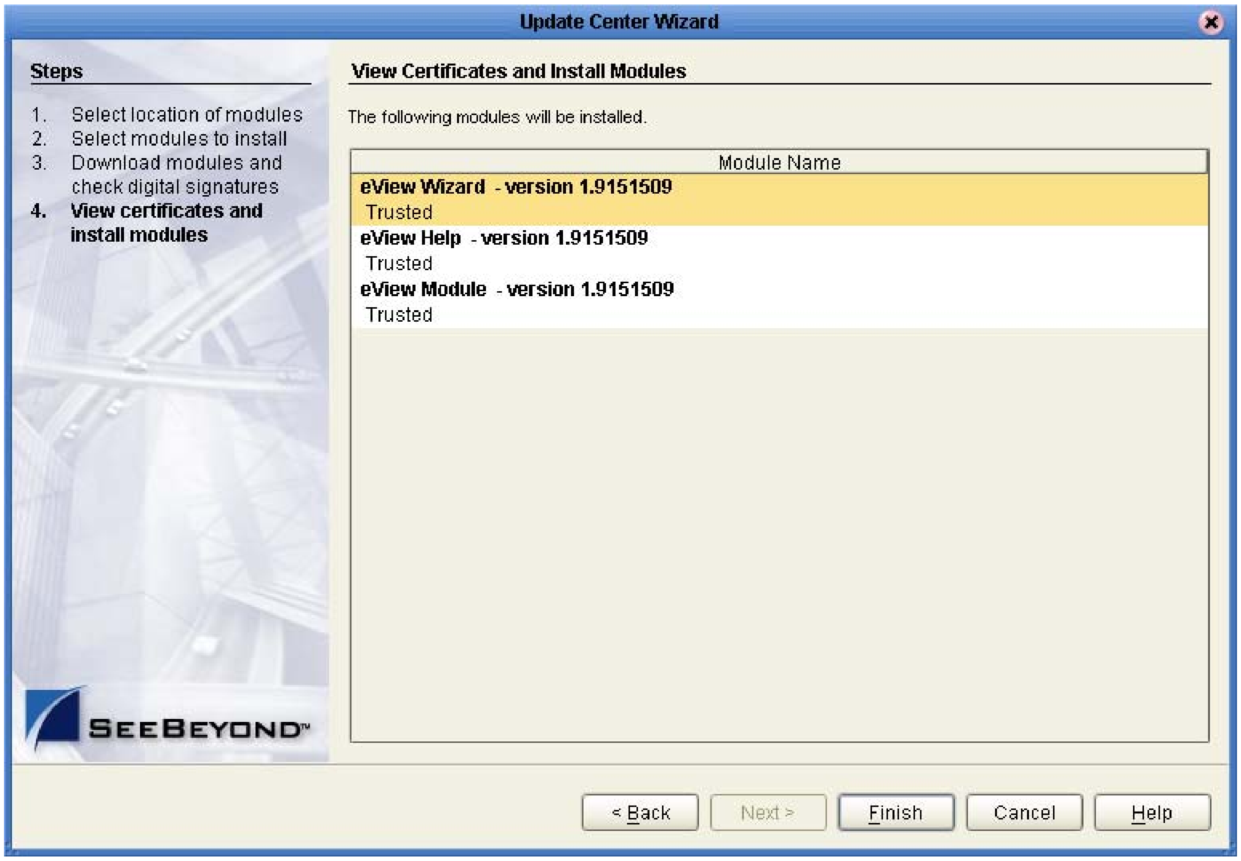
Downloaded 5,920 of 7,545 KBytes.

Download In Progress ...

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


Update Center Wizard
Steps
1. Select location of modules
2. Select modules to install
3. Download modules and check digital signatures
4. View certificates and install modules
SEEBEYOND™
< Back
Next >
Finish
Cancel
Help

View Certificates and Install Modules

The following modules will be installed.

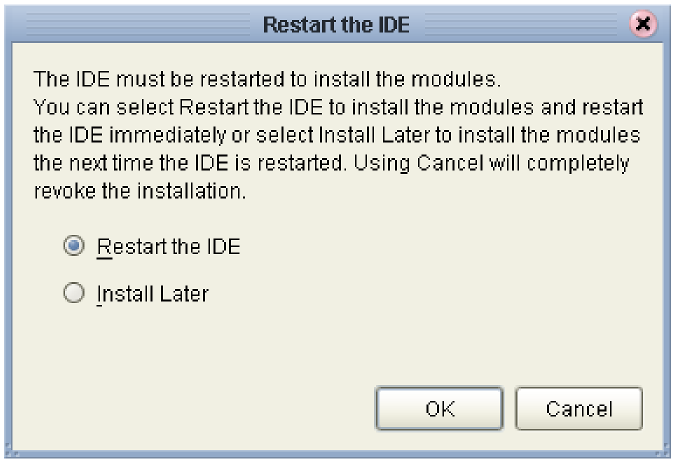
Module Name
eView Wizard - version 1.9151509 Trusted
eView Help - version 1.9151509 Trusted
eView Module - version 1.9151509 Trusted

8 Click **Finish**.

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

9 The modules that were installed must be reloaded for eView 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



The IDE must be restarted to install the modules.
You can select Restart the IDE to install the modules and restart the IDE immediately or select Install Later to install the modules the next time the IDE is restarted. Using Cancel will completely revoke the installation.

[✓ Restart the IDE]
Install Later

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

11 When the Enterprise Designer is restarted, the sample Project appears in the Project Explorer.

Chapter 4

Creating the Master Index Framework

The first step in creating a master index is to use the eView Wizard to define the object, such as a person or company object, that will be stored and cross-referenced in the master index database. This steps creates all of the configuration files required by the master index.

This chapter describes the eView Wizard, and provides instructions for using the wizard to create the basic configuration of the master index.

4.1 The eView Project

The eView Wizard runs in the context of an eGate Project. You must create a Project for the master index before you can use the wizard to create the configuration of the index. The eView Wizard creates the following elements of the eView Project.

- Object Definition
- Configuration Files
- Database Scripts (for processing codes and system information)

Generating the Project creates additional components of the master index based on the information specified in the Object Definition. From these basic pieces, you can create and configure the connectivity components of the Project, and then define the Environments and Deployment Profiles for the application.

4.2 The eView Wizard

The eView Wizard provides a user interface on which you can define the structure of your enterprise object, the deployment environment, and the source systems to be integrated in the eView system. You can also specify characteristics about the appearance of the Enterprise Data Manager (EDM).

When you complete the wizard, eView automatically generates several XML files that are used to define and create the master index and the runtime environment. You can customize these files, if needed, using the XML editor in the Enterprise Designer. **Chapter 5** provides an overview of these files. The *eView Studio Configuration Guide* provides detailed information and instructions for modifying each file. The files are stored in the Repository and can only be modified using the eView editors in the Enterprise Designer.

The wizard also generates database scripts that are used to insert start-up data into the database. The eView Wizard provides the option to generate all application files at once. In addition to the configuration and database files described above, this creates an outbound and a method OTD, complete database scripts, the Custom Plug-ins function, and all required application files.

4.2.1. Working with the eView Wizard

Accessing the eView Wizard

The eView Wizard can only be accessed from an existing Project in Enterprise Designer. Once you create a Project, you can right-click in the Project Explorer pane, and select **New -> eView Application**. This launches the eView Wizard. You should be familiar with the Enterprise Designer and eGate Projects before creating the master index.

eView Wizard Toolbar Buttons

The toolbar buttons described in Table 2 provides one-click shortcuts for executing commands in the eView Wizard. Place the cursor over a toolbar button to display the title of that button.

Note: *If a toolbar button is dimmed, you cannot use it with the selected component.*

Table 2 eView Wizard Toolbar Buttons

Button	Command	Function
📁	Add Primary Object	Adds a parent object, with no predefined child object or fields.
📁	Add Sub Object	Adds a child object, with no predefined fields, under the parent object.
◆	Add Field	Add a new field under the selected object.
✗	Delete	Deletes the selected object or field. If you delete an object, all fields in that object are also deleted.
▾	Templates	Opens a template menu, from which you can select a Company or a Person template with predefined fields and sub-objects.

eView Wizard Navigation Buttons

The navigation buttons described in Table 3 allow you to navigate through the windows of the eView Wizard.

Note: *If a navigation button is dimmed, you cannot use it on the displayed page.*

Table 3 eView Wizard Navigation Buttons

Button	Command	Function
< Back	Back	Returns to the previous step in the wizard. This button is disabled on the first step.
Next >	Next	Goes to the next step in the wizard. This button is disabled on the last step.
Finish	Finish	Saves configuration information, created Project files, and closes the wizard. This button is only enabled on the last step.
Cancel	Cancel	Closes the wizard without saving the configuration information.
Help	Help	Displays the online help documentation for the eView Wizard.

4.3 Before you Begin

Creating a master index requires in-depth analyses of your business requirements, legacy data, and data processing requirements. After the initial analysis, you can plan and design how you will configure the index using the eView Wizard and how you will customize that configuration after running the wizard. In addition, you must plan and design each physical component of the eView Project. For additional information about analyzing, planning, and designing eGate components, see the *eGate Integrator Deployment Guide*.

4.3.1. Data Analysis

Before you run the eView Wizard to create the master index, you perform an analysis against the data that will be stored in the index database. Analyzing your data requires extracting a set of records from each system that needs to share data with the eView master index. At a minimum, each extracted data record should include all fields used for matching. SeeBeyond or a qualified third party performs the preliminary data analysis. It is important that SeeBeyond receives the data extracts for analysis as early in the implementation process as possible.

The data analysis process helps you define the best object definition for your index, and to configure the EDM, matching and standardization rules, survivor calculator, and queries. It also helps identify over-used default field values, field-formatting inconsistencies, frequently unpopulated or incorrectly populated fields, and so on.

4.3.2. Project Planning

Before you create the eView Project in the Enterprise Designer, you must analyze the business requirements of the project and determine the Project components that will help you meet those requirements. Planning the Project includes defining how each external system will share information with the master index, and how the master index will share information with those external systems. In addition, the Collaboration of the Project contains the Java methods that define how the master index processes incoming data. Collaborations can also be used to transform the data sent from external systems into a format that can be read by the master index.

One additional consideration is whether to integrate the eView methods into an eInsight Business Process by creating Web pages through eVision to access the eView master index database.

4.3.3. Project Initiation Checklist

Before you begin using the eView Wizard to create your master index, make sure you have obtained the following information:

- The primary object to be indexed, such as a person, customer, business, and so on
- Any secondary objects, such as telephone numbers and addresses
- All fields to be stored in the index, for both the primary and secondary objects
- The name of each field as it appears on the EDM, and whether the field will be a standard text field or will be populated from a menu list (if a field will be populated from a menu list, you should also know the name of the list)
- Which fields are required
- Which fields must be unique to the primary object
- Which fields will be used for matching
- Any special formatting requirements, such as character types, the data type, minimum and maximum values, and field size
- Which fields will appear on EDM search and search results windows
- The processing codes for the source systems being integrated into the index
- The match and standardization engines to be used in the index

4.4 Creating the Master Index Configuration

The eView Wizard provides a simple method for you to create the Object Definition and the runtime configuration files for your master index. This section provides instructions for creating a new eGate Project, and for using the eView Wizard to create the definition and configuration files for the master index. To create the initial master index configuration, follow these steps:

- **Step 1: Create a Project** on page 53

- **Step 2: Launch the eView Wizard** on page 54
- **Step 3: Name the eView Application** on page 54
- **Step 4: Define Source Systems** on page 55
- **Step 5: Define the Deployment Environment** on page 57
- **Step 6: Define Parent and Child Objects** on page 58
- **Step 7: Define the Fields for each Object** on page 63
- **Step 8: Generate the Project Files** on page 69
- **Step 9: Review the Configuration Files** on page 70

4.4.1. Step 1: Create a Project

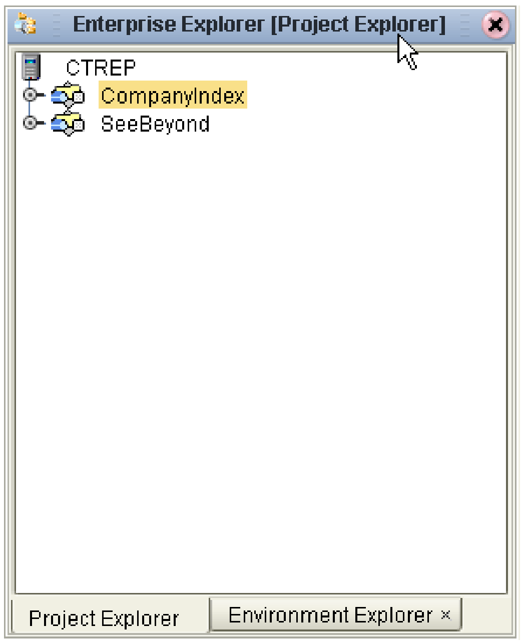
Before you can access the eView Wizard, you must create and name a new Project in the Enterprise Designer.

To create a Project

1. In the Project Explorer pane of the Enterprise Designer GUI, select the Repository name.
2. Right-click to display the **Repository** context menu, and then select **Project**.

   A new Project node appears under the Repository.
3. Enter a unique name for the Project and then press **Enter**.

Figure 14 Enterprise Explorer



4.4.2. Step 2: Launch the eView Wizard

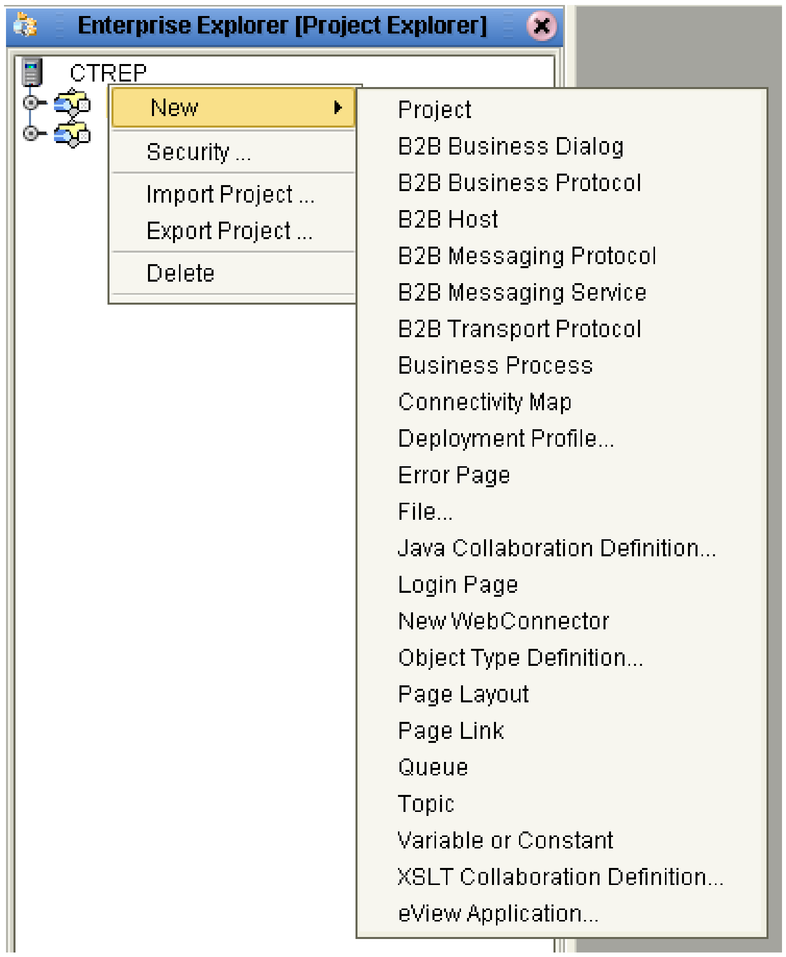
Once you create a Project for the master index, you can launch the eView Wizard and begin defining the eView instance.

To launch the eView Wizard

1 Complete **"Step 1: Create a Project"**.

2 Select the new Project, and then right-click in the **Project Explorer** pane of the Enterprise Designer GUI to display the **Project** context menu (Figure 15).

Figure 15 Project Context Menu



3 From the context menu, select **eView Application**.

4 Continue to **"Step 3: Name the eView Application"**.

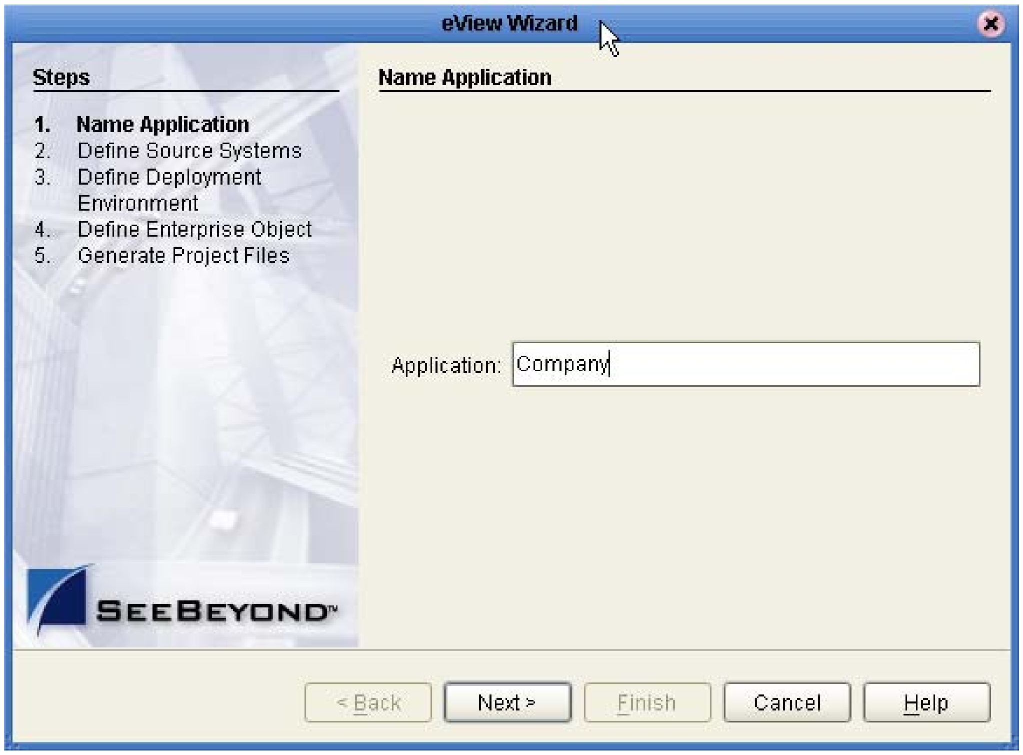
4.4.3. Step 3: Name the eView Application

Each master index you create is an eView application. Before you can configure the new master index, you must name the master index application. The name you specify will become the name of the parent object, so it must follow Oracle naming requirements for database tables.

To name the eView application

1 Complete **"Step 2: Launch the eView Wizard"**.

Figure 16 eView Wizard - Name Application



Name Application

Application: Company

2 In the **Application** field of the **Name Application** window, enter a name for the new eView application, and then click **Next**.

Note: *Make sure the application name you specify is unique in the Project (if you have more than one eView application in the Project).*

The **Define Source Systems** window appears as shown in Figure 17.

3 Continue to **"Step 4: Define Source Systems"**.

4.4.4. Step 4: Define Source Systems

After you specify a name for the new master index application, you need to specify the names of the source systems that will be integrated into the new master index system.

To define source systems

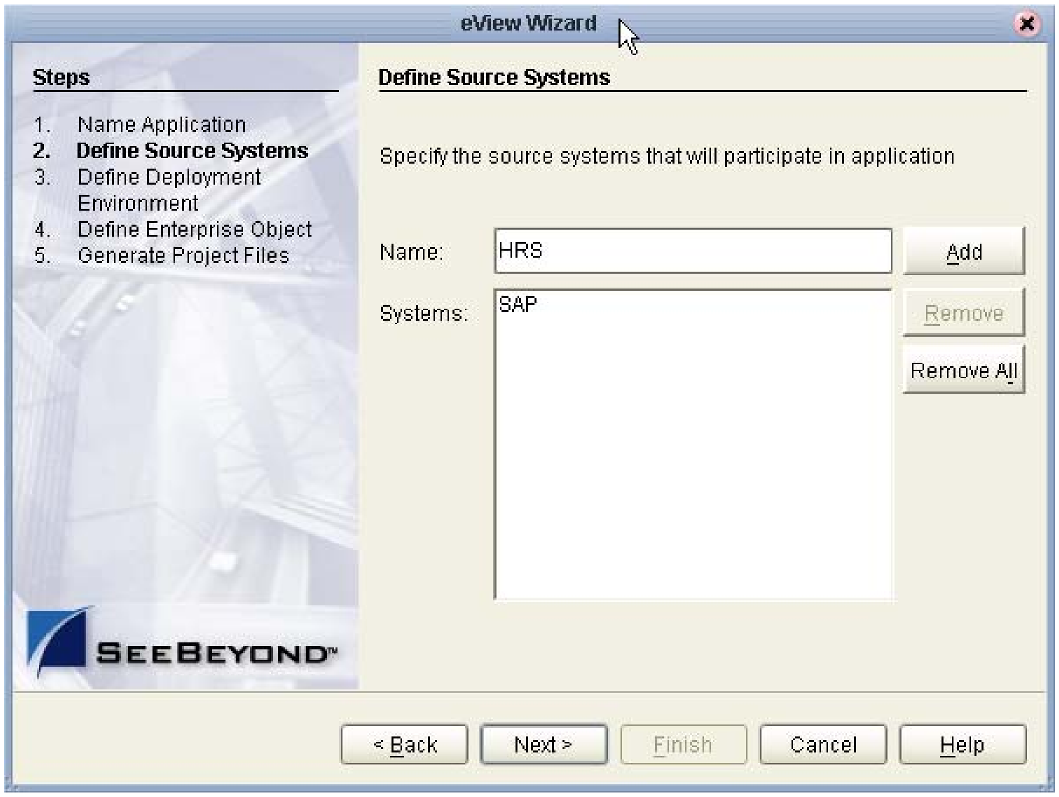
1 Complete **"Step 3: Name the eView Application"**.

2 In the **Name** field of the **Define Source Systems** window, enter the processing code of one of the source systems that will share data in the eView system, and then click **Add**.

The value you entered appears in the **Systems** box.

Important: *Be sure to enter the processing code of the system and not the name. This value is entered in the Best Record file for defining survivor strategies for the SBR.*

Figure 17 eView Wizard - Define Source Systems



Specify the source systems that will participate in application

Name: HRS

Systems: SAP

3 Do any of the following:

- To define additional systems, repeat the above step for each source system that will share information with the master index.
- To remove a system from the list, highlight the name of that system in the **Systems** box, and then click **Remove**.
- To remove all systems from the list, click **Remove All**.

4 When you have defined all required source systems, click **Next**.

The **Define Deployment Environment** window appears.

5 Continue to **"Step 5: Define the Deployment Environment"**.

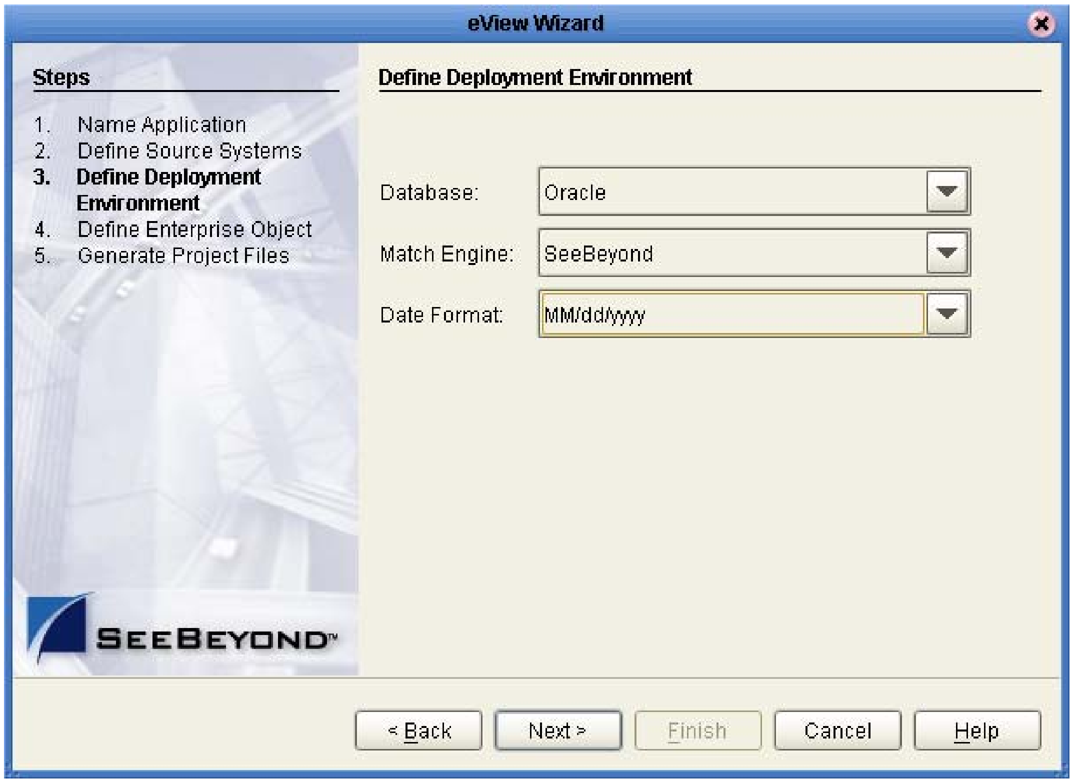
4.4.5. Step 5: Define the Deployment Environment

Once you define systems, you must specify information about the deployment environment, including the database and match engine vendors.

To define the deployment environment

1 Complete **"Step 4: Define Source Systems"**.

Figure 18 eView Wizard - Defining the Deployment Environment



Define Deployment Environment

Database: Oracle
Match Engine: SeeBeyond
Date Format: MM/dd/yyyy

2 On the **Define Deployment Environment** window, select the appropriate values for the fields described in Table 4.

3 When you have defined the deployment environment fields, click **Next**.

4 Continue to **"Step 6: Define Parent and Child Objects"**.

Table 4 Deployment Environment Fields

Field	Description
Database	The type of database being used for the master index. Currently, only Oracle is supported.
Match Engine	The type of match and standardization engine to use for the implementation. You can select SeeBeyond or INTEGRITY.

Table 4 Deployment Environment Fields

Field	Description
Date Format	The date format for the master index system. This defines how dates should be entered and how they appear on the EDM. You can select **MM/dd/yyyy**, **dd/MM/yyyy**, or **yyyy/MM/dd**.

4.4.6. Step 6: Define Parent and Child Objects

After you define the deployment environment for the master index, you can begin to define the structure of the object you want to index. The primary object will be the parent object for any other objects defined.

You can create new undefined objects, create objects using predefined templates, or use a combination of both methods to create the objects in your enterprise object. Perform any of the following actions to define the objects in the enterprise object.

- **Creating Undefined Objects** on page 58
- **Creating Objects from a Template** on page 60
- **Deleting an Object from the Structure** on page 62

Complete **"Step 2: Launch the eView Wizard"** through **"Step 5: Define the Deployment Environment"** before performing these procedures.

Important: *Make sure the name of the parent object is the same as the application name you specified earlier (this is the default name).*

Creating Undefined Objects

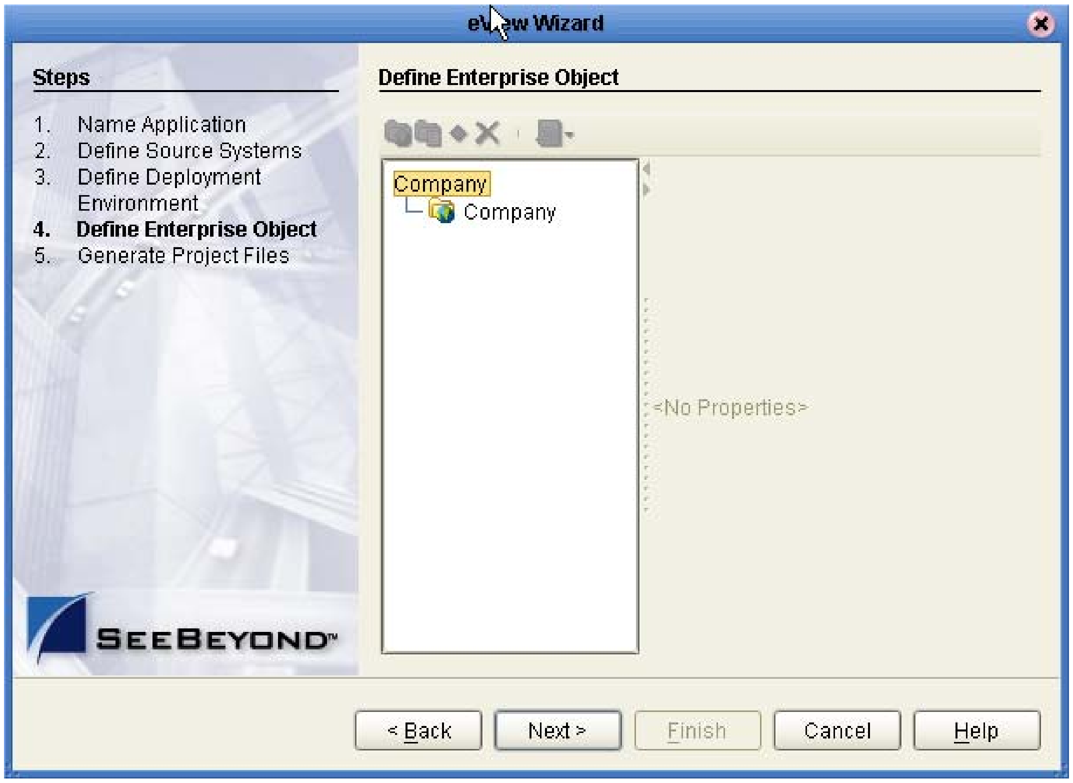
When you create undefined objects, you create an empty object with no predefined fields or child objects.

To create undefined parent and child objects

1 On the **Define Enterprise Object** window, click the **Add Primary Object** icon (you can also right-click in the tree pane and select **New Primary Object** from the context menu).

The initial node appears on the tree, as shown in Figure 19. By default, the name of the field is the same as the name of the application you defined in **"Step 2: Launch the eView Wizard"**.

Figure 19 eView Wizard - New Undefined Parent Object

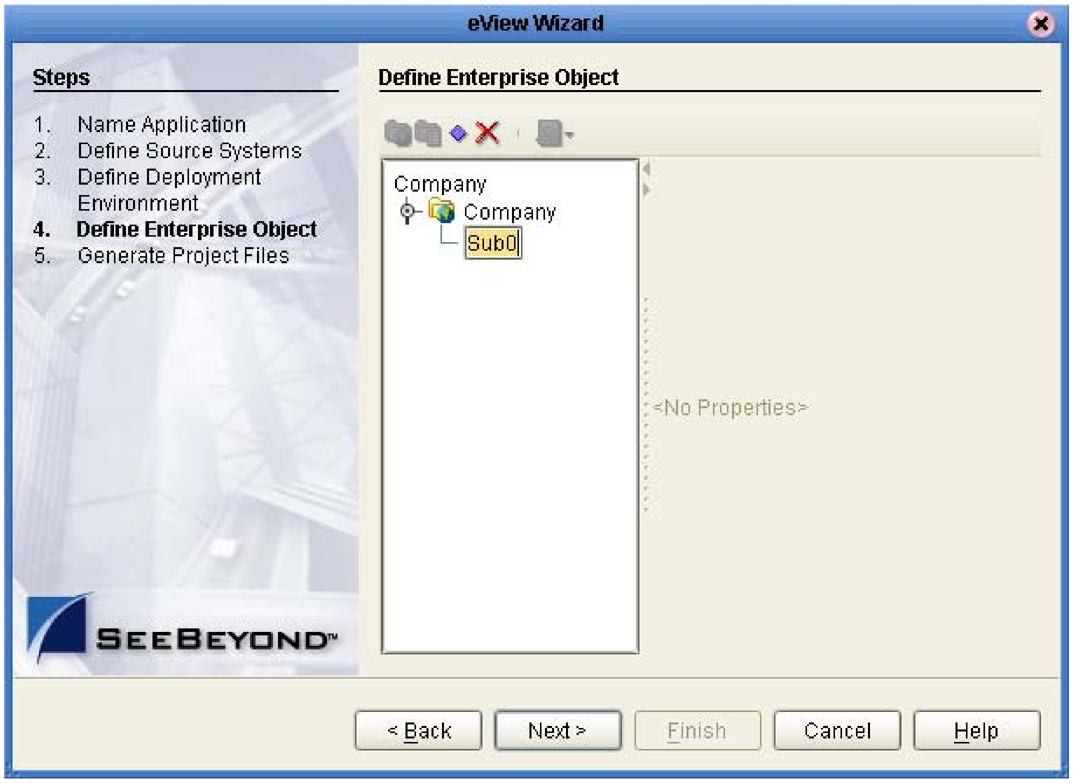


2 Accept the default name by pressing **Enter**.

3 To create a new child object, select the primary object created above, and then click the **Add Sub Object** icon (or right-click to display the context menu and then select **New Sub Object**).

The new child node appears on the tree, as shown in Figure 20.

Figure 20 eView Wizard - Creating an Undefined Child Object



4 To accept the default name, press **Enter**. To change the name, type the new name and then press **Enter**.

5 Repeat steps 3 and 4 for each child object.

6 Continue to **"Step 7: Define the Fields for each Object"**.

Creating Objects from a Template

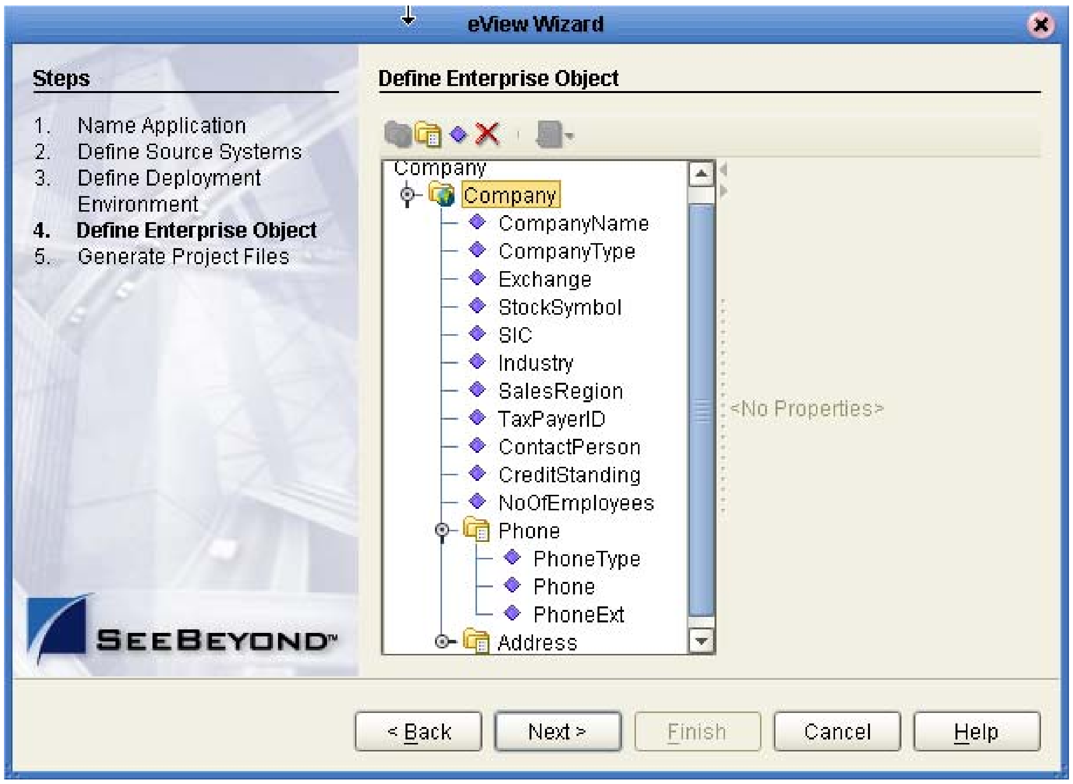
When you create objects from a template, secondary objects and fields are predefined. You can modify the predefined attributes if necessary.

To create parent and child objects from a template

1 On the **Define Enterprise Object** window, click the Templates icon and select the template you want to use (or right-click in the tree-view pane to display the **New Primary Object** context menu, point to Templates, and then select the template name).

The objects and fields from the template appear in the tree-view pane in the center of the window as shown in Figure 21.

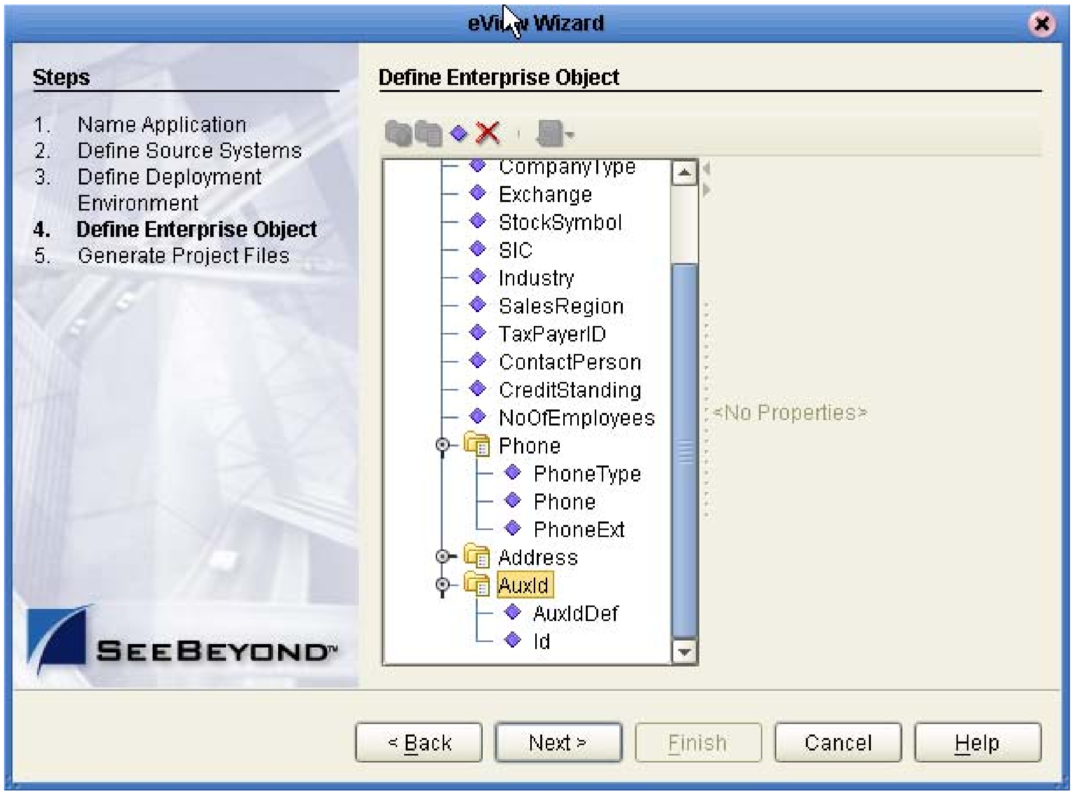
Figure 21 eView Wizard - Company Template



2 To create a child object from a template, right-click the primary object created above to display the context menu, point to **Template**, and then select the name of the template you want to use.

The new object and any defined fields appear in the object tree, as shown in Figure 22.

Figure 22 eView Wizard - Creating a Child Object from a Template



3 If necessary, change the name of the new object by doing the following:

- Click twice on the name.
- Type the new name.
- Press **Enter**.

4 Repeat steps 2 and 3 for each child object template you want to create.

5 Continue to **"Step 7: Define the Fields for each Object"**.

Deleting an Object from the Structure

If you add an object in error, or do not want to use one of the objects in a predefined template, you can delete the object from the structure.

To delete an object from the structure

1 On the **Define Enterprise Object** window, select the object you want to remove.

2 Do any of the following:

- Right-click in the object tree pane to display the context menu, and then select **Delete**.
- Press the **Delete** key.
- Click the **Delete** icon in the eView Wizard toolbar.

The object and any fields associated with that object are deleted. If you remove the parent object, all child objects are deleted.

4.4.7. Step 7: Define the Fields for each Object

After you define all the parent and child objects for your enterprise object, you must define the fields for each object. Every field has a set of properties that must be configured before creating the master index configuration files. If you chose a predefined template to create your objects, be sure to check the properties for all predefined fields to be sure they are configured correctly for your implementation.

After you have defined the parent and child object, you can perform any of the following actions to define the fields for those objects.

- **Creating a Field** on page 63
- **Configuring Field Properties** on page 64
- **Deleting a Field** on page 69

Creating a Field

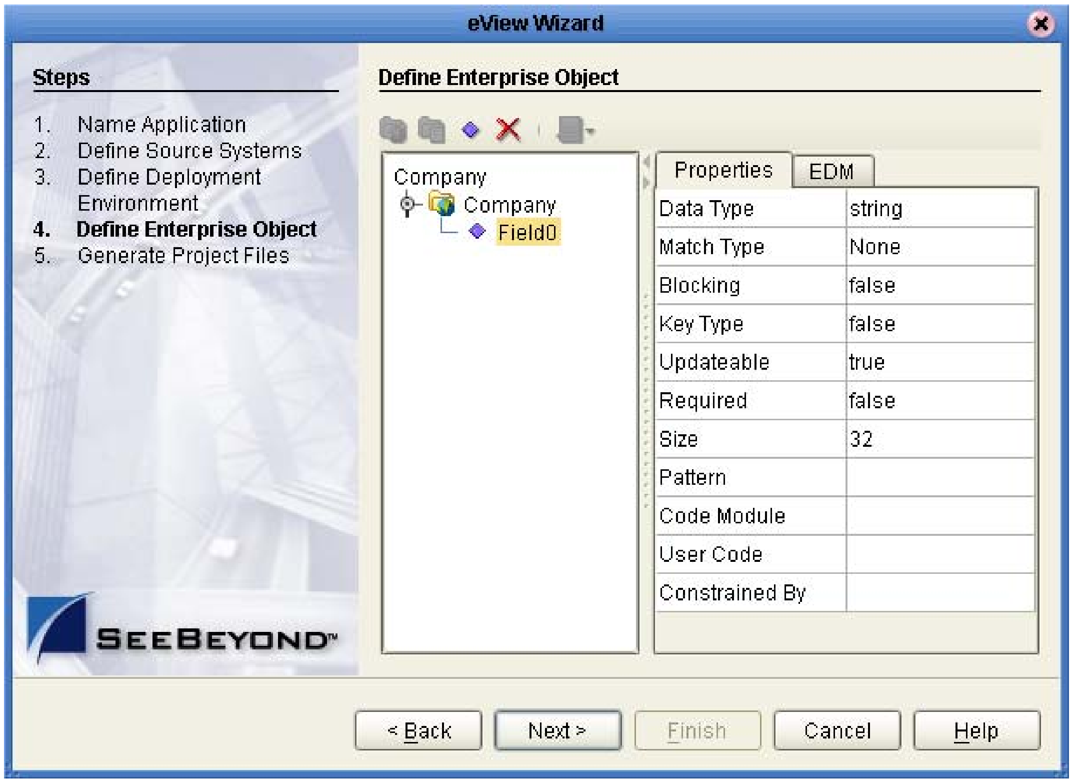
If you created an empty object in **"Step 6: Define Parent and Child Objects"**, you must create each field that belongs to the object. If you created objects using a predefined template, you can add new fields to the object if needed.

To create a field

1. Complete **"Step 6: Define Parent and Child Objects"**.
2. In the object tree pane of the **Define Enterprise Object** window, do one of the following:
   - To add the field to the end of the object's field list, select the name of the object to which you want to add a new field and then click the **Add Field** icon (or right-click in the object tree pane to display the context menu and then click **New Field**).
   - To add the field immediately following an existing field, select the field after which you want to add the new field and then click the **Add Field** icon (or right-click in the object tree pane to display the context menu and then click **New Field**).

   The tree expands and a new field is inserted.

Figure 23 eView Wizard - New Field Properties



Data Type	string
Match Type	None
Blocking	false
Key Type	false
Updateable	true
Required	false
Size	32
Pattern	
Code Module	
User Code	
Constrained By	

3 To accept the default name, press **Enter**. To change the name, type the new name and then press **Enter**.

Important: *If you enter a field name longer than 20 characters, a warning dialog appears. While Oracle can handle names up to 30 characters, eView appends text to the end of fields defined for phonetic encoding or standardization. For fields that will be parsed, normalized, or phonetically encoded, make sure the name of the original field does not exceed 20 characters. Any other field can have a name up to 30 characters long. For information about the names of the fields automatically created by the eView Wizard, see* **Appendix B**.

4 Continue to **"Configuring Field Properties"**.

Configuring Field Properties

When a field is created, a set of default properties are defined for that field. You can modify the property configuration for each field to suit your data processing, storage, and display requirements.

To configure field properties

1 Complete the steps under **"Creating a Field"**.

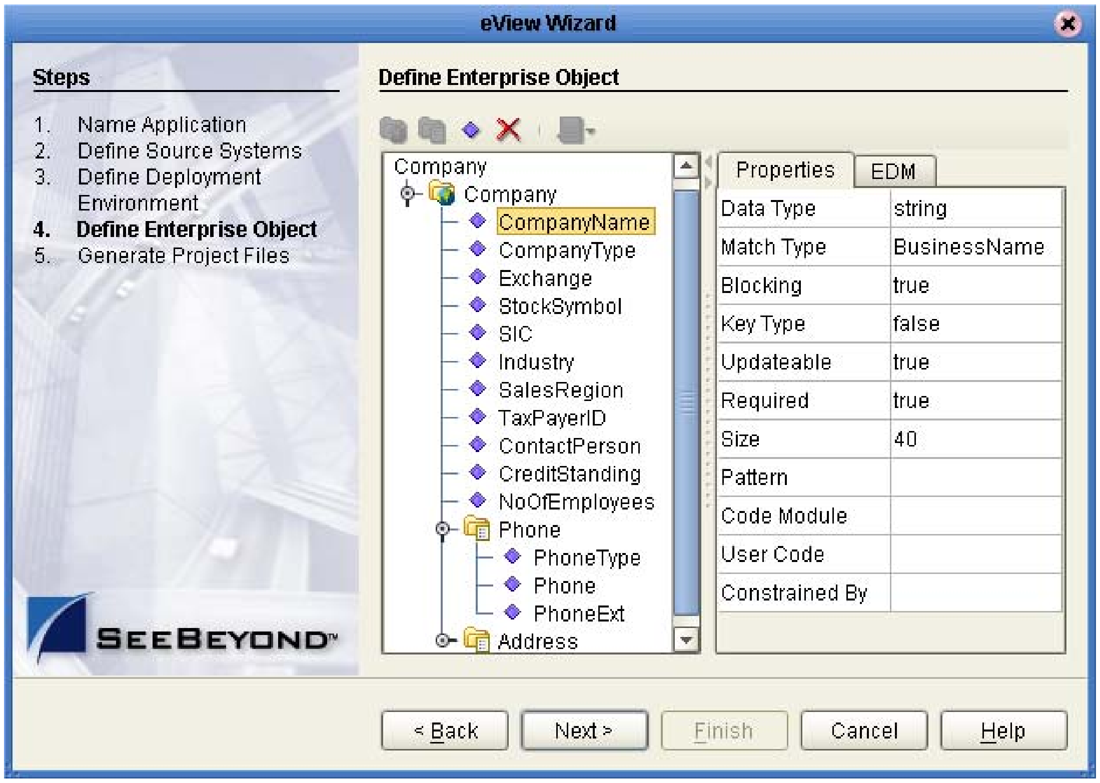
2 In the object tree pane of the **Define Enterprise Object** window, select the field you want to configure.

3 On the **Properties** page in the right side of the window, modify the value of any of the listed properties (shown in Figure 24).

To see a list of descriptions and restrictions for each property, see **Table 5 on page 66**.

Note: *After you modify a property value, press **Enter** to apply the change.*

Figure 24 Field Properties Page

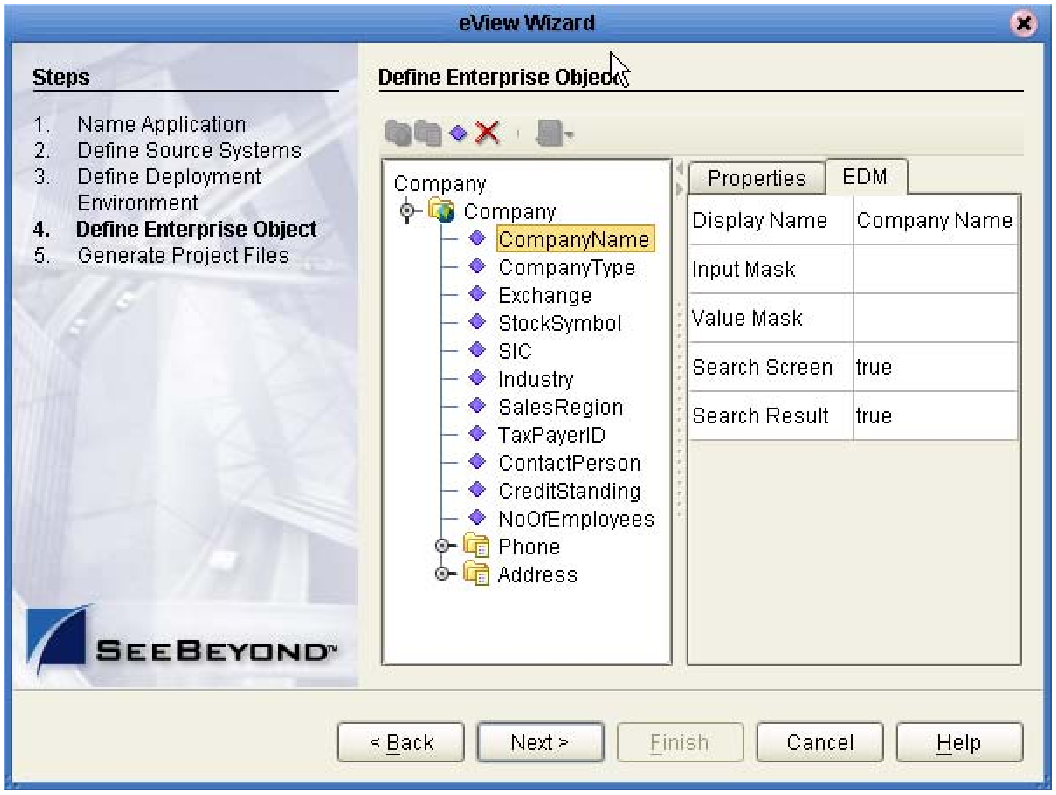


Data Type	string
Match Type	BusinessName
Blocking	true
Key Type	false
Updateable	true
Required	true
Size	40
Pattern	
Code Module	
User Code	
Constrained By	

4 On the right side of the window, click the **EDM** tab, and then modify the value of any of the listed properties on the **EDM** page.

To see a list of descriptions and restrictions for each property, see **Table 6 on page 68**.

Figure 25 Field EDM Page



Display Name	Company Name
Input Mask	
Value Mask	
Search Screen	true
Search Result	true

5 When you have created and configured all of the necessary fields for each object, click **Next**.

6 Continue to **"Step 8: Generate the Project Files"**.

Table 5 Field Properties

Property	Description
Data Type	The eView data type of the field, such as string, object, date, and so on. The possible values are string, date, long int, int, short, or boolean.
Match Type	The type of matching to be performed against the field. The available options for this field are dependent on the match engine you selected. ***Note:*** *The match types you specify here define the standardization, normalization, and phonetic encoding structure of the Match Field file. They also define the match string. The match types in the Match Field file might differ from the eView Wizard match types. For complete information about match types, see* **Appendix B**.
Blocking	An indicator of whether the field will be used in the blocking query. Specify **true** to add the field to the blocking query; specify **false** to omit it from the blocking query.

Table 5 Field Properties

Property	Description
Key Type	An indicator of whether the field is used to identify unique objects. For example, in a business index you might store several addresses for each business. Each address is assigned an address type, which can be used to identify an address for a business. Specify **true** if the field is a unique record identifier, or **false** if it is not. Key type fields should be required, unless a combination of fields are specified as key types for an object (see below).
Updateable	An indicator of whether the field can be updated from the EDM and external system messages. Specify **true** if the field can be updated, or **false** if it cannot.
Required	An indicator of whether the field is required in order to save an enterprise object to the database. Specify **true** if the field is required, or **false** if it is not. If only one key type field is defined for an object, it should also be required.
Size	The number of characters allowed in each field. This determines the number of characters allowed in the database columns, and defines the maximum number of characters that can be entered into each field on the EDM.
Pattern	The required data pattern for the field. 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. You might want to define patterns for date, telephone, or SSN fields.
Code Module	The identification code for the drop-down list that will appear for this field in the EDM. ***Note:*** *This must match an entry in the* ***code*** *column of the sbyn_common_header database table. You can define code lists in the* ***Code List*** *file after completing the wizard.*
User Code	The processing code for the drop-down list that appears for the fields defined by the **Constrained By** property. For more information, see the description of the **Constrained By** property below. ***Note:*** *This must match an entry in the* ***code_list*** *column of the sbyn_user_code database table.*

Table 5 Field Properties

Property	Description
Constrained By	The name of the field containing the corresponding **User Code** value (described above). **User Code** and **Constrained By** are used in conjunction 1) to define a drop-down list for the field that has the **Constrained By** value, and 2) to validate that field against definitions for the field with the **User Code** value. For example, if you want to verify city and postal code combinations, the PostalCode field would have a **User Code** value similar to "city_zip", and the City field would have a **Constrained By** value of "PostalCode". Based on the definitions for "city_zip" in the sbyn_user_code table, when you enter a zip code you could then select a city name from the values defined for that zip code. This could also be used to validate non-unique identification codes, such as account numbers or insurance policies.

Table 6 EDM Properties

Property	Description
Display Name	The name of the field as it will appear on the Enterprise Data Manager (EDM).
Input Mask	A mask used by the GUI to add punctuation to a field. For example, if users enter the date in the format MMDDYYYY, you can add an input mask to display the dates as MM/DD/YYYY. To define an input mask, type a character type for each character in the field, and place any necessary punctuation between the character types. For example, the input mask for the above date format is DD/DD/DDDD. The following character types are allowed: ▪ **D**—indicates a numeric character. ▪ **L**—indicates an alphabetic character. ▪ **A**—indicates an alphanumeric character.
Value Mask	A mask used by the index 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 as is entered for the input mask, but type an "x" in place of each punctuation mark. For example, if an SSN field has an input mask of DDD-DD-DDDD, you need to specify a value mask of DDDxDDxDDDD in order to strip the dashes before storing the SSN. A value mask is not required for date fields.

Table 6 EDM Properties

Property	Description
Search Screen	An indicator of whether the field will appear on the search windows of the EDM. Specify **true** to display the field, or **false** to hide it.
Search Result	An indicator of whether the field will appear on the search results windows of the EDM. Specify **true** to display the field, or **false** to hide it.

Deleting a Field

If you add a field in error, or do not need one of the predefined fields from a template, you can delete the field.

To delete a field

1. In the object tree pane of the **Define Enterprise Object** window, select the field you want to delete.
2. Right-click in the object tree pane to display the context menu.
3. From the context menu, select **Delete**.

   The field is removed from the object tree.

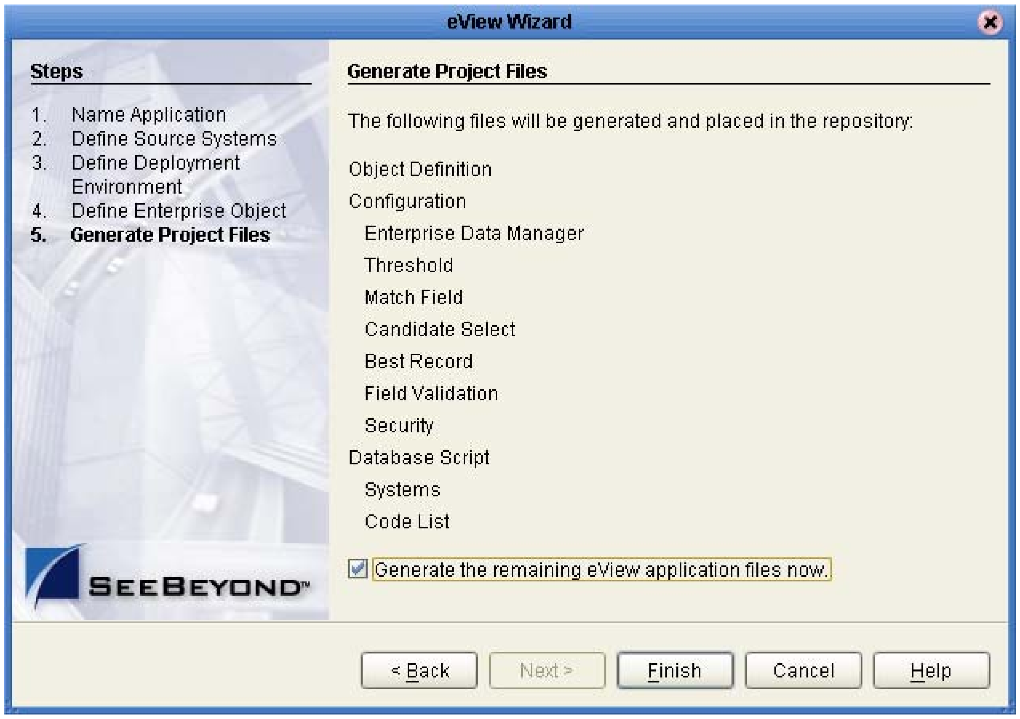
4.4.8. Step 8: Generate the Project Files

Once you have named the application and configured the source systems, deployment environment, objects, and fields for the master index, you must generate the configuration files and database scripts for the index. You have the option to create all remaining Project files at this time, or to wait until you have modified the configuration of the eView application. Either way, you should review the configuration files to be sure the application is set up correctly for your data processing environment.

To generate the configuration files

1. Complete **"Step 7: Define the Fields for each Object"**.

Figure 26 eView Wizard - Generating the Configuration Files



Generate Project Files

The following files will be generated and placed in the repository:

Object Definition
Configuration
  Enterprise Data Manager
  Threshold
  Match Field
  Candidate Select
  Best Record
  Field Validation
  Security
Database Script
  Systems
  Code List

[✓] Generate the remaining eView application files now.

2 Verify that all of the information you have entered is complete and correct.

3 To generate all application files for the Project, select the check box at the bottom of the window. To generate them later, after reviewing the configuration files, leave this check box unchecked.

4 On the Generate Configuration Files window, click **Finish**.

  The configuration files are generated, and are stored in the eGate Repository.

5 Continue to **"Step 9: Review the Configuration Files"**.

4.4.9. Step 9: Review the Configuration Files

After the eView Wizard is complete, several nodes representing eView configuration files are placed in the Project for the master index. Verify that the following nodes exist in the Project tree, and that they are configured correctly for your implementation. If you need to modify the configuration files, **Chapter 5** provides an overview of these files. See the *eView Studio Configuration Guide* for information about modifying the files.

Two folders, the **Match Engine** and **Standardization Engine** components, are only included if you specified the SeeBeyond Match Engine; if the INTEGRITY match engine was specified, these nodes do not appear.

Chapter 5

Configuring the Master Index

The eView Wizard creates several nodes in the eView Project that represent configuration files for the master index. Before generating the eView Project, make sure that each of these files is configured as required 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 *eView Studio Configuration Guide*.

5.1 Configurable Options

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

Object Definition

By customizing the Object Definition file, you can configure the structure of the data stored in the master index. This file contains the configuration of each object in the master index and their relationships to one another. It also defines the fields contained in each object, as well as certain 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 generated for the eView 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 the master index are all defined in the Candidate Select file. You can customize this file by configuring the types of queries used by the master index, including those that are performed from the EDM and those that are used during the match process. You can define both phonetic and alphanumeric searches for the EDM. By default, these are called basic searches. You can also define blocking queries, which define 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 the master index 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.

5.2 About the eView Configuration Files

The files that configure the components of the master index are created by the eView Wizard and define certain characteristics, such as how data is processed, queried, and matched. These files configure runtime components of the master index.

The configuration files include the following:

- **Object Definition** on page 73
- **Candidate Select** on page 73
- **Match Field** on page 74
- **Threshold** on page 74
- **Best Record** on page 74
- **Field Validation** on page 74
- **Enterprise Data Manager** on page 75

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

5.2.1. Object Definition

In the eView Wizard, you define the objects and fields contained in the object structure, along with attributes for those fields. The information you specify is written to the Object Definition file in the eView Project. This file defines each object in the master index and their relationships to one another. It also defines the fields contained in each object, as well as certain 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.

5.2.2. Candidate Select

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

5.2.3. Match Field

This file configures the Matching Service by defining standardization and matching fields for the master index. 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.

5.2.4. Threshold

This file configures the eView 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 the master index. 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.

5.2.5. 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.

5.2.6. 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 the master index database, and then specify those classes in this file.

5.2.7. Security

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

5.2.8. 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.

5.3 Modifying the eView 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 eView 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 eView Studio* or *Implementing Ascential INTEGRITY with eView Studio* 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 master index application.

5.4 Match Engine Configuration Files

If you chose to implement the SeeBeyond Match Engine in the eView Wizard, 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. eView 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 eView Studio*.

Chapter 6

Generating the Project

Once all of the configuration files are created, and are customized completely, you must generate the application to create the customized components. This chapter gives instructions for generating the Project, and describes the Project components that are created when you generate.

6.1 Generated Application Components

Generating an eView application is the process that actually creates the indexing application. Several custom components are created in the Project, along with the executable files for the application. eView uses the Object Definition to generate these components based on the configuration you defined. These components include:

- **Database scripts**—The scripts for creating and dropping tables and indexes are created based on the Object Definition file.
- **Custom Plug-ins module**—This component allows you to define additional customized processing rules for the master index 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 the master index. 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 the master index.

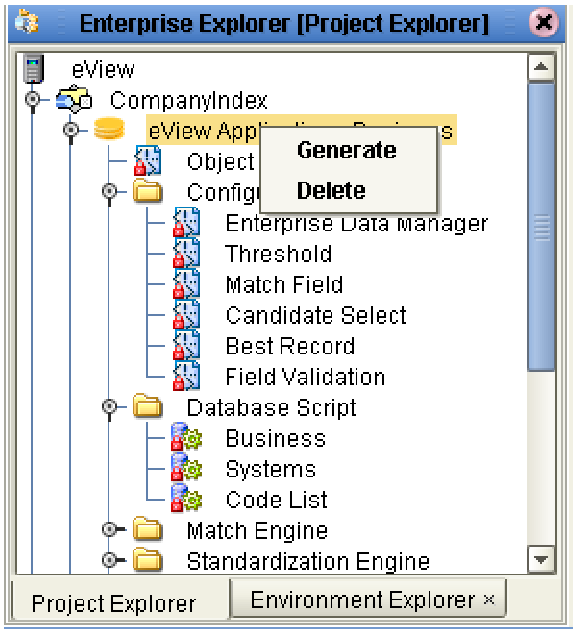
6.2 Generating the Project

Once all modifications to the configuration files are complete, you can generate the eView application to create 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 eView application in the **Project Explorer** pane to display the **Generate** context menu (shown in Figure 27).

Figure 27 Generate Context Menu



3. Select **Generate**. eView creates the new 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 7

Creating Custom Plug-ins

eView provides the ability to create custom update procedures to be performed on a record once standard eView 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 eView to use the custom procedures.

7.1 About Custom Plug-ins

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

7.1.1. Update Policies

For the primary transactions performed by the master index you can define additional custom processing to perform against the record that results from 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.

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.

7.1.2. 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 *eView Studio Configuration Guide*.

7.1.3. Custom eView Components

eView provides a flexible framework, allowing you to create custom Java classes to plug in to most eView 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 the master index. 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 the master index. 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.

7.2 Implementing Custom Plug-ins

eView provides a simple method of incorporating custom Java code into an eView 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 eView 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 eView 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 eView 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 eView Project, click the **Custom Plug-ins** folder and then right-click.
2. Select **Build** from the context menu that appears.

Chapter 8

Creating the Database

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

8.1 Database Scripts

The eView Wizard creates two scripts based on information you specified about code lists and about external systems. Use these scripts to define startup data for the master index. Generating the Project creates additional scripts for creating or dropping database tables. These scripts appear under the **Database** node of the eView Project, and are named **Systems**, **Code List**, **Create <*application_name*> database**, **Drop <*application_name*> database** (where **<*application_name*>** is the name you defined for the application in the eView Wizard). You can modify these scripts as needed to customize the tables, indexes, startup data, and distribution of the database.

8.2 Requirements

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

Database Platforms

The master index 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.*

8.3 Database Structure

The master index 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 *eView Studio Reference Guide*.

8.4 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.

8.4.1. 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.

8.4.2. Data Analysis

Before beginning the master index implementation, you performed an analysis of the legacy data to help you define the object structure and the attributes of each field. You can use this data analysis to determine the amount of data that will be stored in the database, which will help you size the master index database and decide how to best distribute the database. Knowing the volume of existing data plus the expected daily transaction volume will help you plan the requirements of the database server, such as networking needs, disk space, memory, swap space, and so on.

The data analysis will also help you define the processing codes and the descriptions to define in the common tables.

8.4.3. 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 objects in your database. For example, if you are indexing person objects, there may be processing codes for genders, marital statuses, nationalities, and so on. The processing codes and their descriptions are stored in a set of database tables known as common maintenance tables. The eView Wizard creates a script to help you load the processing codes into the database.

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

8.4.4. 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.

8.4.5. Considerations

When you create the master index database, you need to consider several factors, such as sizing, distribution, indexes, and extents. By default, all of the master index database tables are installed in the Oracle “system” tablespace. You 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. 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.

8.5 Creating the Database

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

- **Step 1: Analyze the Database Requirements** on page 88
- **Step 2: Create an Oracle Database** on page 88
- **Step 3: Customize the Database Scripts** on page 88
- **Step 4: Modify the Database** on page 93
- **Step 5: Specify a Starting EUID (optional)** on page 94

8.5.1. Step 1: Analyze the Database Requirements

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

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

8.5.2. Step 2: Create an Oracle Database

Before beginning this step, be sure that the correct version of Oracle is installed on the database server. eView 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.

8.5.3. Step 3: Customize the Database Scripts

The database script that installs the database components specific to eView is created from the information you specify in the eView Wizard. You can modify any of the database script as needed.

Defining Indexes

To optimize data processing in the master index, you can define additional indexes for the database tables that store object data. SeeBeyond recommends defining indexes for each field used for searching, blocking, or matching. You can define these indexes in the **Create <Object> database** 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 <Object> database** 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 <Object> database** file

Defining Systems

The eView Wizard defines SQL statements to insert the systems you specified. These statements are provided in the **Systems** file in the eView Project.

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 7. 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 7 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 7 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 eView Wizard defines SQL statements to insert custom data into 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 eView Wizard creates a stanza in the **Code List** file (located under the **Database** node of the Project) for each code list you specified in the field properties. You must specify the information to enter for each stanza. 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 must be customized.

3. In the first code list stanza, change “module description” in the first line to a brief description of the code type. For example:

   ```
   -- ****  PHONTYPE   ****
   tCode('L', 'PHONTYPE', 'TELEPHONE TYPE'),
   ```

4. Copy the second line of the stanza (this line begins with “tCode(‘V’...”), and paste the copied line into the same stanza for each data element you need to define for that type.

5. In the copied lines, change “code” to the processing code of the data element, and change “code description” to the description of the element as you want it to appear on the Enterprise Data Manager windows. For example:

   ```
   -- ****  PHONTYPE   ****
   tCode('L', 'PHONTYPE', 'TELEPHONE TYPE'),
   tCode('V', 'H', 'HOME'),
   tCode('V', 'C', 'CELL'),
   tCode('V', 'F', 'FAX'),
   tCode('V', 'O', 'OFFICE'),
   tCode('V', 'HB', 'HOME BUSINESS'),
   ```

6. Repeat steps 3 through 5 for each code list type defined in the file.

   If you specified additional code list fields in the Object Definition file, add a new stanza for each new code type.

7. 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')
   ```

8. Save and close the **Code List** file.

Defining User Code Lists

If you specified a value for the **Constrained By** and **User Code** properties 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.

```
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 right-click.
2. In the Database context menu, select **New**.
3. Enter the name of the script, and then click **OK**.

   The file opens in the text editor.
4. Use the above sample to define a value for the user code drop-down list and the required format for the dependent fields.
5. Repeat step 4 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 8 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

Table 8 SBYN_USER_CODE Table Description

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

Creating a Custom Script

You can insert additional information into the database by creating a custom script under the **Database** node. For information about the structure of the master index database, see the *eView Studio Reference Guide*.

To create a custom script

1. In the eView 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.

8.5.4. Step 4: Modify the Database

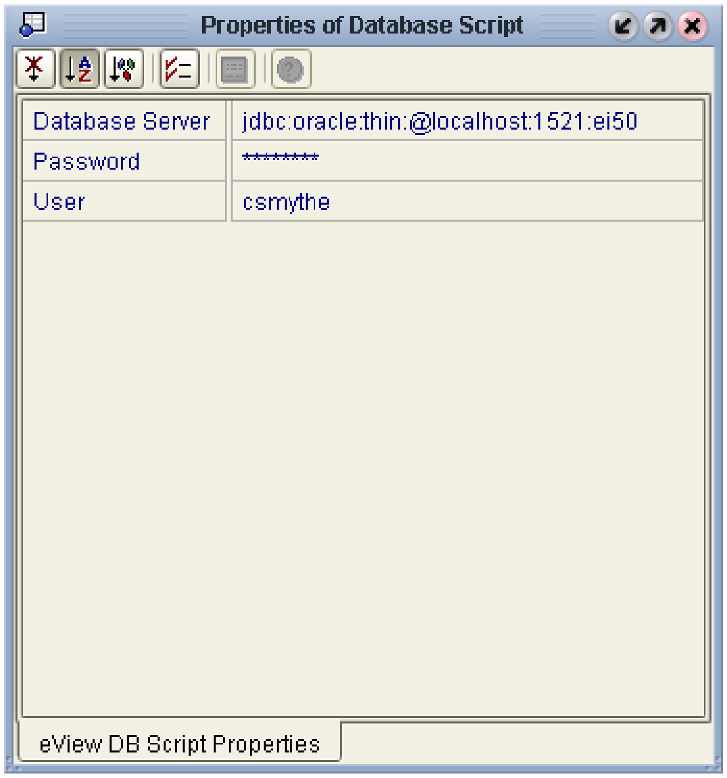
After you create the database instance and customize the database scripts, you can create the master index tables and insert the custom data.

To modify the database

1. In the eView 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 28 on page 94**.

Figure 28 Database Properties Dialog



Database Server	jdbc:oracle:thin:@localhost:1521:ei50
Password	********
User	csmythe

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 <*app_name*> Database** (where **<*app_name*>** is the name of the eView application), 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**.

8.5.5. Step 5: Specify a Starting EUID (optional)

By default, the EUIDs assigned by the master index 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 *eView Studio 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';
```

8.6 Deleting the Master Index Tables

If you need to remove the database tables created in **Step 4: Modify the Database** on page 93, you can run the “drop script” created by the eView Wizard. This is useful while testing the master index implementation.

To delete the master index tables

1. Right-click **Drop <*app_name*> database** (where **<*app_name*>** is the name of the eView application).
2. Select **Run**.
3. On the confirmation dialog, click **OK**.

Chapter 9

Defining Connectivity Components

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

9.1 Overview

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

9.1.1. Connectivity Components

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

eView Project Connectivity Components

The connectivity components in an eView Project include:

- **Connectivity Map**—Graphically describes the relationship between the web application files and the master index application files.
- **Application file**—Contains the logic used by the master index to process data into and out of the master index database. This file is automatically created when you generate the eView Project.

- **Web application file**—Contains the logic used by the Enterprise Data Manager to process data and access the master index logic and database. This file is automatically created when you generate the eView Project.
- **JMS Topic**—A message destination conforming to the *publish-and-subscribe* (pub/sub) messaging paradigm. In this case, eView publishes to the topic to broadcast messages to external systems.

Client Project Connectivity Components

The connectivity components in an eView 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 eView master index application. The Connectivity Map also contains the configuration information for each component's connections—for example, the polling interval and transactional behavior.
- **eView application**—Represents the master index application accessed by the client Project. Each external system or Web connector in the eView system must include the eView application in its Connectivity Map in order for those components to exchange information with the master index.
- **Service**—Provides a framework for a process or a Collaboration, 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 master index 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 eView Project for use in the client Projects to define how data is processed between the master index and external systems or eInsight. A custom OTD is also created to publish messages from the master index to a JMS Topic.
- **External Applications**—Logical representations of external software applications (called *external systems*) that are integrated by the eGate system. External Applications allow the master index to connect with external systems via eGate. 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 the master index 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.

9.2 Defining Connectivity Components

Defining connectivity components begins with creating a graphical representation of the connectivity components (the Connectivity Map). You must define connectivity components for the eView Project and for any client Projects integrating with the eView application. This section describes how to define connectivity components for eView Projects, external system Projects, and Projects integrating eView with eInsight. Perform the following tasks to configure connectivity for the master index and connected systems.

- **Defining eView Application Connectivity Components** on page 98
- **Defining External System Connectivity Components** on page 100
- **Defining eInsight Integration Connectivity Components** on page 109

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

9.2.1. Defining eView Application Connectivity Components

In the eView Project, the Connectivity Map contains business logic and information about how data is processed in the master index. This section describes how to create a Connectivity Map for the eView Project, add components to the map, and then connect those components. Perform these tasks to create and configure the eView Project Connectivity Map.

- **Creating the eView Project Connectivity Map** on page 98
- **Connecting Connectivity Map Components** on page 99

Creating the eView Project Connectivity Map

This section describes how to create, and add components to, the eView Project Connectivity Map.

To create the eView Project Connectivity Map

1. In Enterprise Explorer, select the eView 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 as shown in Figure 29.

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

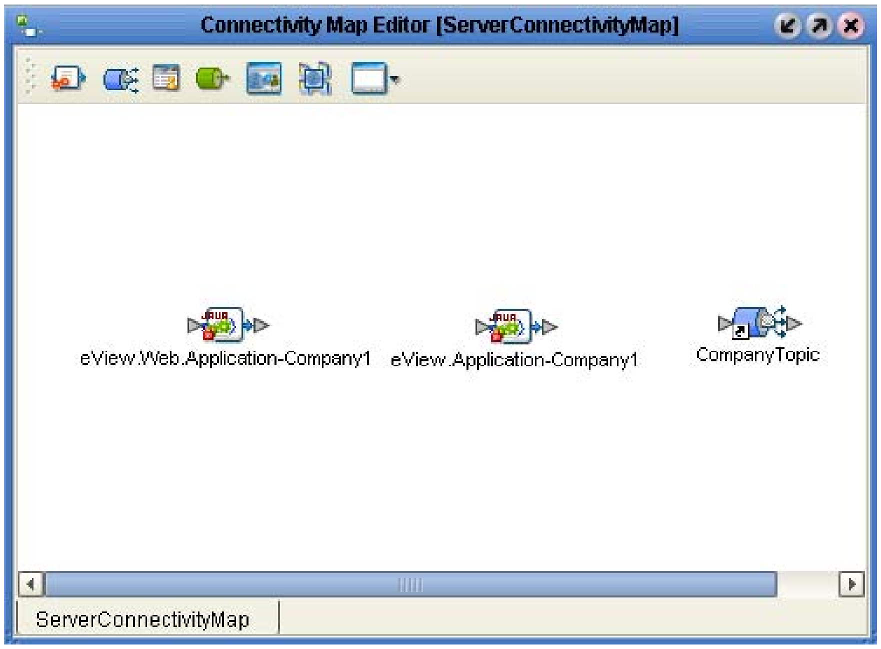
5 Drag the **eView.Web.Application-<application_name>** icon from the Project Explorer onto the Connectivity Map Editor.

6 Drag the **eView.Application-<application_name>** icon from the Project Explorer onto the Connectivity Map Editor to the right of the web application icon.

7 To publish messages from the master index to external systems, select the **Topic** icon in the Connectivity Map toolbar, and then drag it to the right of the **eView.Application-<application_name>** icon in the editor.

The Connectivity Map should now look like Figure 29.

Figure 29 eView Server Connectivity Map


Connectivity Map Editor [ServerConnectivityMap]
eView.Web.Application-Company1
eView.Application-Company1
CompanyTopic
ServerConnectivityMap

8 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 within the application. Before you connect the components, make sure you have completed all of the steps in **"Creating the eView Project Connectivity Map" on page 98**.

1 In the Connectivity Map Editor, place the cursor over the arrow to the right of the **eView.Web.Application-<application_name>** icon until the cursor turns into a hand.

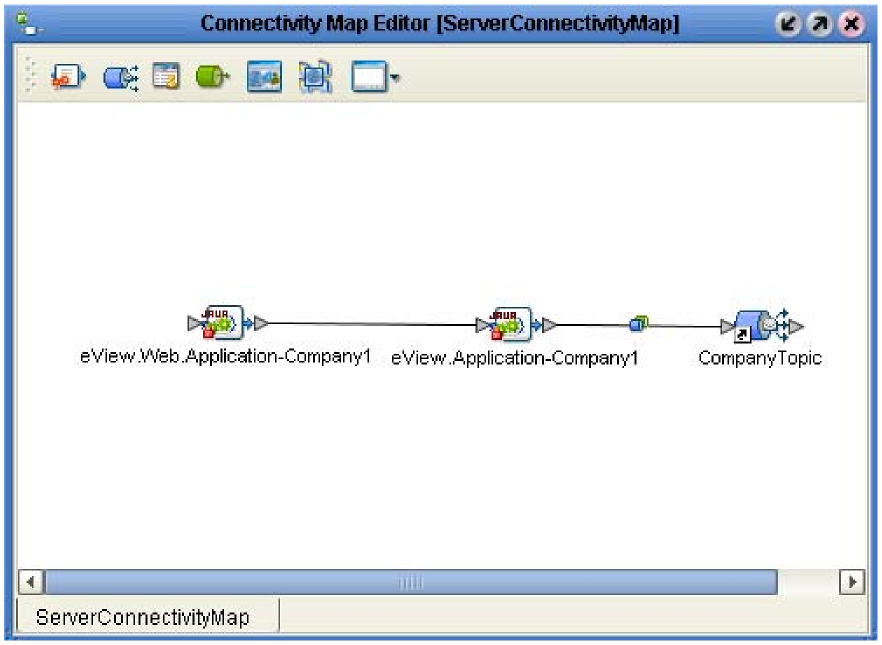
2 Click the arrow and drag it to the **eView.Application-<application_name>** icon.

3 Release the mouse button to link the two components.

4 If you added a JMS Topic to the Connectivity Map, repeat steps 1 through 3, dragging the arrow from the right side of the **eView.Application-<application_name>** icon to the **Topic** icon.

Figure 30 illustrates the completed Connectivity Map for the eView Project.

Figure 30 eView Server Connectivity Map with Connections


Connectivity Map Editor [ServerConnectivityMap]
eView.Web.Application-Company1
eView.Application-Company1
CompanyTopic
ServerConnectivityMap

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

6 Save the Connectivity Map to the Repository.

9.2.2. Defining External System Connectivity Components

In the client Projects for external systems sharing data with the master index, the Connectivity Map contains business logic and information about how data is transferred between the master index and external systems. This section describes how to create a Connectivity Map for an external system Project, add and configure map components, and then connect those components. Perform these tasks to configure the connectivity components.

- **Adding eView Methods to a Java Collaboration** on page 101
- **Creating the External System Project Connectivity Map** on page 102
- **Connecting Connectivity Map Components** on page 104

Adding eView Methods to a Java Collaboration

This section describes how to incorporate the eView method OTD into Java Collaborations for external systems. For a complete reference of the methods included in the eView OTD, see the *eView Studio Reference Guide*.

To add eView methods to a Java Collaboration

1. Create the Java Collaboration for the external system Project using the Java Collaboration Wizard (select the Project, right-click, and then select **New -> Java Collaboration Definition**).
2. Enter information into the wizard as it applies to the external system (for more information, see the *eGate Integrator User's Guide*).
3. In step 3 of the Java Collaboration Wizard (**Select OTDs**), select the input OTD, the output OTD, and the method OTD from the eView Project.

Note: *The eView method OTD is contained in the eView Project and is named after the eView application.*

4. Use the Java Collaboration Editor to define custom processing using Java methods. To use eView methods in the Collaboration, do the following:

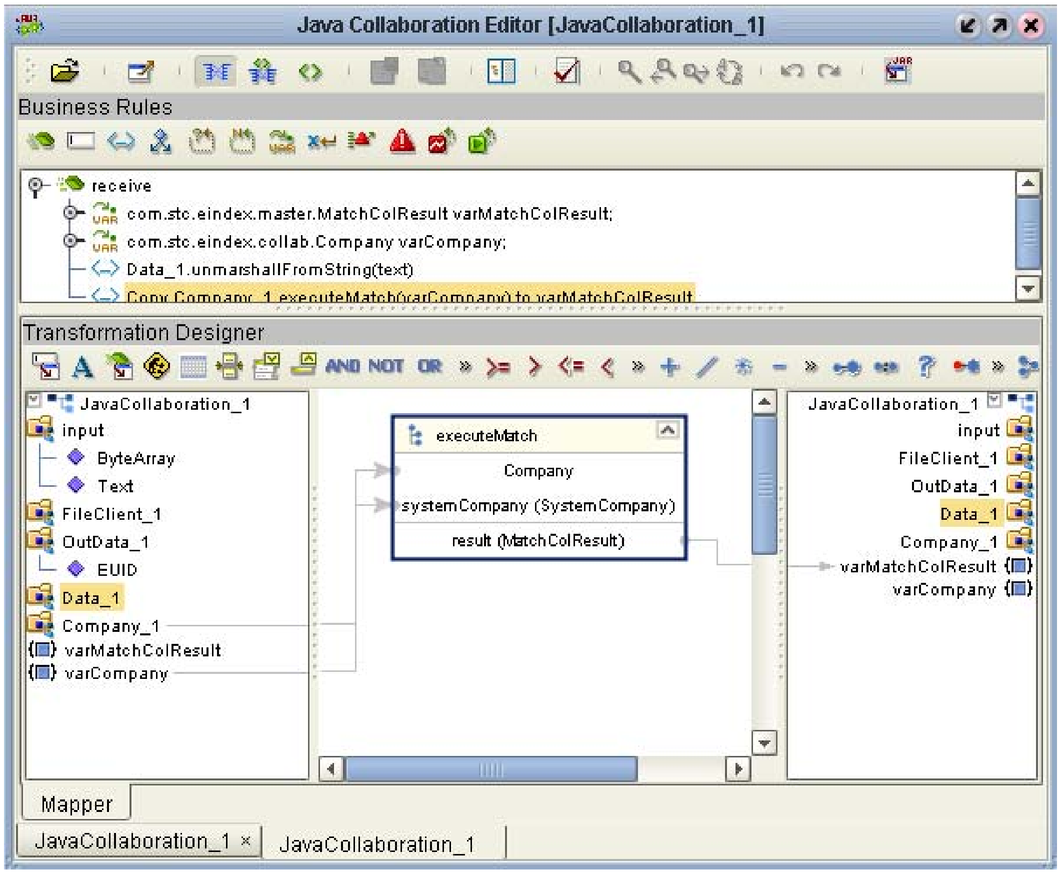
   A. In the left pane of the Transformation Designer, right-click the eView method OTD. A list of available methods appears.

   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 31 illustrates a sample of the **executeMatch** method in the Java Collaboration Editor.

Figure 31 eView Method OTD in Java Collaboration Editor



```
receive
    com.stc.eindex.master.MatchColResult varMatchColResult;
    com.stc.eindex.collab.Company varCompany;
    Data_1.unmarshalFromString(text)
    Copy Company_1.executeMatch(varCompany) to varMatchColResult
```

5 When you are done defining the processing rules, save the Collaboration.

Creating the External System Project Connectivity Map

To define connectivity between the eView application and external systems, you must include the eView application in the Connectivity Maps of the external systems. This section describes how to incorporate the eView application into the Connectivity Map.

Note: *Before beginning this procedure make sure an external system component is defined for a system that sends information to the master index (source system) and possibly for a system that receives information from the master index (destination system).*

To create an external system 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.

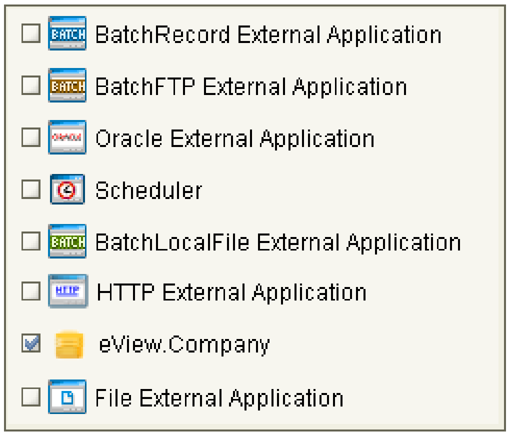
5 Drag the external source system icon from the Project Explorer to the Connectivity Map Editor.

6 Drag the Java Collaboration you created using the eView method OTD onto the Connectivity Map Editor to the right of the external source system icon.

7 If you defined an external destination system, drag its icon from the Project Explorer onto the Connectivity Map Editor to the upper right of the **Collaboration** icon.

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 eView Application you want to integrate.

Figure 32 External System Menu



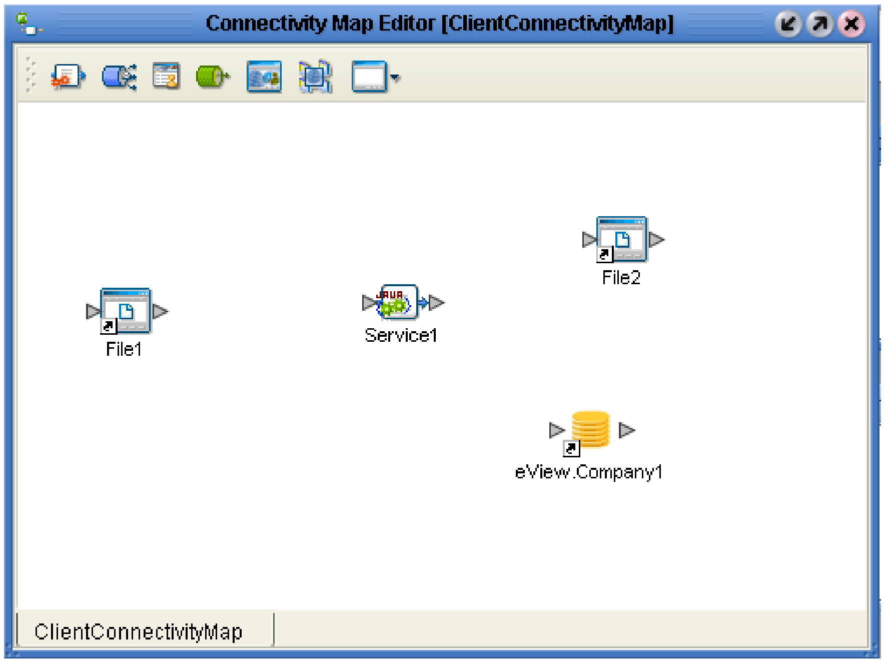
☐ BatchRecord External Application
☐ BatchFTP External Application
☐ Oracle External Application
☐ Scheduler
☐ BatchLocalFile External Application
☐ HTTP External Application
☑ eView.Company
☐ File External Application

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

9 Drag the eView application icon from the Connectivity Map Editor toolbar onto the Connectivity Map Editor to the lower right of the **Collaboration** icon.

The Connectivity Map should now look similar to Figure 33.

Figure 33 External System Connectivity Map



10 Save the Connectivity Map.

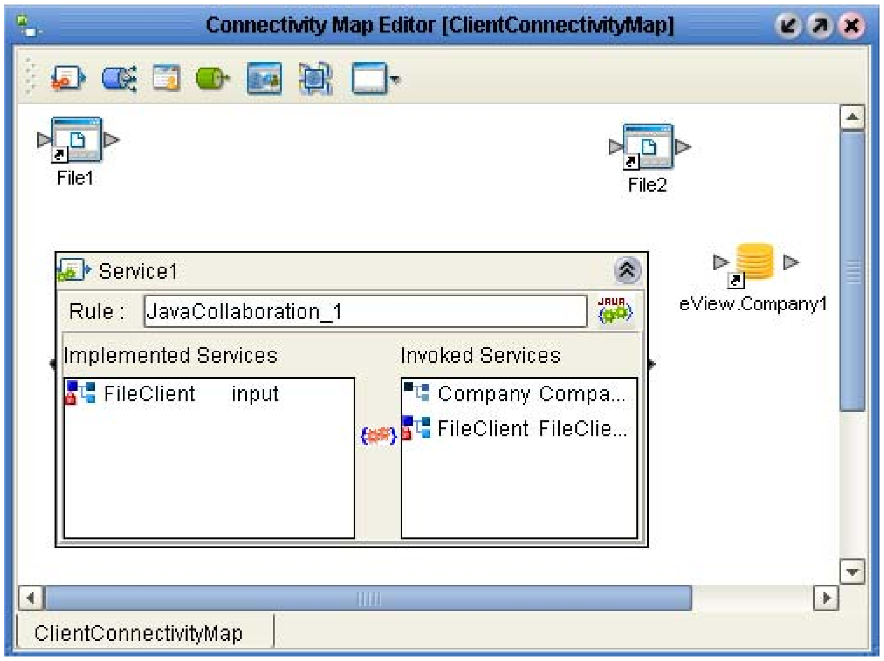
Connecting Connectivity Map Components

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 External System Project Connectivity Map" on page 102**.

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 34.

Figure 34 Collaboration Binding Window, External Systems



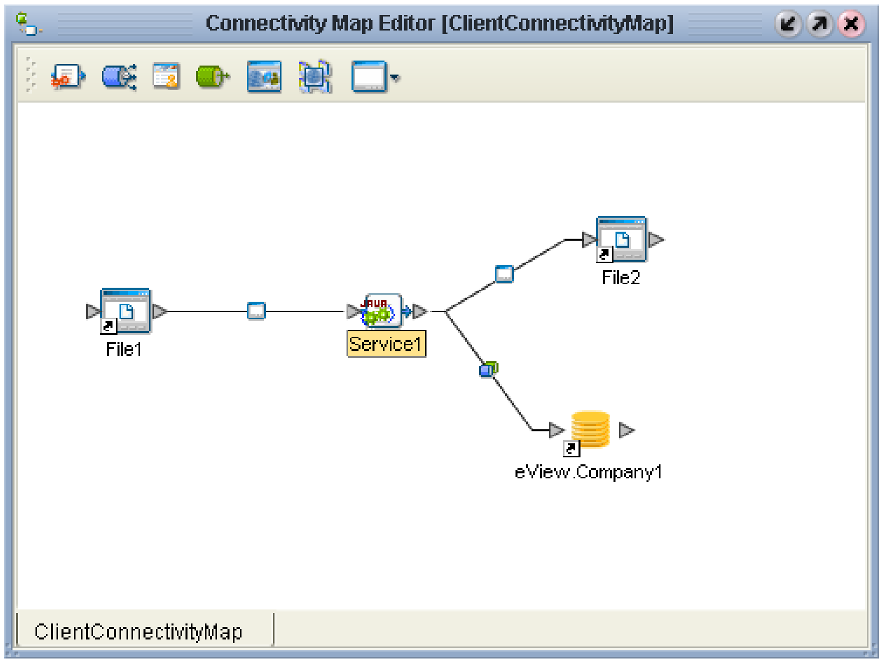
2 Drag the source system from the **Implemented Services** box in the Collaboration Binding window to the external source system icon on the Connectivity Map Editor.

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

4 If you defined a destination system, drag the appropriate service from the **Invoked Services** box in the Collaboration Binding window to the external destination system icon on the Connectivity Map Editor.

5 Close the Collaboration Binding window. The Connectivity Map connections should look similar to Figure 35.

Figure 35 External System Connectivity Map With Connections



6 Configure the eWays (for more information, see the *eGate Integrator User's Guide*).

7 Save the Connectivity Map to the Repository.

Incorporating the JMS Topic into the Connectivity Map

If you defined a JMS Topic in the eView server Connectivity Map to which eView messages will be published, you must add the topic to the client Connectivity Map in order to publish the messages to external systems. This involves two primary steps: adding the JMS Topic and associated components to the Connectivity Map and configuring the Collaboration for the connected Service.

To add the JMS Topic to the Connectivity Map

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

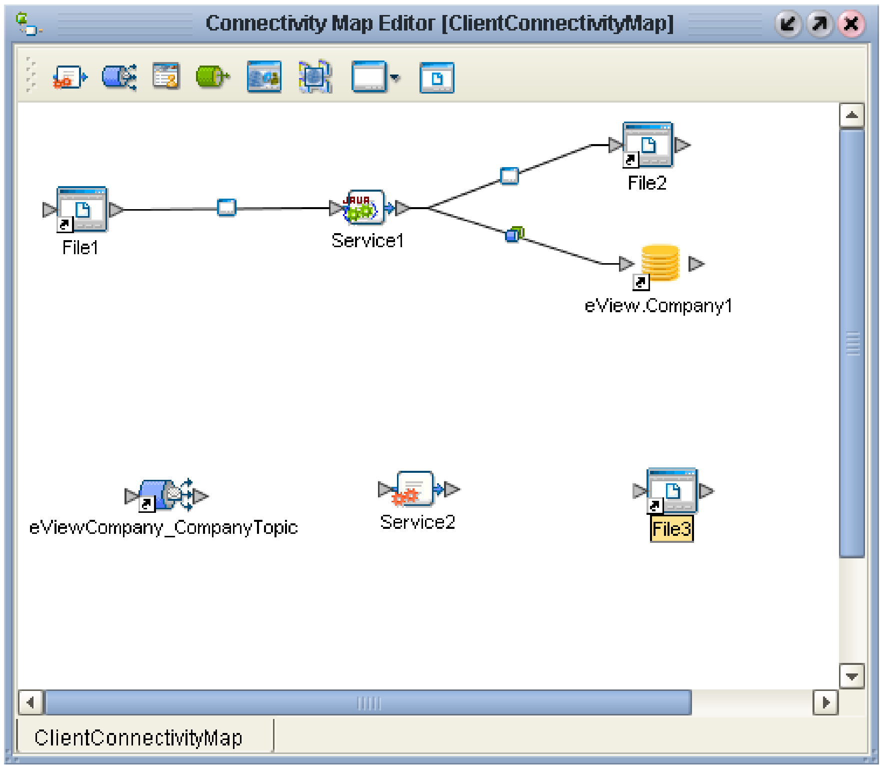
2 Drag the JMS Topic from the eView Project to the Connectivity Map Editor, below the existing connectivity diagram (this is the same topic you created in **"Defining eView Application Connectivity Components" on page 98**).

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 like Figure 36.

Figure 36 eView Client Connectivity Map with JMS Topic

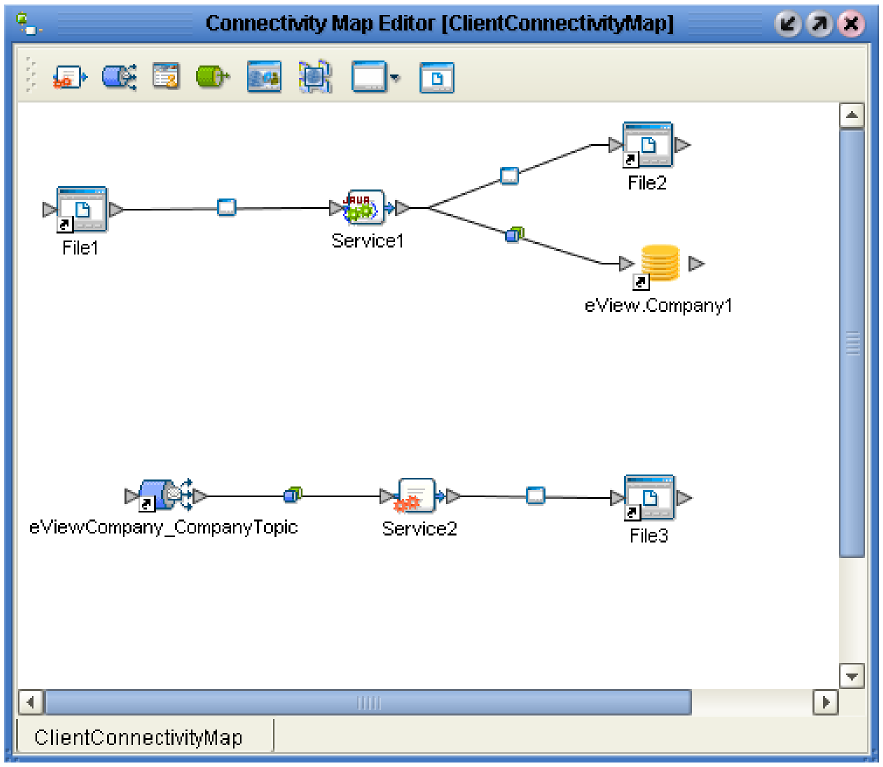

Connectivity Map Editor [ClientConnectivityMap]
File1
Service1
File2
eView.Company1
eViewCompany_CompanyTopic
Service2
File3
ClientConnectivityMap

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 37.

Figure 37 eView Client Connectivity Map with Connections


Connectivity Map Editor [ClientConnectivityMap]
File2
File1
Service1
eView.Company1
eViewCompany_CompanyTopic
Service2
File3
ClientConnectivityMap

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

8 Double-click the new external system eWay to configure the eWay parameter settings.

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 eView JMS Topic. Before you begin, make sure you have completed all of the steps in **"Incorporating the JMS Topic into the Connectivity Map" on page 106**.

To configure the outbound Collaboration

1 In the Project Explorer, right-click the eView client Project.

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

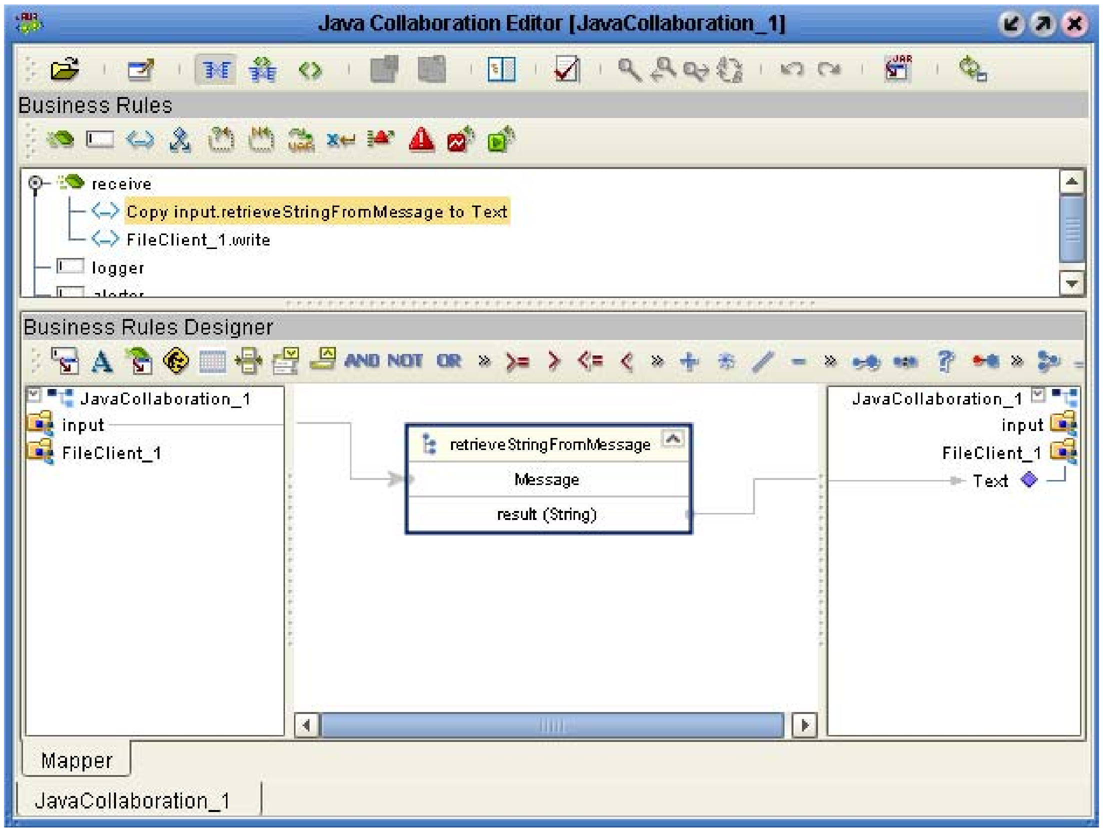
3 Enter information into the Java Collaboration Definition Wizard, with the following guidelines:

- 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 38).

Figure 38 Outbound Java Collaboration



5 Save the Collaboration to the Repository.

6 Open the eView client Connectivity Map, and drag the newly created Collaboration onto the Service connected to the JMS Topic.

7 Save the Connectivity Map to the Repository.

9.2.3. Defining eInsight Integration Connectivity Components

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

- **“Including eView Methods in a Business Process” on page 110**

- **"Creating the eInsight Integration Connectivity Map" on page 112**
- **"Connecting Connectivity Map Components" on page 114**

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 eView Methods in a Business Process

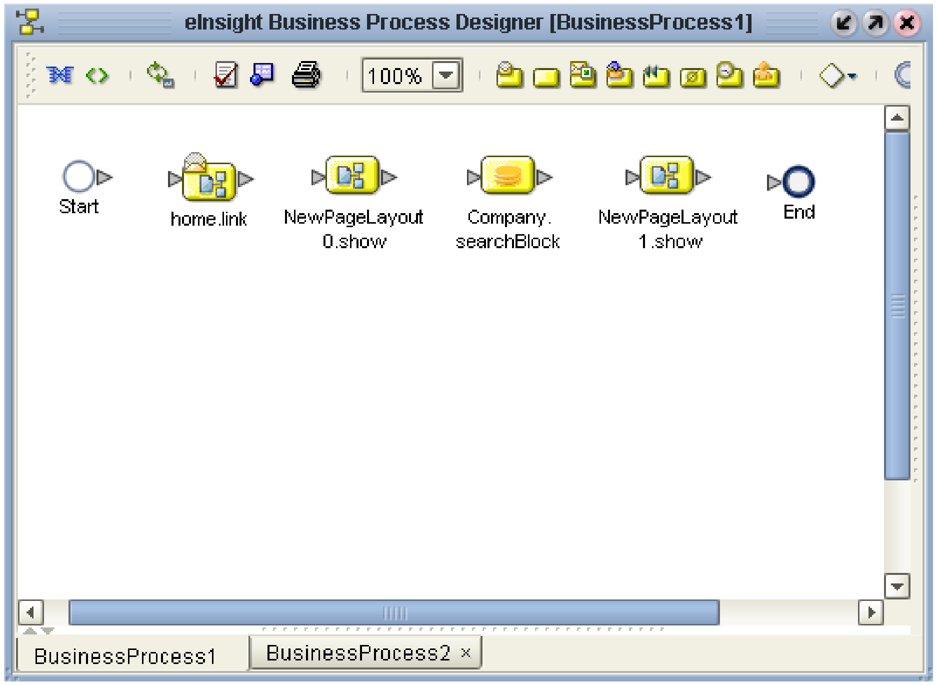
Before including the eView application in the Connectivity Map for eInsight integration, you must add eView methods to a Business Process. This section provides instructions for adding an eView method to an eVision Web page in a Business Process. For more information about the available methods, see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

To include eView 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 master index 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 eView 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 39).

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

Figure 39 eInsight Business Process with eView 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 eView methods to eVision Web pages. Make sure you have completed all of the steps in **"Including eView Methods in a Business Process" on page 110**.

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.

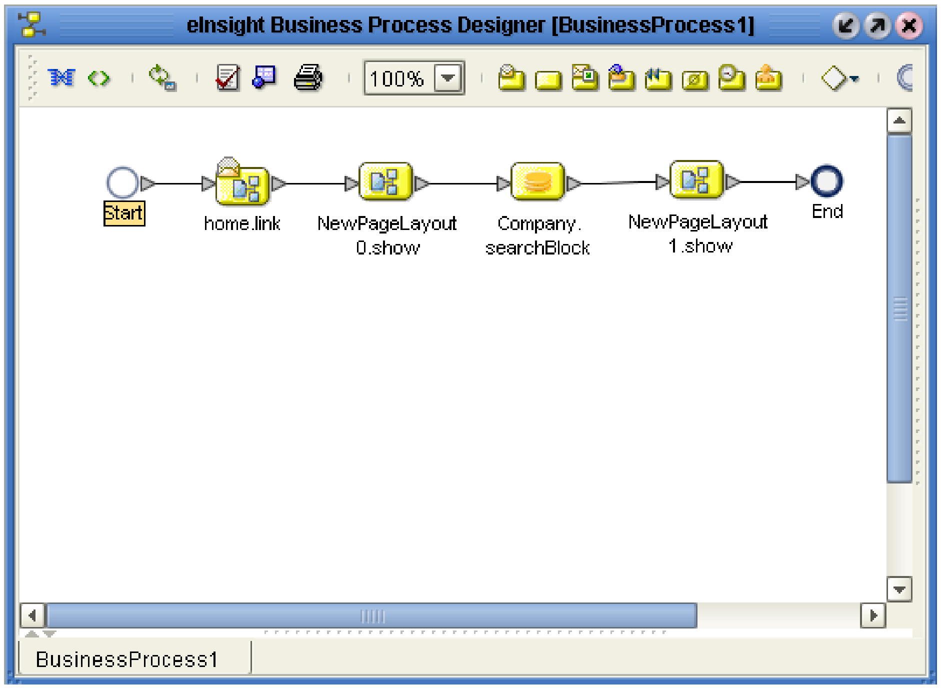
3 Follow the same procedure to link the following activities:

- Link the **Page Link** activity to the input web page activity.
- Link the input web page activity to the **eView** activity.
- Link the **eView** 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 40.

Figure 40 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.

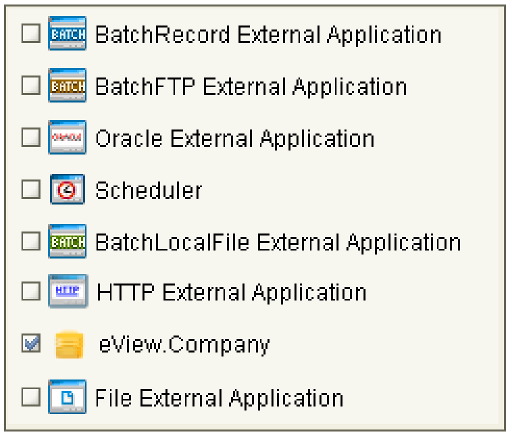
5 Drag a **Web Connector** icon from the Connectivity Map Editor toolbar onto the canvas.

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 eView 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 eView application you want to integrate.

Figure 41 External System Menu

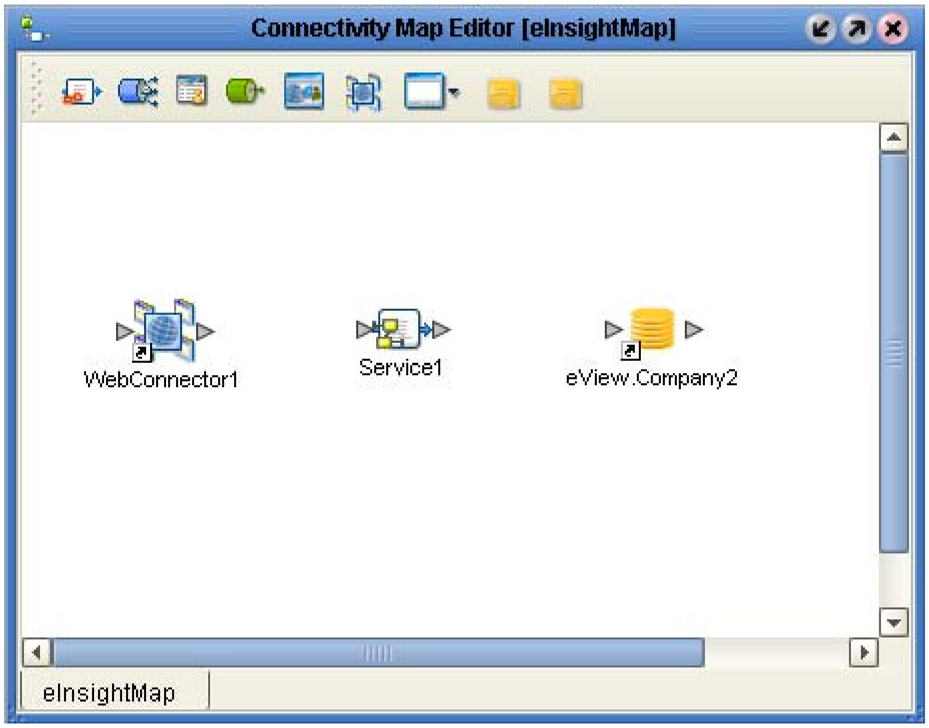

☐ BatchRecord External Application
☐ BatchFTP External Application
☐ Oracle External Application
☐ Scheduler
☐ BatchLocalFile External Application
☐ HTTP External Application
☑ eView.Company
☐ File External Application

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

9 Drag the eView 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 42.

Figure 42 eInsight Integration Connectivity Map



10 Save the Connectivity Map to the Repository.

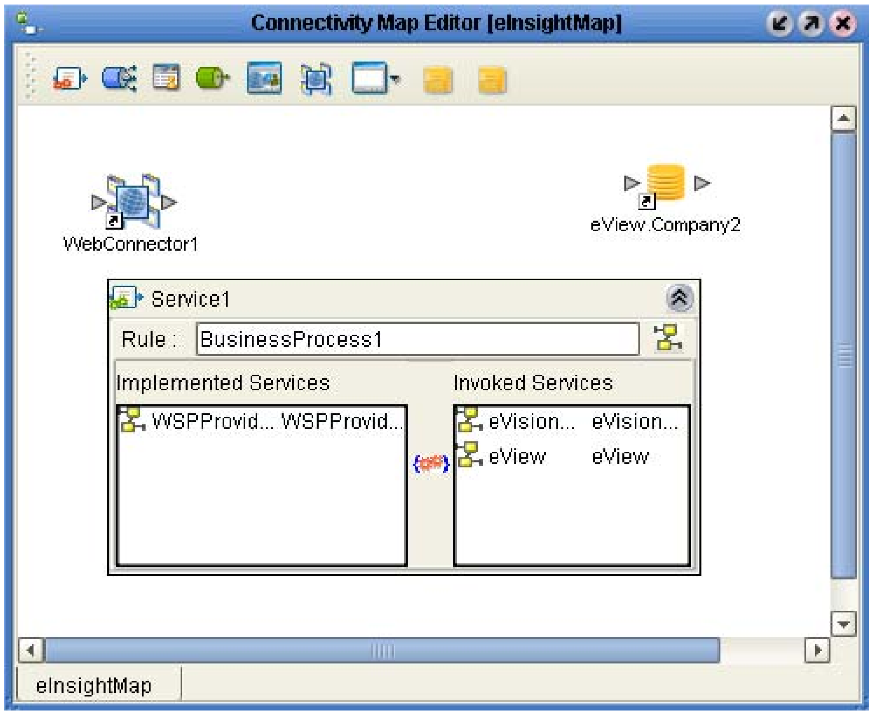
Connecting Connectivity Map Components

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 112**.

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 43.

Figure 43 Collaboration Binding Window, eInsight



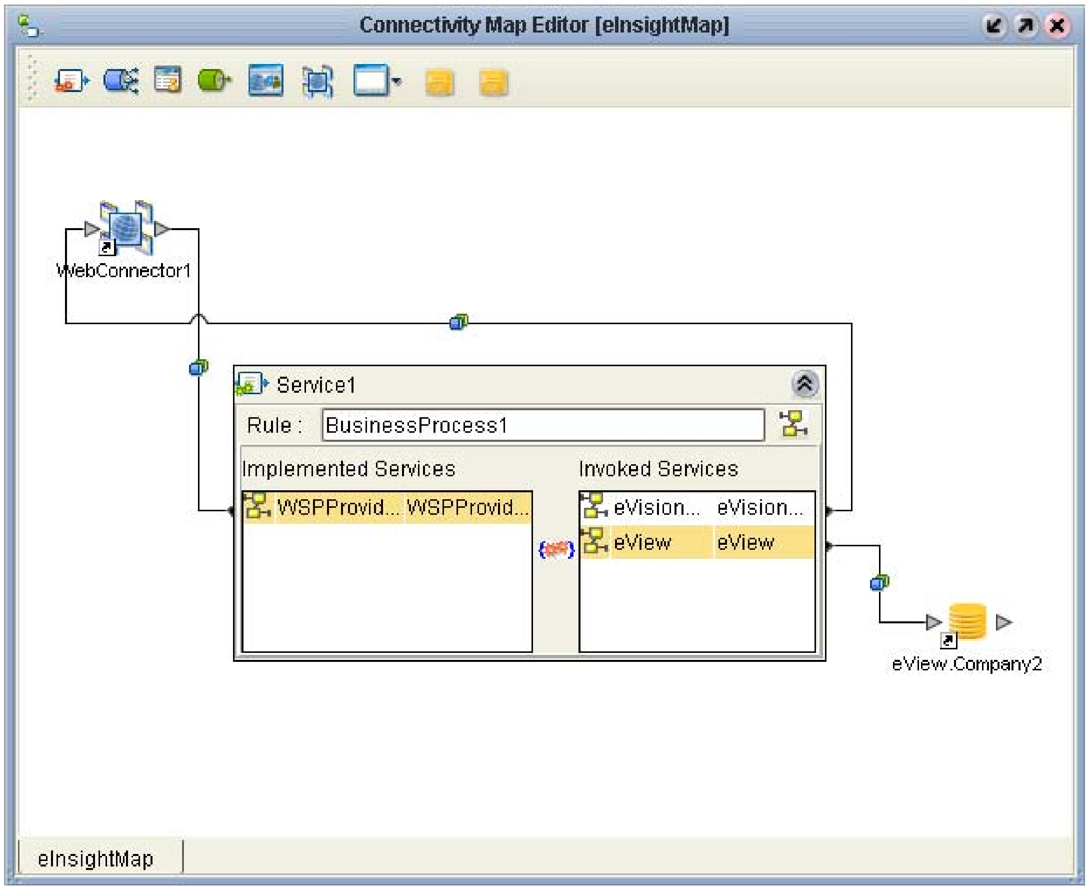
2 Drag **eVision** from the **Invoked Services** box in the Collaboration Binding window to the **Web Connector** icon on the Connectivity Map Editor.

3 Drag the eView application from the **Invoked Services** box in the Collaboration Binding window to the **eView 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.

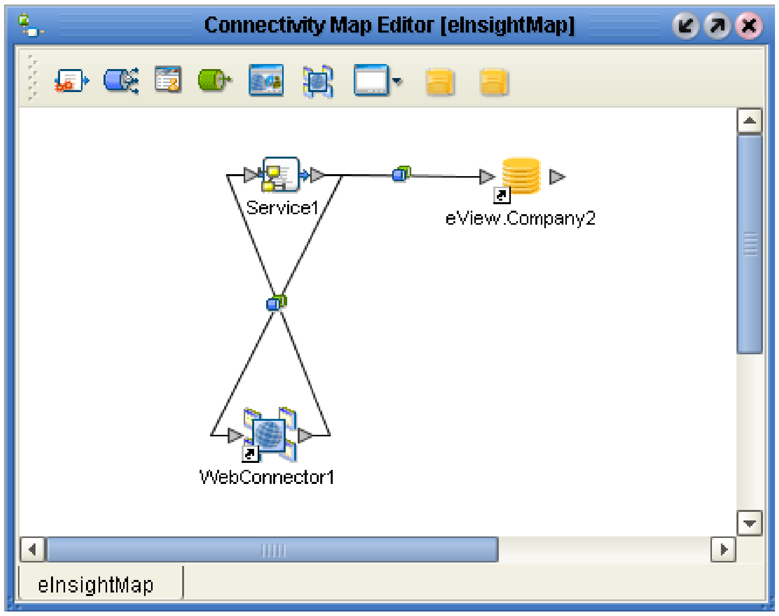
Figure 44 illustrates the Connectivity Map with the connections in place.

Figure 44 Collaboration Binding Window Connections, eInsight



5 Close the Collaboration Binding window. The Connectivity Map window should now look similar to Figure 45.

Figure 45 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 10

Defining the Environment

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

10.1 Environment Components

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

Environment Components

The Environments configured to support the eView 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 eView application.
- **eVision External Systems**—An eVision external system is a representation of an eVision Web application. This component is required for Projects integrating eView with eInsight.

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

10.2 Building an Environment

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

- **"Creating an eView Environment" on page 118**
- **"Adding a Logical Host" on page 119**
- **"Adding Servers" on page 120**
- **"Adding an External System" on page 121**
- **"Configuring the Integration Server" on page 123**

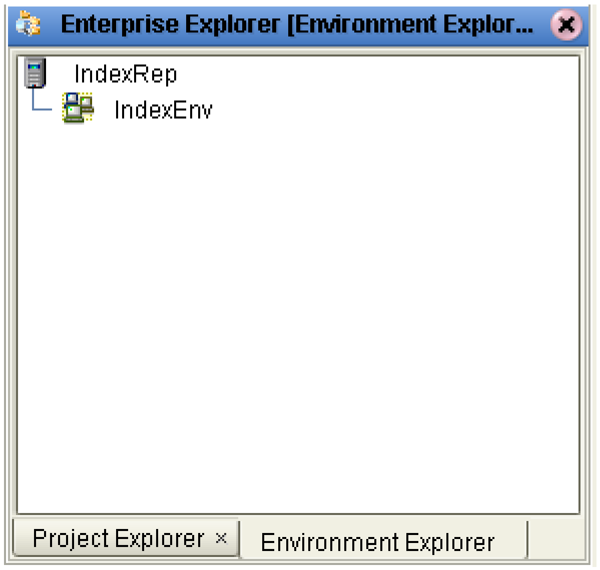
10.2.1. Creating an eView Environment

This section describes how to create an Environment for an eView 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 46.

Figure 46 New Environment



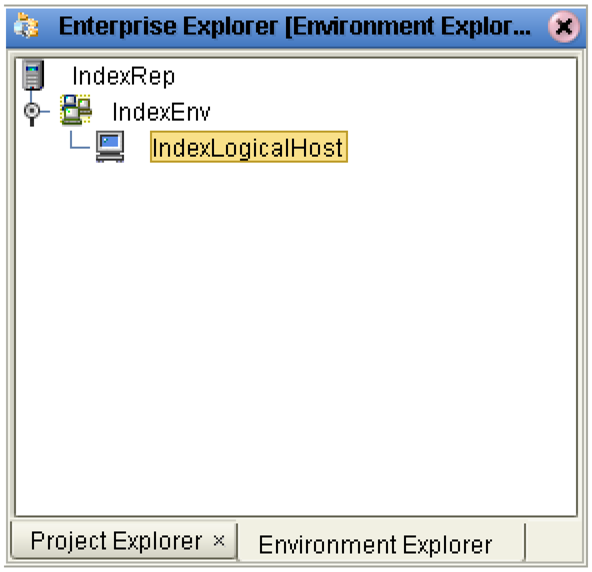
10.2.2. Adding a Logical Host

This section describes how to add a logical host to the eView Environment. This is a required step for all eView 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 47.

Figure 47 eView Logical Host



10.2.3. Adding Servers

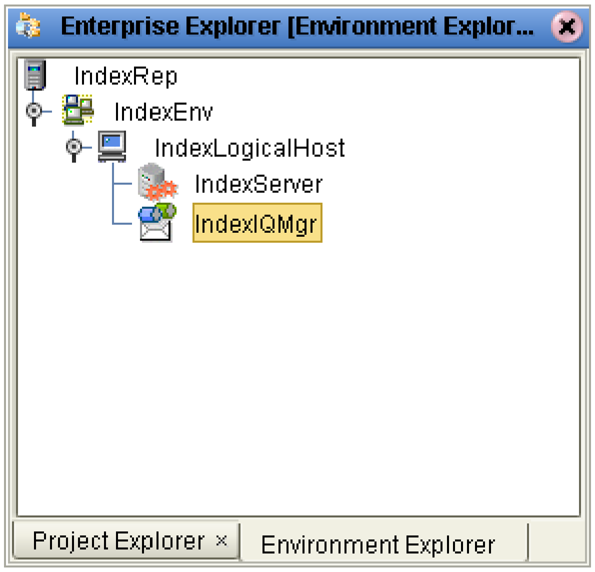
This section describes how to add integration servers and JMS IQ Managers to the eView Logical Host. You must add an integration server to the Environment.

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 eView 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 48 illustrates an Environment with a Logical Host, SeeBeyond Integration Server, and SeeBeyond JMS IQ Manager.

Figure 48 Integration and JMS Servers



7 Configure the connection to the database, as described in **"Configuring the Integration Server" on page 123**.

10.2.4. Adding an External System

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

To add an external system

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

2 Select the eView Environment icon.

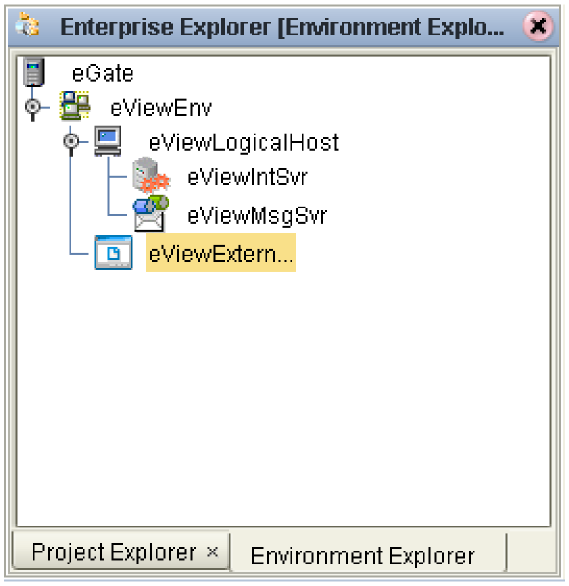
3 Right-click the eView Environment icon to display the **Environment** context menu.

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 49.

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

Figure 49 File External System



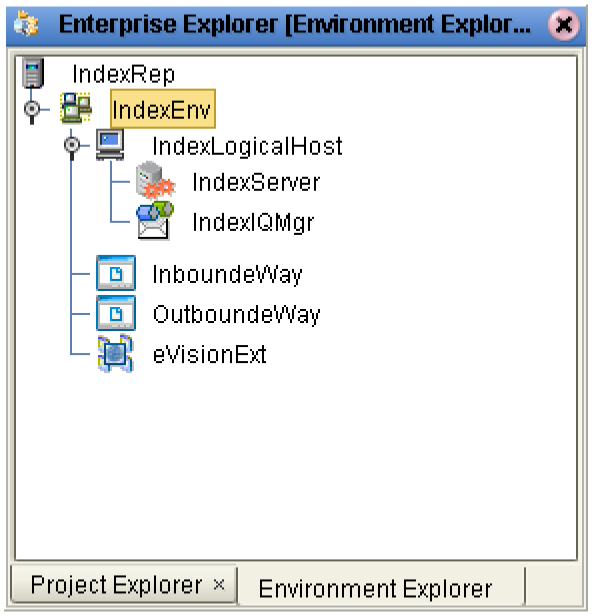
10.2.5. Adding an eVision External System

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

To add an eVision external system

1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the eView Environment icon.
3. Right-click the eView 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 50.
6. Configure the eVision external system properties.

Figure 50 eVision External System



10.2.6. 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 the master index, 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

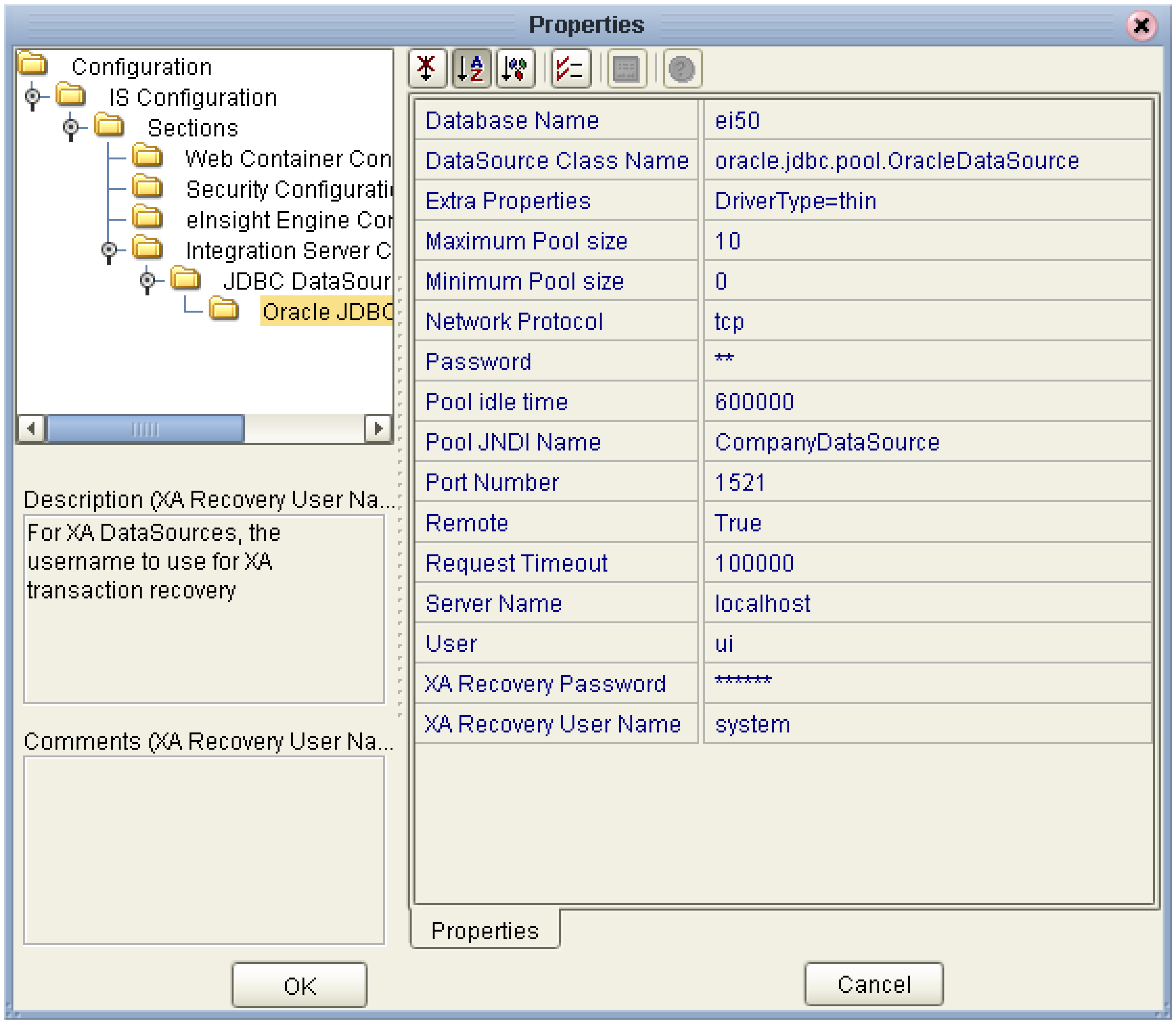
1. In Enterprise Explorer, click the **Environment Explorer** tab.
2. Select the Integration Server icon in the Environment you want to configure.
3. Right-click the icon to display the **Integration Server** context menu.
4. Select **Properties** to display the **Properties** window.
5. Expand the configuration 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 the master index database you created. For more information about the properties on this window, see Table 9 on page 125.

Note: *Creating the database is described in* Chapter 8 *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 51 Integration Server Properties



Description (XA Recovery User Na...

For XA DataSources, the username to use for XA transaction recovery

Comments (XA Recovery User Na...

Database Name	ei50
DataSource Class Name	oracle.jdbc.pool.OracleDataSource
Extra Properties	DriverType=thin
Maximum Pool size	10
Minimum Pool size	0
Network Protocol	tcp
Password	**
Pool idle time	600000
Pool JNDI Name	CompanyDataSource
Port Number	1521
Remote	True
Request Timeout	100000
Server Name	localhost
User	ui
XA Recovery Password	******
XA Recovery User Name	system

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

For the ***Pool JNDI Name*** *property, you must enter* ***<app_name>DataSource****, where* ***<app_name>*** *is the name of the application defined for the eView implementation. This allows you to configure one integration server for multiple eView applications. 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 9 Data Source Properties

Property	Description
Database Name	The SID name of the eView 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 eView application as defined in the eView Wizard.
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 eView 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.

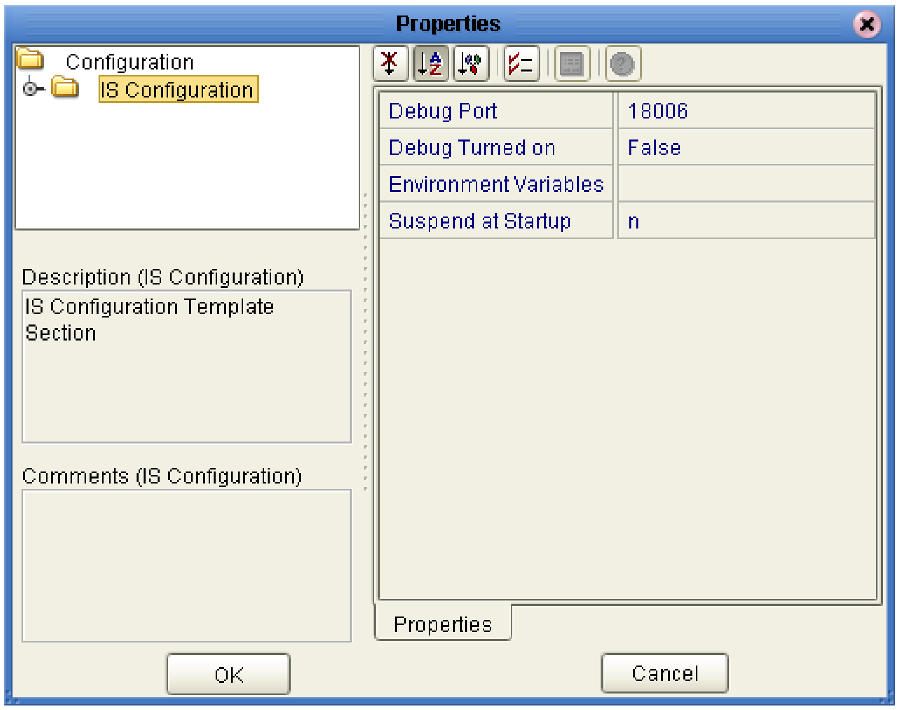
Defining Environment Variables for INTEGRITY

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 52.

Figure 52 Integration Server Configuration Properties



Description (IS Configuration)
IS Configuration Template Section

Comments (IS Configuration)

Debug Port	18006
Debug Turned on	False
Environment Variables	
Suspend at Startup	n

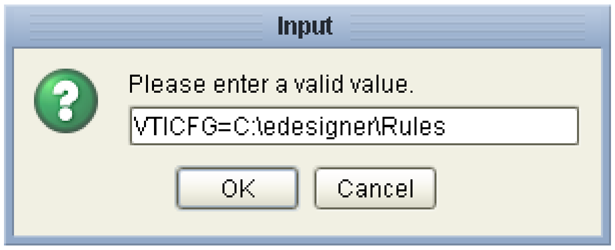
6. Click in the **Environment Variables** field, and then click the ellipses to the right of the field.

   The **Environment Variables** properties dialog appears.
7. For each environment variable listed in **Table 10 on page 128**, 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 53.

Figure 53 Input dialog



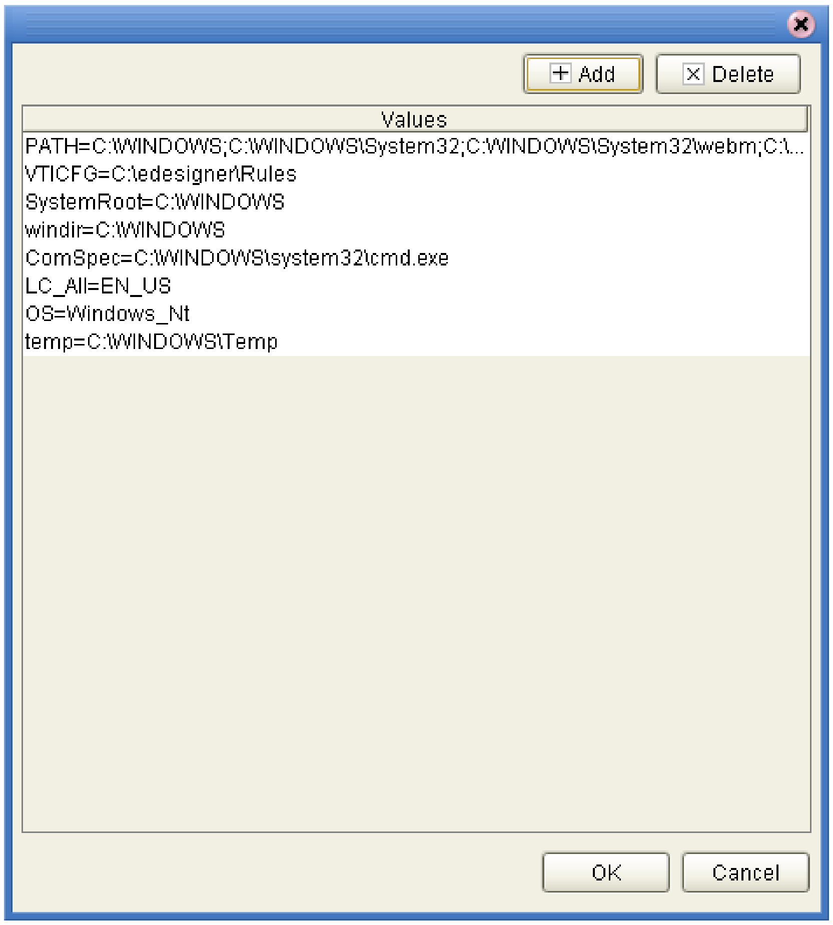
Please enter a valid value.

VTICFG=C:\edesigner\Rules

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

After you defined all variables, the window should look similar to Figure 54.

Figure 54 Defined Environment Variables



Values
PATH=C:\WINDOWS;C:\WINDOWS\System32;C:\WINDOWS\System32\wbem;C:\...
VTICFG=C:\edesigner\Rules
SystemRoot=C:\WINDOWS
windir=C:\WINDOWS
ComSpec=C:\WINDOWS\system32\cmd.exe
LC_All=EN_US
OS=Windows_Nt
temp=C:\WINDOWS\Temp

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 10 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

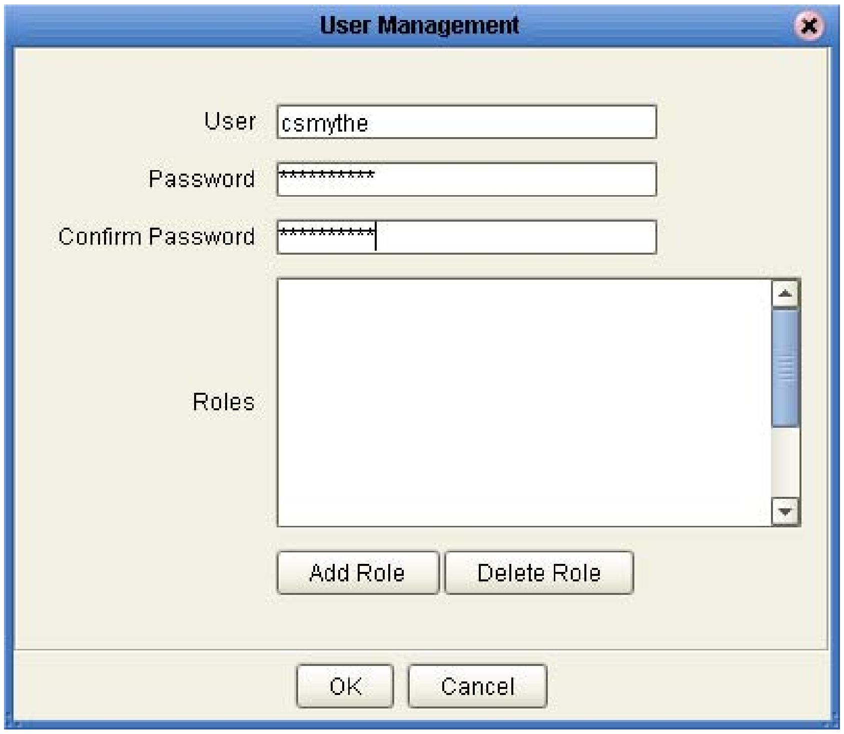
10.2.7. Defining Security

A secure user name and password must be defined for the integration server in order to connect to the master index 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 eView Environment icon.
3. Right-click the eView Environment icon to display the **Environment** context menu.
4. Select **User Management** to display the **User Management** dialog as shown in Figure 55.
5. Enter the user name, password, and confirmation password of the eView user.

Figure 55 User Management



User: csmythe

Password: ***********

Confirm Password: ***********

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**.

eView user roles are listed and described in Table 11.

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 11 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 any custom masking logic specified by the Security configuration file.
AL.View	Gives access permission to view audit log entries.

Table 11 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 11 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 11

Deploying the Project

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

11.1 Overview

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

11.2 Deploying a Project

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

- **"Defining a Deployment Profile" on page 132**
- **"Activating the Project" on page 140**
- **"Running the Bootstrap" on page 141**

11.2.1. Defining a Deployment Profile

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

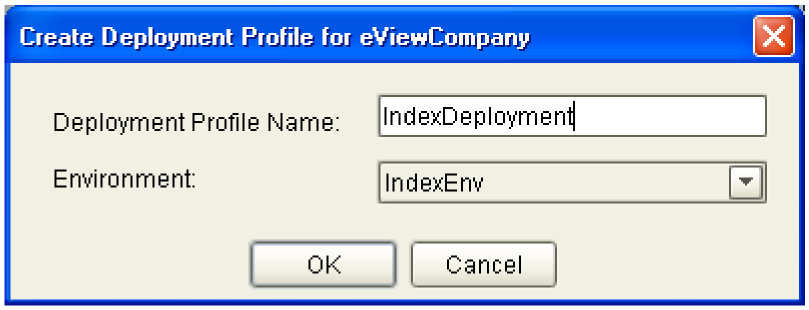
Defining an eView Application Deployment Profile

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

To create an eView application Deployment Profile

1. In Enterprise Explorer, click the **Project Explorer** tab.
2. Select the eView 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 56.

Figure 56 Create Deployment Profile Dialog Box



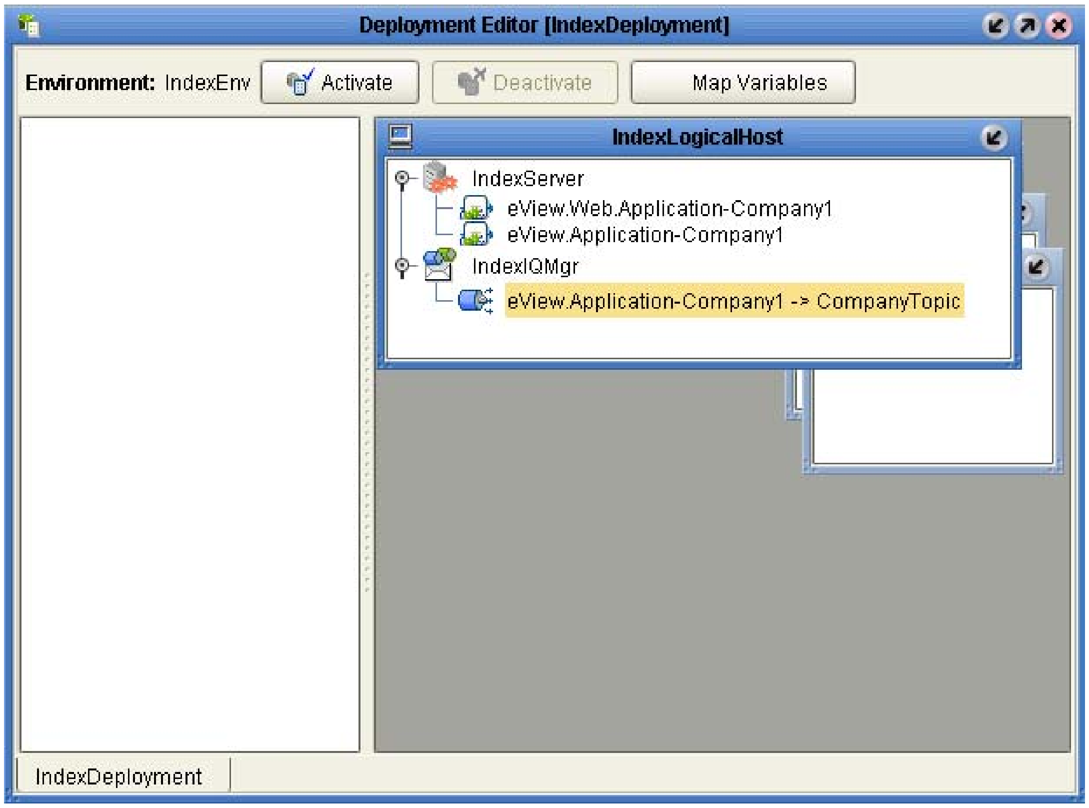
Create Deployment Profile for eViewCompany

Deployment Profile Name: IndexDeployment

Environment: IndexEnv

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 eView Project.
7. Click **OK** to add a **Deployment Profile** icon to the eView Project and display the Deployment Editor window shown in Figure 57.

Figure 57 Deployment Editor Window - Server Project


Deployment Editor [IndexDeployment]
Environment: IndexEnv
Activate
Deactivate
Map Variables
IndexLogicalHost
IndexServer
eView.Web.Application-Company1
eView.Application-Company1
IndexIQMgr
eView.Application-Company1 -> CompanyTopic
IndexDeployment

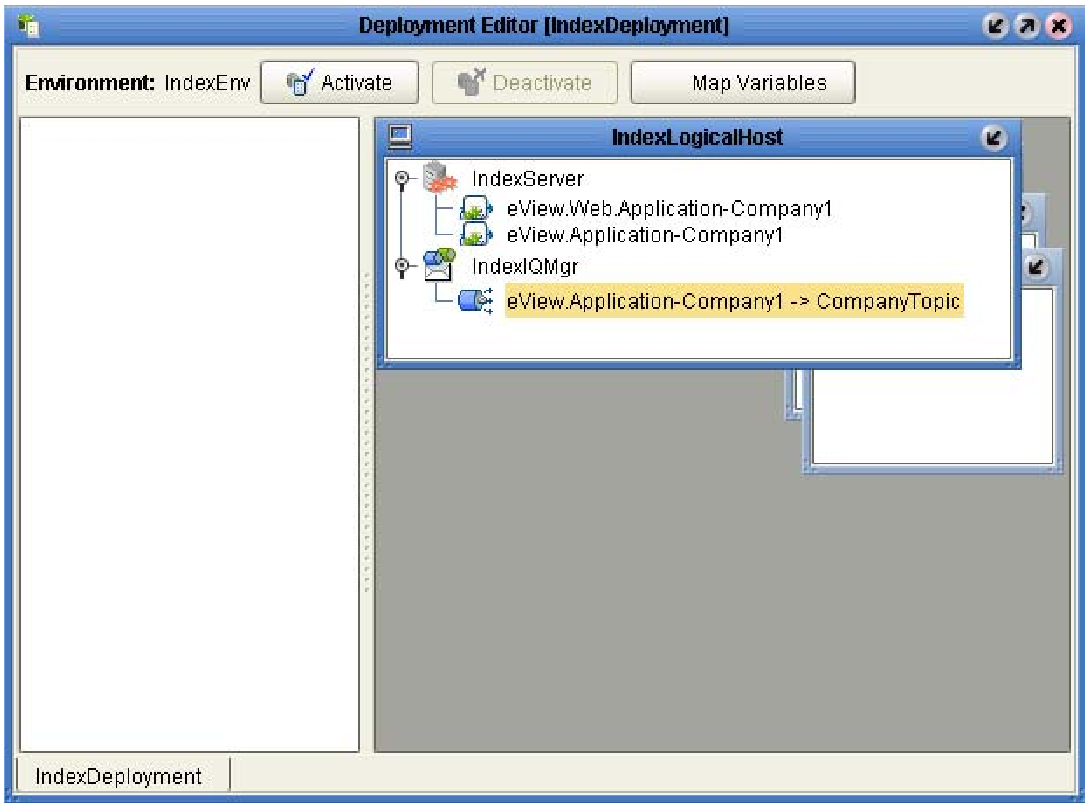
To map eView Project components

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 eView Project Deployment Profile open in the Deployment Editor, drag the **eView.Web.Application-<*application_name*>** and **eView.Application-<*application_name*>** 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 the master index messages, drag the JMS Topic icon to the JMS IQ Manager in the Logical Host.

   Figure 58 illustrates the updated Deployment Profile.

Figure 58 Mapped eView Components in the Deployment Editor


Deployment Editor [IndexDeployment]
Environment: IndexEnv
Activate
Deactivate
Map Variables
IndexLogicalHost
IndexServer
eView.Web.Application-Company1
eView.Application-Company1
IndexIQMgr
eView.Application-Company1 -> CompanyTopic
IndexDeployment

3 Deploy the application, as described in **"Activating the Project" on page 140**.

Creating an eView Client Deployment Profile

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

To create an eView client Deployment Profile

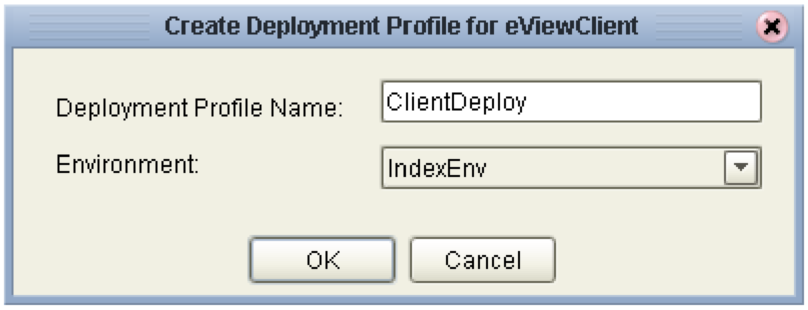
1 In Enterprise Explorer, click the **Project Explorer** tab.

2 Select the eView 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 59.

Figure 59 Create Deployment Profile Dialog Box



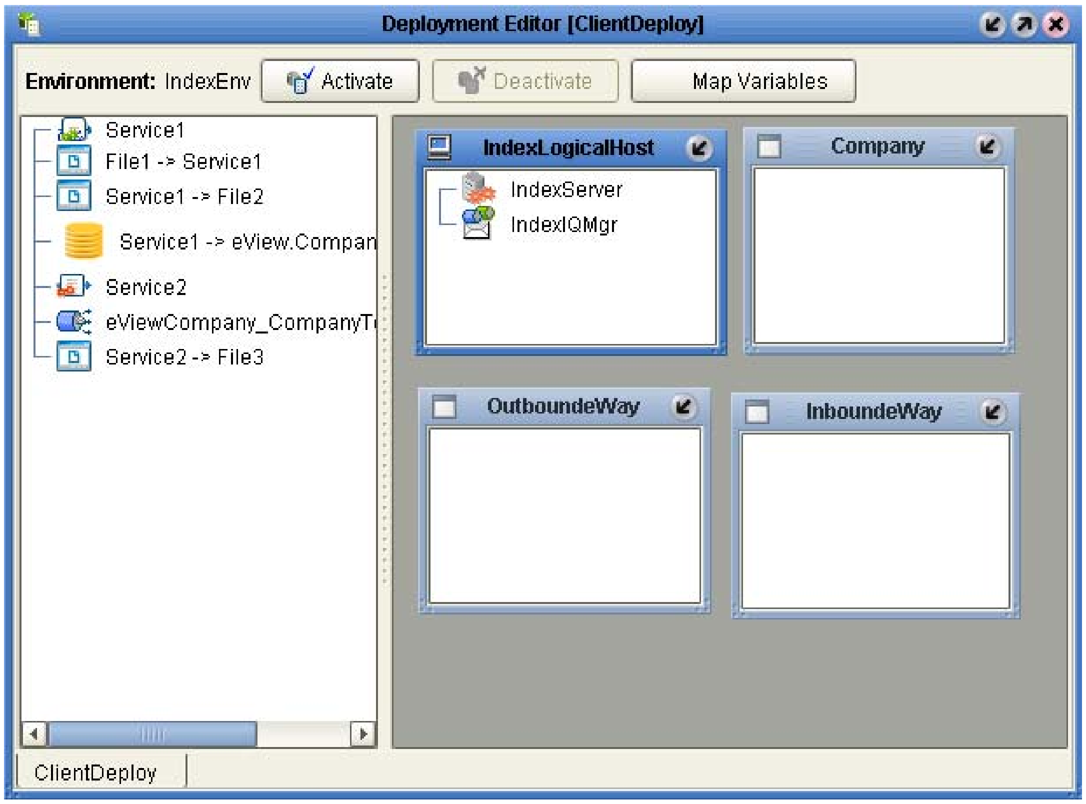
Deployment Profile Name: ClientDeploy
Environment: IndexEnv

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 eView Project.

7 Click **OK** to add a **Deployment Profile** icon to the eView Project and display the Deployment Editor window shown in Figure 60.

Figure 60 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 eView 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.

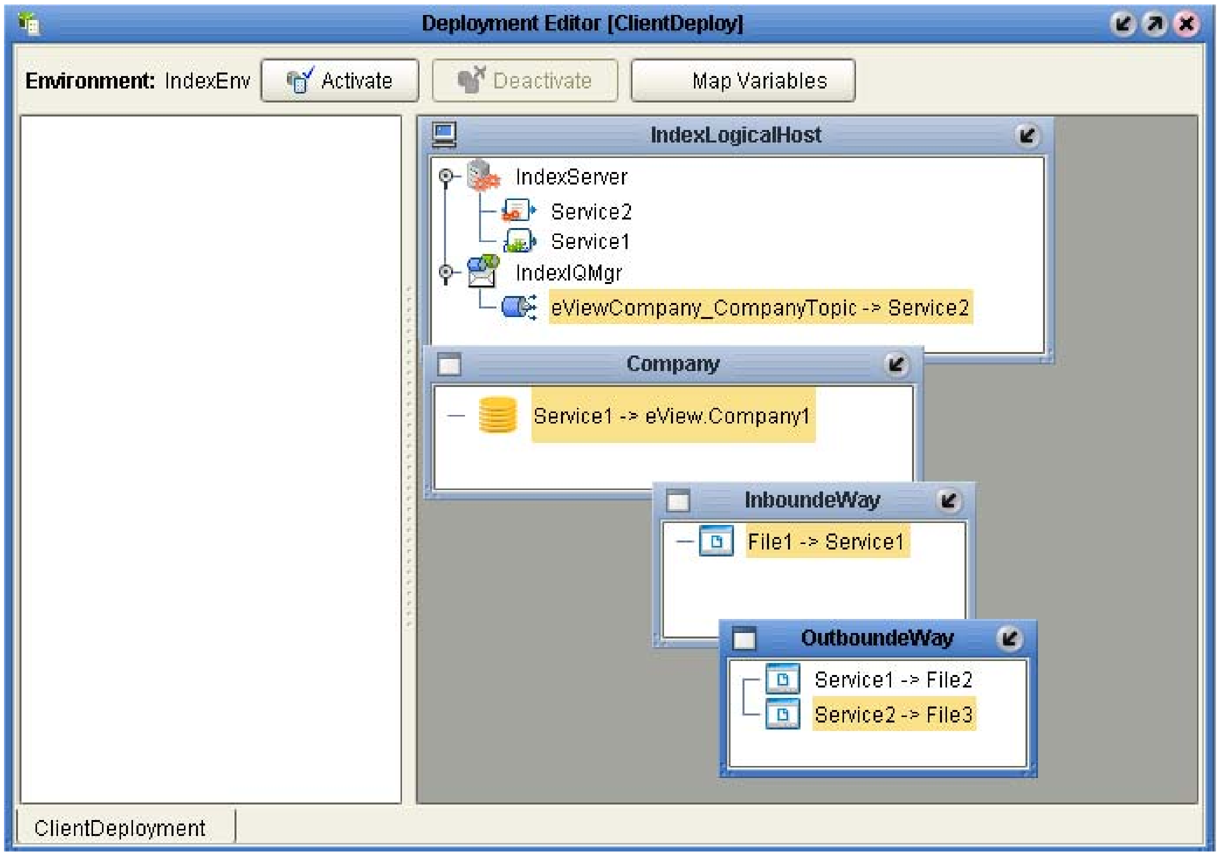
1 With eView Project Deployment Profile open in the Deployment Editor, drag the Service icon(s) onto the SeeBeyond Integration Server in the Deployment Editor (the servers appear in the Logical Host).

2 Drag the eView application icon onto the eView application deployment component (this is named after the eView application).

3 Drag the external system eWays to the appropriate inbound and outbound deployment components (in Figure 61, 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 61 Mapped Client Components in the Deployment Editor



5 Deploy the application, as described in **"Activating the Project" on page 140**.

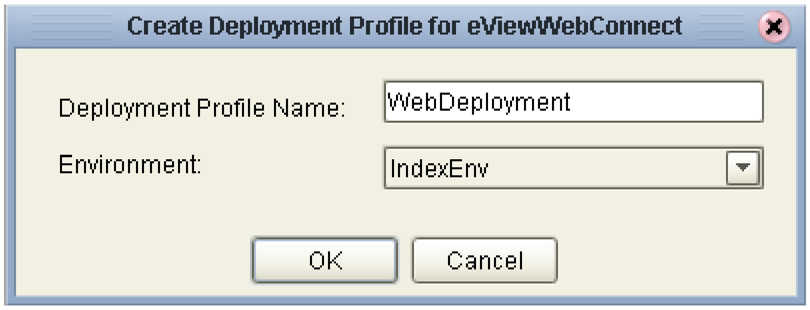
Creating an eInsight Integration Deployment Profile

This section describes how to create a Deployment Profile for the Projects defining Web pages for eView 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 eView 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 62.

Figure 62 Create Deployment Profile Dialog Box

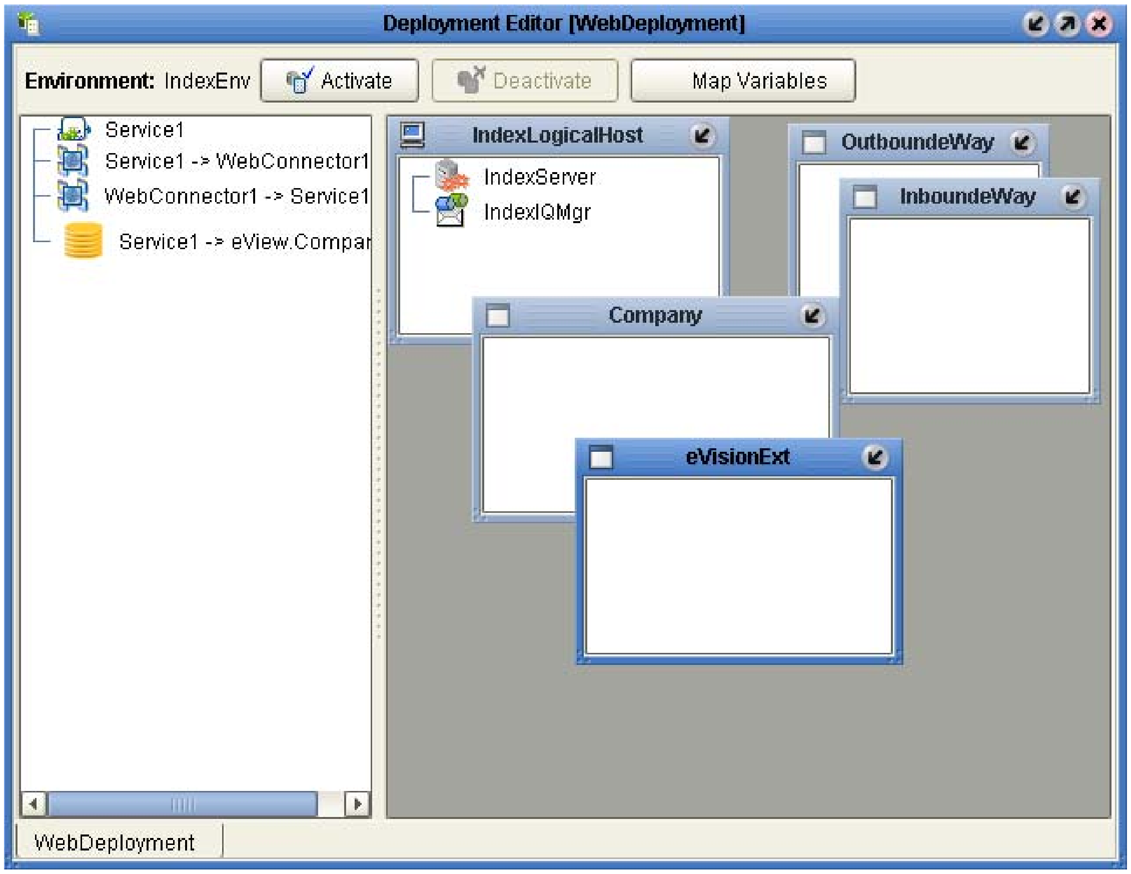

Create Deployment Profile for eViewWebConnect

Deployment Profile Name: WebDeployment

Environment: IndexEnv

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 eView Project.
7. Click **OK** to add a **Deployment Profile** icon to the eView Project and display the Deployment Editor window shown in Figure 63.

Figure 63 Deployment Editor Window



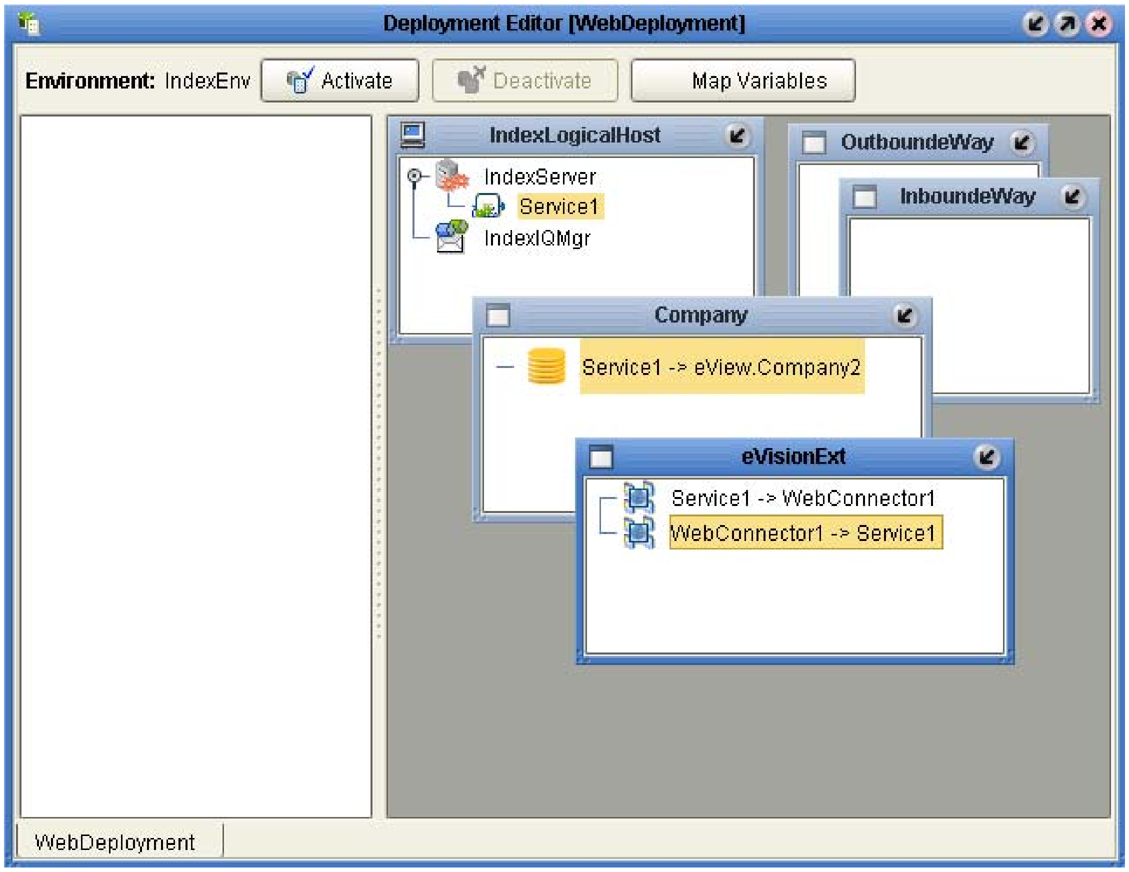
To map eInsight Integration Project components

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 eView application icon onto the eView application deployment component (this is named after the eView application).
3. Drag the eVision Web service(s) to the eVision External Application in the Environment.

   Figure 64 illustrates the updated Deployment Profile.

Figure 64 Mapped Client Components in the Deployment Editor



4 Deploy the application, as described in **"Activating the Project" on page 140**.

11.2.2. Activating the Project

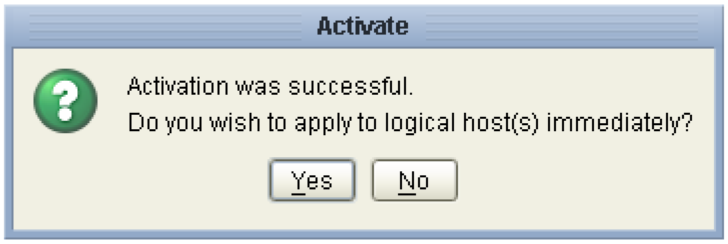
With Project components mapped in the Deployment Profile, you are now ready to activate the Project.

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 65.

Figure 65 Activate Dialog



Activation was successful.
Do you wish to apply to logical host(s) immediately?

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 eView Project before creating client Project Deployment Profiles makes the eView application available to those Profiles.*

11.2.3. Running the Bootstrap

To run the eView master index application, you must run the bootstrap for the Logical Host to which the application was deployed. If any client Projects connected to the master index application 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*.

Chapter 12

Implementing the eView Sample

Sample Projects for a master company index are provided with eView. You can install and run the samples to better understand how the master index works, and how the various components of eView correlate. This chapter explains how to import the sample files, customize the Projects, and then run data into the index.

Important: *In order to work with the sample Projects, you must have a standard Oracle database installed on your computer, and the File eWay must be installed.*

12.1 Overview

The eView sample Project implements a simple master company index. It includes a server and a client Project, along with a small data file in XML format to enter data into the master index through an eGate Service. Once you implement the master index, you can create your own data files to enter through the Service, and you can create and modify data through the EDM.

12.2 Importing the Sample Projects

In order to work with the eView sample Projects, you must import the Projects into your eGate environment. Before you begin this step, make sure you have installed the sample files, as described in **"Uploading the eView Documentation and Sample" on page 39**.

To import the server Project

1. In the Enterprise Designer, select the Repository name and then right-click.
2. From the context menu, select **Import Project**.
3. On the **Select file to import** dialog, navigate to the directory where you extracted the sample files, select **eView_Sample.zip**, and then click **Open**.
4. On the File Destination dialog, select **Import to a new Project**.
5. Enter "eView_Sample" for the name of the Project, and then click **OK**.
6. On the message dialog that appears, click **OK**.

7 On the Enterprise Designer toolbar, click **Refresh all from Repository**.

To import the client Project

1 In the Enterprise Designer, select the Repository name and then right-click.

2 From the context menu, select **Import Project**.

3 On the **Select file to import** dialog, navigate to the directory where you extracted the sample files, select **eView_Sample_Client.zip**, and then click **Open**.

4 On the File Destination dialog, select **Import to a new Project**.

5 Enter "eView_Sample_Client" for the name of the Project, and then click **OK**.

6 On the message dialog that appears, click **OK**.

7 On the Enterprise Designer toolbar, click **Refresh all from Repository**.

12.3 Implementing the Sample Projects

Implementing the sample Project includes the following tasks:

- **Configuring for INTEGRITY** on page 143
- **Regenerate the Application** on page 144
- **Create the Database** on page 144
- **Define the Environment** on page 145
- **Configure the Client Connectivity Map** on page 146
- **Create and Activate the Server Deployment Profile** on page 146
- **Create and Activate the Client Deployment Profile** on page 146
- **Start the Logical Host** on page 147
- **Run the Sample File** on page 147
- **Working with the EDM** on page 147

12.3.1. Configuring for INTEGRITY

If you are using the INTEGRITY match engine, you must customize the Match Field file for the eView sample. If you are using the SeeBeyond Match Engine, you can skip this step. To perform this step, you must have access to the location where the eView INTEGRITY Add-on files were downloaded.

To configure for INTEGRITY

1 In the Project Explorer, expand the **Configuration** folder in the **eView_Sample** Project.

2 Check out, and then open, the Match Field file.

3 In Windows Explorer or a command prompt, navigate to the location where the eView INTEGRITY Add-on files were downloaded, and then to the **eViewSampleApplication** folder.

4 Open the file **mefa.xml**.

5 Copy all text from this file, and paste it into the Match Field file in Enterprise Designer, replacing all existing text.

6 Right-click on **Match Field** in the Project Explorer, and then click **Validate**.

7 If the validation shows no errors, save and close the file.

12.3.2. Regenerate the Application

Before you can work with the application files, you must regenerate the application.

To regenerate the application

1 In the Project Explorer, expand the **eView_Sample** Project.

2 Right-click **eView Application - Company**.

3 Select **Generate** from the context menu.

4 Select **Yes** on the Confirm dialog.

5 When the application is regenerated, close the output window.

12.3.3. Create the Database

To create the sample database tables, you must have a standard Oracle database installed on your computer. If you have not done so already, create the database before continuing. You can use the database files as they are, or you can add additional systems and common table data (see **"Step 3: Customize the Database Scripts" on page 88** for more information). Make sure not to delete or change any of the existing information.

To create the database

1 Create an administrator user for the database. Use the following scripts as a sample.

```
create user <username> identified by <password>;
grant dba to <username> with admin option;
grant select any table to <username> with admin option;
grant create user to <username> with admin option;
```

2 In the Project Explorer, expand the **Database** folder of the **eView_Sample** Project.

3 Right-click **Database Script**, and then select **Properties**.

4 On the Properties window, enter the database server information, administrator login ID, and administrator password.

5 Close the dialog.

6 Right-click **Create Company Database**, and then click **Run**.

7 Right-click **Systems**, and then click **Run**.

8 Right-click **Code List**, and then click **Run**.

12.3.4. Define the Environment

You need to create an Environment that will run both the server and client Projects.

To define the Environment

1 In the Environment Explorer, create a new Environment and rename it to "eViewEnvironment".

2 In **eViewEnvironment**, create a new LogicalHost and rename it to "eViewLogicalHost".

3 In **eViewLogicalHost**, create a new SeeBeyond Integration Server and rename it to "eViewServer".

4 In **eViewEnvironment**, create a new File External System. Specify "InboundFile" for the name, and select "Inbound File eWay" for the type.

Create a User

Following the instructions under **"Defining Security" on page 128**, create a new user. Add the **eView.Admin** and **eView.User** roles to the user.

Define the Data Source

You must define a data source in order to connect to the database. Be sure to follow this guideline listed below.

To define the data source

1 Right-click **eViewServer**, and then select **Properties**.

2 Expand **IS Configuration** until you see **JDBC DataSource Connection Pools**.

3 Right-click **JDBC DataSource Connection Pools**, and select **Create New Section**.

4 Rename the new section to "Sample DataSource".

5 Close and reopen the Properties window to refresh the fields.

6 Enter the database properties following these guidelines:

- In the **DataSource Class Name** field, enter "oracle.jdbc.pool.OracleDataSource".
- In the **Extra Properties** field, enter "DriverType=thin".
- In the **Password** and **User** fields, enter the password and logon ID of the administrator user you created for the database.
- In the **Pool JNDI Name** field, enter "CompanyDataSource".
- In the **Server Name** field, you can enter "localhost" if the database and integration server are on the same machine. Otherwise enter the name of the database server.

- In the **XA Recovery Password** and **XA Recovery User Name**, enter the system login ID and password for the database.

12.3.5. Configure the Client Connectivity Map

The client Connectivity Map is predefined. You only need to configure the eWay connection so it knows where to locate the data input file.

To define the client connectivity map

1 In the Project Explorer, expand the **eView_Sample_Client** Project.

2 Check out the Connectivity Map (CMap1), and then open the map.

3 In the Connectivity Map Editor, double-click the eWay icon between the File eWay and the Service.

The Properties window appears.

4 Change the value of the **Directory** field to the location of the sample files (where you extracted the samples .zip file).

5 Change the value of the **Multiple records per file** field to “True”.

6 Save your changes to the Repository

12.3.6. Create and Activate the Server Deployment Profile

This task links the components of the server Connectivity Map with the physical components defined for the Environment.

To create and activate the server Deployment Profile

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

2 Point to **New**, and then click **Deployment Profile**.

3 Name the profile “eViewDeployment”, and then select “eViewEnvironment” for the Environment.

4 Click **OK**.

5 In the Deployment Editor window, drag both eView application files onto **eViewServer** in the **eViewLogicHost** box.

6 In the **Deployment Editor** toolbar, click **Activate**.

7 Do not apply the changes to the logical host.

12.3.7. Create and Activate the Client Deployment Profile

This task links the components of the client Connectivity Map with the physical components defined for the Environment.

To create and activate the client Deployment Profile

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

2 Point to **New**, and then click **Deployment Profile**.

3 Name the profile "ClientDeployment", and then select "eViewEnvironment" for the Environment.

4 Click **OK**.

5 In the Deployment Editor window, do the following:

- Drag **eViewsampleJavaCollaboration1** onto **eViewServer** in the **eViewLogicalHost** box.
- Drag **eViewSample.fin -> eViewsampleJavaCollaboration1** into the **InboundFile** box.
- Drag **eViewsampleJavaCollaboration1 -> eViewCompany1** into the **Company** box.

6 In the Deployment Editor toolbar, click **Activate**.

7 Do not apply the changes to the logical host.

12.3.8. Start the Logical Host

You can start the Logical Host through a command line prompt, or you can define the properties in the **logical-host.properties** file. This section describes how to run the bootstrap using a command line.

To run the bootstrap

1 Open an MS-DOS command prompt.

2 Navigate to the Logical Host home directory, and then to **\bootstrap\bin**.

3 At the prompt, type the following:

```
bootstrap -r <server_URL> -i Administrator -p STC -e eViewEnvironment
-l eViewLogicalHost
```

where <server_URL> is the URL of the SeeBeyond Integration Server.

12.3.9. Run the Sample File

Running the sample file inserts a few company records into the eView sample database. To run this file, navigate to the directory where you extracted the sample files and change name of the **eViewSample.~in** file to **eViewSample.fin**. When the file has been processed, the file name changes back to **eViewSample.~in**.

12.3.10. Working with the EDM

You can view the records created by processing the **eViewSample.fin** file using the EDM. You can also create new records, compare records, merge records, and so on. For instructions on working with the EDM, see the *Enterprise Data Manager User's Guide*.

To log into the EDM

1 Open a Web browser.

2 In the Address field, type the following:

```
http://localhost:<port>/Companyedm
```

where <*port*> is the port number of the Web Connection Container. (This is displayed in the Enterprise Designer on the Integration Server Properties window. It is the value of the Connector Port field on the Web Container Configuration page.)

To view the new records

1. The first page to appear is a Search page. Click **Search** to display all records in the database.
2. Click on the EUID of any record to view more information.

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.

1.4 Defining Field Locations

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

1.4.1. 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 master index object structure.
- **ObjectNode**
  An *ObjectNode* represents a parent or child object defined in the master index object structure.
- **ObjectField List**
  An *ObjectField List* is a list of references to certain ObjectFields in the master index object structure.
- **ObjectNode List**
  An *ObjectNode List* is a list of references to certain ObjectNodes in the master index 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.

1.4.2. 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 parent object; that is, the field name uses the name of the parent 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**

1.4.3. 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**

Appendix B

eView Wizard Match Types

You can select a Match Type for each field defined in the eView Wizard. Each match type defines a different type of standardization, normalization, phonetic encoding, and matching logic in the Match Field file.

This appendix describes each match type and how each affects the logic in the Match Field file.

1.1 About Match and Standardization Types

For each field against which matching will be performed in the new index, you can select a *match type* in the eView Wizard. When you select a match type for a field, eView automatically adds that field to the match string and, in many cases, generates additional fields in the Object Definition that are not visible on the eView Wizard. These fields are used for matching and searching, and should not be modified.

If new fields are generated, they are automatically incorporated into the configuration files and the database script that creates the master index tables. These fields are used for standardized, normalized, or phonetic versions of the field, depending on the type of matching you choose. In addition, these fields are assigned a match type in the match string in the Match Field file. They may also be defined for standardization in the Match Field file, in which case they will also be assigned a standardization type. These types differ depending on the match engine you are using.

Note: *The match types specified in the Match Field file for the fields in the match string might not be the same as the match types you specify in the eView Wizard. Information about match types is provided in the following sections. For more information, see the implementation guide for the match engine you are using.*

SeeBeyond Match Engine

The eView Wizard match types for the SeeBeyond Match Engine are described on the following pages.

- **Person Match Types** on page 155
- **BusinessName** on page 155
- **Address** on page 156
- **Miscellaneous Match Types** on page 157

The actual standardization and match types entered into the Match Field file vary for each match type you select in the eView Wizard. The match and standardization types for each type of field are listed in the following descriptions. The match types entered into the Match Field file correspond to the match types defined in the match configuration file, **MatchConfigFile.cfg**.

Person Match Types

The Person match types include PersonLastName, PersonFirstName, and PersonMiddleName. These match types are used to normalize and phonetically encode name fields for person matching. For each field with one of these match types, the eView Wizard adds two fields to the Object Definition for phonetic and standardized versions. If you select a field for blocking with a person match type, the phonetic version of the name is added to the blocking query. The following fields are created when you select one of the Person match types for a field (<*field_name*> refers to the name of the field selected for Person matching).

- ***<field_name>*_Std**
  This field contains the normalized version of the name.
- ***<field_name>*_Phon**
  This field contains the phonetic version of the name.

The corresponding standardization type and match type in the Match Field file for each of the Person match types are listed in Table 12.

Table 12 Person Name Standardization and Match Types for the SeeBeyond Match Engine

eView Wizard Match Type	Match Field File Standardization Type	Match Field File Match Type
PersonLastName	PersonName	LastName
PersonFirstName	PersonName	FirstName
PersonMiddleName	PersonName	MiddleName

BusinessName

BusinessName matching is used to parse, normalize, and phonetically encode a business name for matching on company names. BusinessName matching adds several fields to the Object Definition and to the match string. If you select a business name field for blocking, each parsed business name field is added to the blocking query. The corresponding standardization type in the Match Field file for fields selected for BusinessName matching is **BusinessName**. The actual match type assigned to each field varies depending on the type of information in each field.

The fields created when you select the BusinessName match type for a field are listed below along with their corresponding Match Field match types (<*field_name*> refers to the name of the field selected for BusinessName matching).

Important: *Only specify this type of matching for one business name field; otherwise, the eView Wizard will create duplicate entries in the object structure. If more than one field contains the business name, you can define the additional fields in the* *standardization structure in the Match Field file after the eView Wizard creates the configuration files.*

- ***<field_name>*_Name**
  This field contains the parsed and normalized version of the business name. This field is added to the match string, and the match type assigned to this field is **PrimaryName**.
- ***<field_name>*_NamePhon**
  This field contains the phonetic version of the business name. This field is not added to the match string.
- ***<field_name>*_OrgType**
  This field contains the parsed organization type of the business name. This field is added to the match string, and the match type assigned to this field is **OrgTypeKeyword**.
- ***<field_name>*_AssocType**
  This field contains the association type for the business. This field is added to the match string, and the match type assigned to this field is **AssocTypeKeyword**.
- ***<field_name>*_Sector**
  This field contains a location for the business, such as a floor number. This field is not added to the match string, but if you add it to the match string, assign it a match type of **SectorTypeKeyword**.
- ***<field_name>*_Alias**
  This field contains an alias for the business name. This field is not added to the match string.
- ***<field_name>*_Url**
  This field contains the business' web site URL. This field is added to the match string, and the match type assigned to this field is **Url**.

Address

Address matching is used to parse, normalize, and phonetically encode an address for matching or standardizing address information. Address matching adds several fields to the Object Definition and to the match string. If you select an address field for blocking, the parsed fields are added to the blocking query. The corresponding standardization type for fields selected for Address matching is **Address**. The actual match type assigned to each field varies depending on the type of information in each field.

The fields created when you select the Address match type for a field are listed below along with their corresponding Match Field match types (<*field_name*> refers to the name of the field selected for BusinessName matching).

Important: *Only specify this type of matching for one street address field; otherwise, the eView Wizard will create duplicate entries in the object structure. If more than one field contains the street address, you can define the additional fields in the standardization structure in the Match Field file after the eView Wizard creates the configuration files.*

- ***<field_name>*_HouseNo**
  This field contains the parsed street number of the address. This field is added to the match string, and the match type assigned to this field is **HouseNumber**.
- ***<field_name>*_StDir**
  This field contains the parsed and normalized street direction of the address. This field is added to the match string, and the match type assigned to this field is **StreetDir**.
- ***<field_name>*_StName**
  This field contains the parsed and normalized street name of the address. This field is added to the match string, and the match type assigned to this field is **StreetName**.
- ***<field_name>*_StPhon**
  This field contains the phonetic version of the street name. This field is not added to the match string.
- ***<field_name>*_StType**
  This field contains the parsed and normalized street type of the address, such as Boulevard, Street, Drive, and so on. This field is added to the match string, and the match type assigned to this field is **StreetType**.

If you want to search on street addresses but do not want to use these fields for matching, select the Address match type for only one street address field in the eView Wizard. When the wizard is complete, you can remove the address fields from the match string in the Match Field file.

Note: *If you want to search on street addresses but do not want to use these fields for matching, select the **Address** match type for only one street address field in the eView Wizard. When the wizard is complete, you can remove the address fields from the match string in the Match Field file.*

Miscellaneous Match Types

Several additional eView Wizard match types are defined for the SeeBeyond Match Engine. These match types are used to indicate matching on a string, date, or number field other than those described above, or to indicate matching on a field that is a single character (such as the gender field, which might accept “F” for female or “M” for male). These match types do not define standardization for the specified field, and do not add any fields to the Object Definition. If you select a field for one of these types of matching, it is added to the match string with a match type of **String**, **Date**, **Number**, **Exac**, **Pro**, or **Character**.

INTEGRITY Match Engine

The eView Wizard match types for INTEGRITY are described on the following pages.

- **Person Match Types** on page 158
- **Usfname and Uslname Match Types** on page 158
- **Business** on page 158
- **Address** on page 159

- **Area** on page 160
- **Custom1 Match Types** on page 160

The actual standardization and match types entered into the Match Field file vary for each match type you select in the eView Wizard. The match and standardization types for each type of field are listed in the following descriptions.

The match types entered into the Match Field file correspond to the INTEGRITY rule sets as defined in the **ServerCfg.cfg** and **MatchCfgs.cfg** files. For detailed information about the fields that are automatically created when a specific match type is specified, see *Implementing Ascential INTEGRITY with eView Studio*.

Person Match Types

Person match types include Person.LastName, Person.FirstName, Person.DOB, Person.SSN, and Person.Gender. Person.LastName and Person.FirstName are used to normalize and phonetically encode a person's first, last, and middle name for matching. For each field with a Person.LastName or Person.FirstName match type, the eView Wizard adds the following two fields to the Object Definition (<*field_name*> refers to the name of the field selected for Person matching).

- ***<field_name>*_Std**
  This field contains the normalized version of the name.
- ***<field_name>*_Phon**
  This field contains the phonetic version of the name.

When the Person.DOB, Person.SSN, or Person.Gender match types are specified for a field, no additional fields are created but the field is added to the match string. The corresponding standardization type in the Match Field file for fields selected for Person matching is **UI**. The match type assigned to each field is **Person**.

Usfname and Uslname Match Types

Name match types include Usfname and Uslname. These match types are used to normalize and phonetically encode a person's first, last, and middle name for storing the values in the database. For each field assigned one of these match types, the eView Wizard adds the following two fields to the Object Definition for phonetic and standardized versions (<*field_name*> refers to the name of the field selected for matching).

- ***<field_name>*_Std**
  This field contains the normalized version of the name.
- ***<field_name>*_Phon**
  This field contains the phonetic version of the name.

The corresponding standardization type in the Match Field file for fields selected for Usfname name matching is **USFNAME**. For Uslname matching, the standardization type is **USLNAME**. The corresponding match type in the Match Field file for fields selected for these types of matching is **Person**.

Business

Business matching is used to parse, normalize, and phonetically encode a business name for matching on company names. Business matching adds several fields to the

Object Definition, but only adds the unparsed field specified for matching to the match string. The fields added to the Object Definition include:

- ***<field_name>*_Name**
  This field contains the parsed version of the business name.
- ***<field_name>*_NamePhon**
  This field contains the phonetically encoded version of the business name.
- ***<field_name>*_Type**
  This field contains the parsed version of the business type.
- ***<field_name>*_StandType**
  This field contains the normalized version of the business type.

  where <*field_name*> refers to the name of the field selected for Business matching.

The corresponding match type and standardization type in the Match Field file for fields selected for Business matching is **Business**.

Important: *Only specify this type of matching for one business name field; otherwise, the eView Wizard will create duplicate entries in the object structure. If more than one field contains the business name, you can define the additional fields in the standardization structure in the Match Field file after the eView Wizard creates the configuration files.*

Address

Address matching is used to parse, normalize, and phonetically encode addresses for matching and standardization on address information. Address matching adds several fields to the Object Definition, but only adds the field specified for matching to the match string. The fields added to the Object Definition include:

- ***<field_name>*_HouseNo**
  This field contains the parsed house number component of the street address.
- ***<field_name>*_StName**
  This field contains the parsed and normalized street name component of the street address.
- ***<field_name>*_StDir**
  This field contains the parsed street direction component of the street address.
- ***<field_name>*_StType**
  This field contains the parsed street type component of the street address.
- ***<field_name>*_StPhon**
  This field contains the phonetically encoded version of the street name.

  where <*field_name*> refers to the name of the field selected for Address matching.

The corresponding standardization type and match type in the Match Field file for fields selected for Address matching is **Address**. If you want to search on street addresses but do not want to use these fields for matching, select the Address match type for only one street address field in the eView Wizard. When the wizard is complete, you can remove the address fields from the match string in the Match Field file.

Important: *Only specify this type of matching for one street address field; otherwise, the eView Wizard will create duplicate entries in the object structure. If more than one field contains the street address, you can define the additional fields in the standardization structure in the Match Field file after the eView Wizard creates the configuration files.*

Area

Area matching is used to parse, normalize, and phonetically encode city, state, and other address information for matching and standardization. Area matching adds several fields to the Object Definition, but only adds the unparsed field specified for matching to the match string. he fields added to the Object Definition include:

- ***<field_name>*****_City**
  This field contains the parsed city name component of the local area.
- ***<field_name>*****_State**
  This field contains the parsed state abbreviation component of the local area.
- ***<field_name>*****_PostCode**
  This field contains the parsed postal code component of the local area.
- ***<field_name>*****_PostExt**
  This field contains the parsed postal code extension component of the local area.
- ***<field_name>*****_Country**
  This field contains the country code component of the local area.
- ***<field_name>*****_CityPhon**
  This field contains the phonetically encoded version of the city name.

  where *<field_name>* refers to the name of the field selected for Area matching.

The corresponding standardization type and match type in the Match Field file for fields selected for Area matching is **Area**.

Important: *Only specify this type of matching for one local area field; otherwise, the eView Wizard will create duplicate entries in the object structure. If more than one field contains the local area, you can define the additional fields in the standardization structure in the Match Field file after the eView Wizard creates the configuration files.*

Custom1 Match Types

The Custom1 match types include Custom1.PartName, Custom1.PartType, and Custom1.PartID. This is a customized match and standardization type, designed to be used with the Custom1 INTEGRITY rule set. When you specify the Custom1.PartName match type for a field, one field is added to the Object Definition. This field is named ***<field_name>*****_Phon**, where *<field_name>* refers to the name of the field selected for Custom1.PartName matching. This field contains the phonetically encoded version of the field. For more information about working with this type, see *Implementing Ascential INTEGRITY with eView Studi*o.

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 eView 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 entity.

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 eView 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 eView object. Also known as the *element path*.

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

eView Manager Service
An eView component that provides an interface to all eView components and includes the primary functions of the master index. 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 eView 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 eView configuration file that specifies any custom Java classes that perform field validations when data is processed.

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

master index
A database application that stores and cross-references information on specific objects in a business organization, regardless of the computer system from which the information originates.

Match Field File
An eView configuration file that defines normalization, parsing, phonetic encoding, and the match string for an instance of eView. 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 object.

Matching Service
An eView 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*.

merge
To join two object profiles or system records that represent the same entity into one object profile.

merged profile
See *non-surviving profile*.

non-surviving profile
An object profile that is no longer active because it has been merged into another object 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 an object profile, such as a company object, which contains all of the demographic data about a company, or an address object, which contains information about a specific address type for the company.

object profile
A set of information that describes characteristics of one enterprise object. A profile includes identification and other information about an object and contains a single best record and one or more system records.

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 eView 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 an entity'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 an entity'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 objects in the master index and that shares this information with the master index (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 entity are part of that entity'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 eView 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